Inductive Verification of Cryptographic Protocols Based on Message Algebras – Trace and Indistinguishability Properties –

Lassaad Cheikhrouhou

A dissertation submitted towards the degree
Doctor of Engineering (Dr.-Ing.)
of the Faculty of Mathematics and Computer Science
of Saarland University

UNIVERSITÄT
DES
SAARLANDES

Saarbrücken, 2022

Date of Colloquium:	19 May 2022
Dean of Faculty:	Prof. Dr. Jürgen Steimle
Chair of the Committee:	Prof. Dr. Sebastian Hack
Reporters	
First Reviewer:	PD Dr. Werner Stephan
Second Reviewer:	Prof. Dr. Christoph Weidenbach
Academic Assistant:	Dr. Andreas Nonnengart

To my parents,
Khadija & Mohammed.

Deutsche Zusammenfassung

Seit 1981 wurden zahlreiche formale Methoden zur Analyse kryptographischer Protokolle entwickelt und erfolgreich angewendet. Trotz vieler Verbesserungen, beschränkt sich der Anwendungsbereich gerade induktiver Verfahren auf das einfache *enc-dec* Szenario (Entschlüsseln hebt Verschlüsseln ab) und auf Standardeigenschaften (Vertraulichkeit und Authentifizierung).

In dieser Arbeit erweitern wir den Anwendungsbereich der werkzeug-unterstützten induktiven Methode auf Protokolle mit algebraisch spezifizierten kryptografischen Primitiven und auf Ununterscheidbarkeitseigenschaften wie die Resistenz gegen Offline-Testen. Eine Axiomatisierung von Nachrichtenstrukturen, abgeleitet aus einem konstruktiven Modell (Termersetzung), liefert die Basis für die Definition rekursiver Funktionen und induktives Schließen (partielle Ordnungen, Fallunterscheidungen). Eine neue Beweistechnik für Vertraulichkeitseigenschaften verwendet rekursive Testfunktionen, die beweisbar korrekt bzgl. eines induktiv definierten Angreifermodells sind. Die Formalisierung von Ununterscheidbarkeitseigenschaften durch generische Ableitungen und ein zentrales Theorem erlauben eine Reduktion auf Trace-Eigenschaften.

Die allgemeinen Aspekte unserer Techniken werden zusammen mit zwei vollständig ausgearbeiteten realen Fallstudien, PACE und TC-AMP, diskutiert, die für den deutschen Personalausweis entwickelt wurden. TC-AMP gehört sicher zu den komplexesten algebraisch spezifizierten Protokollen, die formal verifiziert wurden. Insbesondere, sind uns keine Ansätze bekannt, die vergleichbare Fälle behandeln.

Abstract

Since 1981, a large variety of formal methods for the analysis of cryptographic protocols has evolved. In particular, the tool-supported inductive method has been applied to many protocols. Despite several improvements, the scope of these and other approaches is basically restricted to the simple *enc-dec* scenario (decryption reverts encryption) and to standard properties (confidentiality and authentication).

In this thesis, we broaden the scope of the inductive method to protocols with algebraically specified cryptographic primitives beyond the simple *enc-dec* scenario and to indistinguishability properties like resistance against offline testing. We describe an axiomatization of message structures, justified by a rewriting-based model of algebraic equations, to provide complete case distinctions and partial orders thereby allowing for the definition of recursive functions and inductive reasoning. We develop a new proof technique for confidentiality properties based on tests of regular messages. The corresponding recursive functions are provably correct wrt. to an inductively defined attacker model. We introduce generic derivations to express indistinguishability properties. A central theorem then provides necessary and sufficient conditions that can be shown by standard trace properties.

The general aspects of our techniques are thoroughly discussed and emphasized, along with two fully worked out real world case studies: PACE and TC-AMP are (to be) used for the German ID cards. To the best of our knowledge TC-AMP is among the most complex algebraically specified protocols that have been formally verified. In particular, we do not know of any approaches that apply formal analysis techniques to comparable cases.

Acknowledgments

First and foremost, I would like to thank my advisor, Werner Stephan. He continued after retirement to examine carefully my results. His encouragement and the long phone calls we spent discussing his feedback made it possible to complete this thesis.

I would also like to thank Christoph Weidenbach for reviewing my thesis, and Sebastian Hack for acting as the chair of the examination board.

I am grateful to Markus Ullmann and Frank Koob from the Federal Office on Information Security (BSI). They have been initiators and reviewers of many BSI projects I was involved in at the DFKI. This gave me the opportunity to gain expertise in formal analysis of cryptographic protocols. The topics of this thesis arose from project works on password protocols developed by the BSI for the German ID card.

I would like to thank Jörg Siekmann. He gave me the chance to do research in the field of Proof Planning, which facilitated the development of the proof heuristics used in the VSE tool for protocol verification.

Finally, I would like to thank all my former colleagues at the DFKI in the Safe and Secure Systems research group (formerly the Deduction and Multiagent Systems research department), who always provided a good working atmosphere. My special thanks are to Andreas Nonnengart for proofreading the introduction and for serving as the academic assistant on my examination board, to Georg Rock for his cooperation in many BSI projects, to Bruno Langenstein for his extension to the proof data structure of the VSE prover, which permitted to use control knowledge for the composition of proof heuristics, to Christopher Krauß, Oliver Keller, Roland Vogt, Carsten Tichy and to Stefan Denne for their excellent team spirit.

No words are enough to express my gratitude to my father, Mohammed, and my mother, Khadija, for their kindness, sacrifices, and support. I want also to thank my parents in law, Abdelwahed and Naima, for their moral support. My particular thanks are to my family members, my wife Imen and my source of motivation and children Nour, Aymen and Hedi, for their endless patience and sacrifices. Last but not least, I would like to thank my siblings, Maher, Fathia, Sonia and Hamadi for their excellent care of our parents.

Contents

Part I

Background

Chapter 1

Introduction and Related Works

In this thesis we present substantial extensions to the inductive verification of cryptographic protocols. They permit for *machine-checked* security proofs (of protocols) with algebraically specified cryptographic primitives beyond the simple encryption-decryption (*enc-dec*) scenario. We not only broaden the scope of inductive verification techniques to (more) involved algebraic properties of cryptographic primitives, but also to indistinguishability properties like resistance against offline testing attacks, anonymity, and privacy.

The main contributions are

- extension of algebraic specifications of cryptographic primitives to theories that allow for inductive proofs,
- new recursively defined security check functions that are sound wrt. the given inductive attacker model,
- development of a technique to prove resistance against offline testing attacks and other indistinguishability properties,
- and application of the general approach to two real-world protocols.

The general methodology described in this thesis resulted from the tool-based verification of two protocols, PACE and TC-AMP [101], but applies to a wide range of protocols and properties. PACE (protocol for *Password Authenticated Connection Establishment*) and its fall-back solution TC-AMP (protocol for *Terminal-Card Authenticated key agreement via Memorable Passwords*) were developed by the (German) Federal Office for Information Security (BSI) primarily for deployment in electronic identification cards (eIDs), [57]. PACE is recommended as a substitute for the Basic Access Control (BAC) protocol in machine readable travel documents (MRTDs), [66], and recently also in the smart metering scenario, [56].

PACE and TC-AMP aim at the *strongest* security objectives in the *family of password protocols*, [17, 69, 105, 73]. Besides mutual-authentication and forward secrecy of session keys, this includes *resistance against offline password testing attacks*, [17]. The latter is a typical indistinguishability property and we achieved its verification with a generic approach, applicable to all kinds of indistinguishability properties, including

- fairness (no leak of early vote results that may influence subsequent voters) and privacy (no vote can be linked to a voter) in e-voting protocols, [62],
- and various anonymity properties in
  - anonymous routing protocols, [85],
  - e-payment protocols (anonymity of payer's identity), [106],

– and in anonymous authentication protocols, [2].

In our verification of PACE and TC-AMP as concrete instantiations of the general approach we achieved to treat algebraic properties that are to the best of our knowledge out of scope of state-of-the-art protocol verification approaches. So, we expect that our results open the door to the verification of many other security protocols that have not yet been approached because of complex algebraic properties.

Before we outline some more details about the contributions of the thesis in Sec. 1.2, we relate our work to other approaches in the next section: After a brief description of the underlying attacker models in Sec. 1.1.1, we provide in Sec. 1.1.2 a more precise description of the above mentioned protocol properties. We sum up in Sec. 1.1.3 other related tool-supported protocol analysis approaches before we briefly present our field of inductive protocol verification.

1.1 Assessment of Protocols

It was pointed out already in 1978 by Needham and Schroeder in [79] that cryptographic "protocols ... are prone to extremely subtle errors that are unlikely to be detected in normal operation". Furthermore, they estimated that "the need for techniques to verify the correctness of such protocols is great", calling attention to a new research problem. Three years later, Dolev and Yao published a paper [48], which is perhaps the most cited one in this research field. It is *on the decidability* of the verification problem for two families of protocols ("cascade" and "name-stamp") that aim at the confidential transmission of application data between two participants using asymmetric encryption. Since then, a large variety of mathematically sound analysis methods have evolved. The majority of tool-supported approaches rely on (variants of) the so-called Dolev and Yao (DY) attacker model.

1.1.1 DY Models

DY (attacker) models can be characterized by the following (common) features:

- The messages exchanged are modeled as *symbolic expressions* representing message parts and cryptographic primitives.
- The cryptographic primitives are defined as operations that behave according to certain rules.
- Arbitrary many parallel protocol executions are interleaved with the actions of an *active attacker* who controls the network.
- The attacker extracts message parts and fakes new messages using a subset of the cryptographic primitives.

In an *algebraic setting* operations are modeled by equations. For instance, standard encryption and decryption primitives are given by function symbols *enc* and *dec* that satisfy the equation

$$dec(k^{-1}, enc(k, m)) = m, \tag{1.1}$$

where k^{-1} denotes the inverse of a key k and m is an arbitrary message.

Equation (1.1) first of all expresses the fact that knowledge of k^{-1} allows to reveal m which is meant to be protected. Moreover, in the absence of additional laws, the restriction (of the attacker) to *equational reasoning excludes other ways* to obtain m.

Clearly, the choice of equations that give raise to admissible computations is highly critical. It has to be justified by models where guessing and arbitrary computations are allowed, [11, 43]. Security in these models is investigated *relative to* probabilistic polynomial-time bounded Turing (PPT) attacker machines and probability thresholds.

The major contribution of this thesis is a general approach to *inductively* verify protocols in algebraically defined DY-models. State-of-the-art inductive verification techniques, see Sec. 1.1.3.2, only implicitly use the basic equations and it is hard to see how they could be easily extended to protocols like PACE and TC-AMP. Our approach goes far beyond simple equations like (1.1). The specification of TC-AMP is based on 15 equations, where 13 of them are on primitives in elliptic curve cryptography. They in particular include left distributivity, which cannot be handled by advanced protocol verification approaches dealing with algebraic properties of cryptographic primitives, [89] (see Sec. 1.1.3.1.2).

1.1.2 Scope of Verification

To demonstrate the scope of our contribution and to compare it to other approaches, we classify properties handled by state-of-the-art formal analysis techniques. The discussion is based on trace models, where a *trace* is an interleaving of (message sending) actions carried out

- by *ordinary participants* following the rules of their *role*,
- and by *the attacker* according to her ability to generate (*faked*) *messages* out of the knowledge obtained by observing *previous traces*.

The (immediately) *observable intruder knowledge* (*ik*) consists of all messages obtained (without further derivations) from the actions of a trace. For technical reasons we treat *ik* as a *list* of messages.

A *thread* is a projection to actions originated by a given participant.

Trace models have been used in other areas of formal methods. For example in [58] non-interference properties in workflow systems are verified in trace models. The distinguishing features of protocol traces are the use of cryptographic primitives and the inclusion of an explicit attacker whose abilities are defined wrt. assumptions on these cryptographic primitives.

1.1.2.1 Confidentiality

The vast majority of approaches (only) handles confidentiality and authentication properties. Confidentiality guarantees that certain (sub-) messages called *secrets* of a trace cannot be computed by the attacker from *ik*. For example, in Paulson's approach the possible computations are given by *analz*, [82]. In our algebraic setting computations of the attacker are given by the application of available operations to elements of *ik*. A generic derivation or *derivation strategy* δ can be seen as a fixed composition of cryptographic primitives applied to a fixed position selection of elements in a knowledge base *ik* that is given as an argument. We denote $\delta(ik)$ the result of applying δ to *ik*. A secret s is confidential if for all traces tr and all derivation strategies δ the result $\delta(ik_{tr})$ cannot be evaluated to s by equational reasoning, where ik_{tr} denotes the immediately observable messages by the attacker from tr.

Forward secrecy (of session keys) considers the case where a long-term secret used in a protocol run is disclosed afterwards. It guarantees that the confidentiality of session keys established before the disclosure of a long-term secret is preserved.

1.1.2.2 Authentication

Authentication properties provide guarantees about the origin of messages received by a participant at a certain state of her thread. The participant may then safely assume that these messages are part of the threads of ordinary participants (peers) and that the threads have reached a certain current state. Authentication properties come as *implications* where

the premise is about the state of the given participant and where the conclusion describes the corresponding states of the threads of certain peers.

Authentication implications can be easily extended with application-specific details about the states of the involved participants. For example, we have achieved the verification of authentication properties providing guarantees about misuse counters, [31]. This demonstrates that our treatment of authentication implications applies as well to application-specific properties that are defined from the perspective of particular participants, providing guarantees about the states of their peers. Such application-specific properties include for instance, security objectives of e-payment protocols like non-repudiation (of authorized payments).

1.1.2.3 Resistance against Offline Testing Attacks

Relatively few approaches were extended to properties like resistance against offline testing attacks. In offline testing attacks, the (active) attacker observes ik in a first step. After that she uses ik to test *candidates* for a secret hidden in ik by an *offline procedure*. Basically, these attacks use a predicate P on a list $\varpi.ik$, where

$P(\varpi.ik) \Leftrightarrow$ ''The candidate ϖ is the secret hidden in the trace where ik results from''.

In the algebraic approach the basic constituent of computations of predicates is the comparison of the result of (two) derivation strategies. Therefore, if for all derivation strategies δ, δ', all observations ik from protocol runs, all secrets of interest π used in ik, and all candidates ϖ we have

$$\delta(\pi.ik) = \delta'(\pi.ik) \Leftrightarrow \delta(\varpi.ik) = \delta'(\varpi.ik), \tag{1.2}$$

then offline testing attacks are excluded. In other words, *no* knowledge base $kb = \pi.ik$ with a genuine secret π *can be* effectively *distinguished* from a knowledge base $kb' = \varpi.ik$ with a false candidate ϖ.

Resistance against offline testing attacks is relevant for password protocols to protect the passwords [17, 68, 40], and for e-voting protocols to achieve fairness [72]. In a context where $kb = \pi.ik$ and $kb' = \varpi.ik$ are emitted synchronously by two processes communicating with some environment, resistance against offline testing attacks is an instance of *static equivalence* [40].

1.1.2.4 Further Indistinguishability Properties

In another family of indistinguishability properties, the above equivalence (1.2) is about knowledge bases kb and kb' corresponding to the immediately observable messages from two traces: kb belongs to an arbitrary trace tr_{kb} and kb' to a trace $tr_{kb'}$ that results from a *simulation* of tr where certain property-specific parameters are changed or swapped. Except of sent messages, simulation means that the involved participants and their actions are preserved, including the generic derivations used by the attacker for the generation of fake messages.

For instance, *privacy in e-voting* protocols, i.e. the unlinkability of a voter V and her vote vo, is defined for arbitrary traces tr_{kb} having another voter V' with a different vote vo'. The simulation resulting in $tr_{kb'}$ swaps vo and vo', [72, 10]. Privacy requires that the knowledge bases kb and kb' obtained from tr_{kb} and respectively $tr_{kb'}$ are indistinguishable.

In process algebras, unlinkability of voter and her vote is an instance of *observational equivalence* [41], which is equivalent in the context of protocol verification to *trace equivalence* and *labeled bi-similarity*, [3, 29].

Trace equivalence permits also to formalize anonymity, generally from the perspective of weaker adversary models. It was used, for instance in [74] to verify two anonymous routing protocols: Anonymity of the originating sender in the Crowds protocol, [86], (from

the perspective of the (end) server) and unlinkability of a message and its originator in the Onion Routing protocol, [98], (from the perspective of a *passive* DY attacker).

All trace equivalence properties reduce to the indistinguishability of corresponding knowledge bases kb and kb', which is *clearly* defined according to (1.2) in Sec. 1.1.2.3.

1.1.3 Tool Support

Our approach belongs to protocol verification approaches allowing tool support, *in order to ensure soundness and to reach more automatization*. The challenge for these approaches is to deal with infinite sets of traces in trace models mentioned above, consisting of

- an unbounded number of interleaved threads and
- an unbounded number of (faked) attacker messages.

We distinguish two basic approaches:

1. Automatic techniques *execute* (forward or backward) the protocol rules and intruder actions. In addition, the reasoning capabilities of the attacker have to be implemented.
2. Interactive *deduction* techniques reason *inductively* about possible traces and results of recursively defined functions that model the ability of the attacker to deduce information from a set of (immediate) observations.

1.1.3.1 Automatic Tools

1.1.3.1.1 First Generation Tools: Automatic tools like Casper/FDR [46, 47, 49] and the back-ends OFMC [14], SAT-MC [5, 6], and CL-AtSe [100] of the AVISPA environment [102, 4], explore a finite search space to find *bugs*. The finite search space results from bounds on the number of threads and attacker messages. For real-world protocols, the bounds are often very restrictive to avoid an explosion of the search space.

In addition to the generation of interleaved threads, these tools often apply implemented rewriting procedures for decidable intruder deduction problems. Only few tools allow for message algebras beyond the simple *enc-dec* scenario. There are generic approaches for the class of subterm-convergent equational theories, [1], with a compile mechanism of such given equations in deduction rules. The remaining approaches use built-in techniques with predefined deduction rules integrating exclusive-or (xor), exponentiation (for Diffie-Hellman (DH) exchange) and/or associative pairing, [37, 8].

Tools of this category have been applied to confidentiality and certain classes of authentication properties. Some of them implement methods to search for bugs wrt. specific indistinguishability properties, [29, 34]. In this context, we mention that the privacy property in e-voting protocols could be analyzed completely automated only by the tool AKiss, [29]. It is a special-purpose tool that handles exclusively certain kinds of indistinguishability properties. This tool was applied for the privacy property of two e-voting protocols (FOO and Okamoto, [62, 81]).

1.1.3.1.2 Second Generation Tools: Automatic tools like NPA-Maude [54] (a successor of the NRL Protocol Analyser [76]), Athena [96], Scyther [44] and Tamarin [77], rely (more) on techniques for automated theorem proving. Termination without detecting a bug can be seen as a verification. However, since they employ techniques to *soundly* avoid infinite search paths, termination is not guaranteed.

Regarding message algebras beyond the simple *enc-dec* scenario, the generic approach of NPA-Maude is worth mentioning. It is based on the separation of equations into a set A of axioms with a finitary unification algorithm and a set R of rewrite rules that is both

convergent and satisfies the *finite variant property* (FVP) modulo A, [38]. FVP is a generalization of subterm-convergence, and guarantees for a given term t the existence of a finite set of term patterns $\{t_1, \ldots, t_n\}$ that subsume the infinite equivalence class $[t]_{R \cup A}$. The Maude framework uses built-in procedures to handle several equational theories relevant for protocol verification. They implement both the computation of the term patterns and unification modulo A, as required by the inference engine. Yet, the built-in unification procedure deals only with three theories: C (commutativity), AC (C plus associativity), and ACU (AC plus identity constant), [52].

Note that FVP is not satisfied by the TC-AMP equations, as these include one-side distributivity, [89].

Tamarin handles also user-defined cryptographic primitives where it allows for many (but not all) FVP equational theories, [90, 50]. Furthermore, built-in primitives were successively integrated in Tamarin, [99]. This includes the primitives of a multiplicative abelian group together with exponentiation (for Diffie-Hellman) and with primitives of bi-linear groups (scalar multiplication of a point and the application of bi-linear map to two points), [91, 92], xor, [51], and primitives of more refined models of Diffie-Hellman groups to capture attacks based on small subgroups and invalid curve points, [45]. The generic and the built-in approaches of Tamarin do/seem not to cover one-side distributivity, which belongs to the TC-AMP equational theory.

The tools mentioned above have been applied to confidentiality and (certain classes of) authentication properties. Indistinguishability properties are addressed to the best of our knowledge only by NPA-Maude and Tamarin, [88, 13]. Both tools have adopted the approach of ProVerif (see below) to define indistinguishability properties "based on a notion of diff-equivalence, and therefore suffer from the same drawbacks", [63]. The (first) case studies by NPA-Maude present only found attacks and no succeeded verifications, because of "non-termination due to state space explosion", [88]. This could be noticed also for Tamarin, [51].

1.1.3.1.3 Automatic Tools based on Approximation: Abstracting protocol rules (and sometimes also equations in message algebras) leads to *over-approximation*. In this approach bugs detected by a terminating computation do not necessarily occur in the original protocol, while termination without finding a bug is a verification of the corresponding property. The most prominent tool is ProVerif, [23], where the approach of Weidenbach, [103], to verify protocols by resolution on Horn clauses is adopted.

Basically, approximations are tailored for specific kinds of properties: For instance, the tool TA4SP, [25], used as a back-end in AVISPA, [102, 4], transforms protocol specifications in HLPSL, [36], to approximations that are suitable *only* for confidentiality. Similarly, the first approximation implemented in ProVerif applies only to confidentiality, [20]. To deal with authentication, ProVerif extended the approximation with new predicates to model chronological order between events, [21].

To the best of our knowledge, only ProVerif has been applied to indistinguishability properties, using the so-called diff-equivalence for their definition. For that purpose, the specification language (a process algebra) was extended with a specific binary function symbol "*diff*" to express twin executions of the protocol with different tuples of parameters, [24]. The over-approximation employ predicates that encode information about the twin executions using a sequence of quasi twin arguments. This approach implemented in ProVerif (version 1.85) does not terminate in the verification of the resistance property for a simplified version of PACE, [78]. Recently, a new version of ProVerif generates a more adjusted approximation to avoid false attacks due to "else" paths, [35]. This allowed to prove anonymity for a simplified model of one of the authentication protocols in [2]. "Despite ... improvements on diff-equivalence checking [35] intended to prove unlinkability of the BAC protocol (used in e-passport), ProVerif still cannot be used off-the-shelf to establish unlinkability properties, and therefore cannot conclude on the case studies pre-

sented in" [63]. These case studies could be handled by a reduction of the corresponding privacy properties (unlinkability and anonymity) to two sufficient conditions, a kind of an adjusted diff-equivalence (frame opacity) and well-authentication, which could be verified by ProVerif, [63]. Noteworthy, the presented approach discussed the verification of PACE focusing on privacy in the e-passport scenario, without referring to resistance against offline password testing. The verified privacy property requires to distinguish the final message of the reader from that of the initiator. This necessitates to extend them with tags, as we have equally proposed to avoid few anomalies identified in our verification of (the first version of) PACE, [101].

ProVerif allows for message algebras beyond the simple *enc-dec* scenario that can be specified in form of rewrite rules and equations, [22, 23]. The equations are compiled into specific rewrite rules that are intended to satisfy the FVP. Rewrite rules given in the specification and those resulting by compilation are translated into Horn clauses as part of the approximation.

1.1.3.2 Inductive Reasoning

The inductive approach to protocol verification goes back to the pioneering work of L. Paulson, [82]. The basic technique is induction on the length of traces. In the standard approach messages are modeled as freely generated data types treating *enc* as a constructor. The corresponding selector(s) must not be confused with the *dec* function in message algebras. Instead, the reasoning abilities of the attacker are given by a function $analz$ defined recursively. The function $analz$ defines all possible ways the attacker might deduce new knowledge items from immediate observations. It *implicitly* models equation (1.1) for decryption, where extraction of the encrypted message makes it necessary to have the right key at hand.

It is unclear how Paulson's $analz$ function could be defined for protocols like TC-AMP, not to speak of a *justification* for such a new version (of $analz$). Instead, our approach uses a sound extension of equational specifications that permits for recursively defined functions allowing us to do without $analz$. In particular, they are *provably* sound wrt. the *algebraically defined* intruder knowledge.

Paulson implemented his method in Isabelle/HOL, [80]. Isabelle as well as systems like KIV, [53], Coq, [67], and VSE, [65], which was used to implement the method described in this thesis, allow for sound extensions of a basic deduction mechanism. Beyond the experimental use of tools, this is a crucial issue.

All the above mentioned tools are interactive but allow for an automatic extension of elementary steps including proof search that can be tailored to specific applications without touching the kernel inference machine. In particular, the major facilities that we implemented in VSE for inductive protocol verification are domain-specific proof heuristics, [30]. These heuristics allow for a relatively high degree of automation: Over 95 % of the required interactions in a step-by-step mode are saved up in the verification of real-world protocols based on the simple *enc-dec* scenario, [31]. *Currently*, more than 80 % of the required interactions are saved up in the verification of protocols like PACE, where other cryptographic primitives specified by equations are used, [33].

Fully automatic inductive theorem provers, like CLAM [27], and INKA [7], have not been applied intensively to protocol verification. The reason for this could be twofold: Firstly, many protocol properties such as authentication are very complex compared to the theorems typically handled by these tools. Second, the proofs of the main protocol properties rely on a relatively high number of lemmas. The majority of these lemmas cannot be generated automatically without any guidance by a user or by additional domain-specific knowledge.

The inductive method in the simple *enc-dec* scenario has been applied to confidentiality, forward secrecy, and to all kinds of authentication properties [82, 16, 83, 15]. Few works

tried to extend the scope to indistinguishability properties: The approach in [74] on the verification of anonymity for two routing protocols (see Sec. 1.1.2.4) proposes a formal definition for sender anonymity and for unlinkability of a message and its originator based on the notion of trace equivalence. This relies on a tailored definition for indistinguishable message sequences where failed decryption by *analz* is incorporated. It does not apply to resistance against offline testing attacks.

The paper [28] is about the verification of the privacy property in e-voting protocols, i.e. the unlinkability between voter and her (valid) vote. This property is formalized based on the association of honest sender and intended receiver with the message and with every message part that can be extracted by *analz* using the available keys. Associations, as sets containing message parts with associated participants, are gathered through a function *aanalz*. They are then extended, based on the transitivity of associations, using an additional function *asynth*. Unlinkability is finally defined by $asynth(aanalz(spies(tr)))$. It is unclear whether and how this definition is related to the indistinguishability notion.

It seems as if there are only a very few approaches that extend the scope of the inductive method to algebraic protocol specifications. Weeks of Internet search only led to the work, reported in [12], about the verification of a protocol family for distance bounding, [95]. Here, the authentication of a peer is coupled with the establishment of a distance bound to her location. The protocol family includes a *measurement phase* where the response to a challenge has to be generated by the application of *xor*. It was therefore necessary to extend the original protocol theories in Isabelle with lots of additional notions to integrate the algebraic properties of xor. In particular, message terms are associated with their normal-forms that are defined based on a reduction function implemented in ML (within Isabelle) according to *xor*-equations. Furthermore, *analz* and *synth* (for message synthesis using the output of *analz*) are replaced with a (predicate) function *dm* to define the derivable messages. These and other additional notions make it difficult to verify the main security property directly in Isabelle. Instead of that, the verification in Isabelle focused on sufficient conditions of the measurement phase, which are formulated in a meta-theoretical theorem proven by hand.

1.2 Contributions and Thesis Outline

Prior to our work on this thesis, we have implemented a protocol verification framework in the VSE tool that provides specification and proof support. The first version adopted the notions of the inductive method going back to the work of Paulson. During the verification of real-world protocols like a Chip-Card-Based Biometric Identification Protocol, [31, 87], and the Protocols for Extended Access Control on Machine Readable Travel Documents, [61], we added some extensions to the original concepts and implemented more powerful proof heuristics to enhance the degree of automatization, [30].

The verification of PACE was the starting point of the work whose results are presented in this thesis since the existing techniques were completely unsuitable for both, the algebraic specification of cryptographic primitives and the property of being resistant against offline password testing. After implementing an initial version of the new approach, [33], we identified a few anomalies of the PACE protocol, [101]. In a follow up project, we verified a new version of PACE with additional checks addressing parts of the anomalies, [60, 32], although they were not security-relevant in the eID scenario, [57]. A first attempt to apply the new techniques to TC-AMP, [101], failed since the complex algebra of TC-AMP revealed significant shortcomings in the methodology elaborated so far. This was the motivation for subsequent research work resulting in the general theory presented below.

The initial plan of the thesis was to present the protocol verification framework in VSE, [30], and its application to real-world protocols, [31, 87, 32], including only the initial

version of the new approach applied to PACE, [33]. Due to significant progress towards a general theory that turned out to be appropriate for a treatment of TC-AMP we changed the structure of the thesis focusing on these recent results. Our previous work on the verification of real-world protocols in the original VSE framework is not part of this thesis, except of re-used solutions for the implementation of the new approach.

The basic ideas of our general theory were presented in [33], using PACE as running example. Results on our verification of PACE and TC-AMP were presented in technical reports, in order to be evaluated and used by the BSI as projects' partner. Many sections of the thesis are adopted from these reports and from [32]. The chapter about the implementation in VSE includes re-used results presented in former works, [31, 87, 30].

1.2.1 Major Contributions

Fig. 1.1 shows the main constituents of the overall method for inductive verification of protocols based on algebraic specifications. Contributions are marked in blue. They can be structured in three groups.

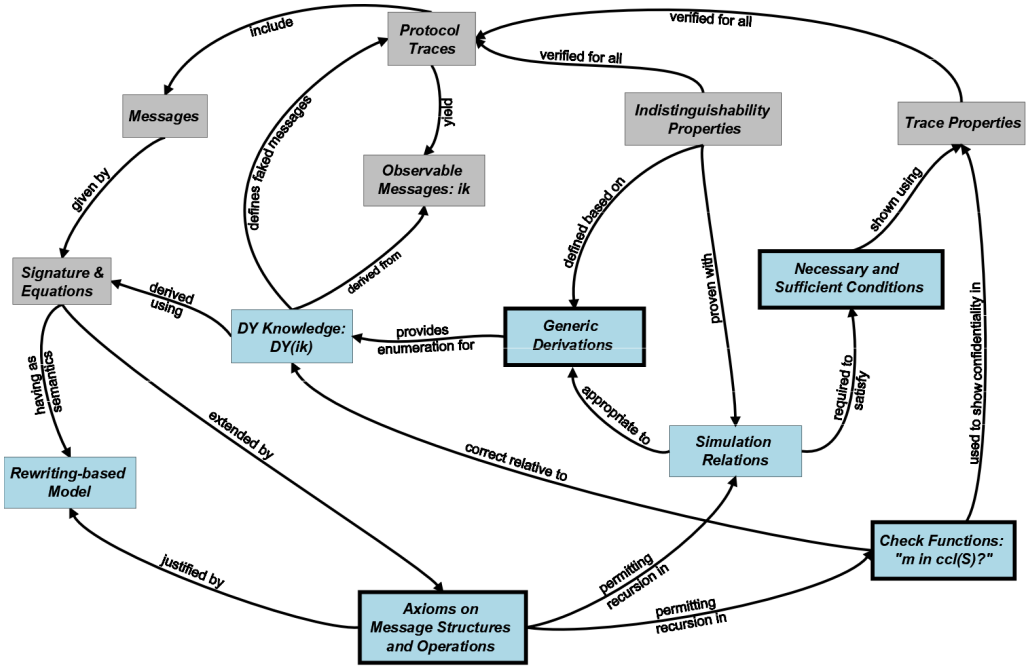

Figure 1.1: A method for the inductive verification of protocols based on algebraic specifications

(1) We present a general method to extend equational systems in a sound way by axioms that allow for reasoning about the underlying structure of initial models. In particular, this includes

- derivation of inequalities,
- complete case distinctions,
- well-founded orders,
- recursive definitions based on these, and

- inductive proofs.

The hidden algebraic structure is made explicit by a careful analysis of the execution of operations in a *rewriting based model* that is equivalent to the original non-constructive standard model of equations.

Generally, rewriting techniques are used in deduction systems as part of the inference engines. In our approach, we employ complete rewriting systems to identify the structure of objects and to describe and clarify basic operations and types of messages. In particular, we distinguish between merging operations and different kinds of extraction and composition operations. Moreover, we classify messages according to their definite top-symbols. For instance, composed messages $\oplus(a,b)$ for atomic a,b belong to the $\oplus$-messages.

(2) A simple and straightforward inductive definition of what can be deduced by the attacker from a finite set of immediate observable items *ik* is given by the so-called Dolev-Yao closure DY(ik). Such a simple and easy to analyze definition is crucial for an adequate basic setting. However, inductive proofs heavily rely on arguments about items that *cannot* be obtained from *ik* by the attacker. We present a novel approach to solve this key problem by two techniques that are basically complementary and based on recursive *check-functions*.

(A) Basic check-functions take a finite set of secrets S as a parameter. For a given (fixed) S and a given item m the test of m, $check(S,m)$, is positive (*true*), if, like in medicine, m is "critical" and should *not* be immediately observable by the attacker. Basic check-functions can be designed to be *pessimistic* in the sense that not every item classified as "critical" actually permits the derivation of some element in S. But, they have to be proved *correct* wrt. the given Dolev-Yao closure, meaning that "uncritical" items cannot be used to derive elements of S. One has to show that a negative outcome, i.e. $check(S,m) = false$ for *all* messages $m \in ik$, allows to conclude that S is actually secure, i.e. $DY(ik) \cap S = \emptyset$.

The sets S used for confidentiality proofs have to be carefully chosen according to the protocol messages. If some $s \in S$ can be *extracted* from a protocol message m, additional items required for this derivation have to be included in S to obtain $check(S,m) = false$.

This technique is not restricted to atomic secrets. For composed elements S has to include substructures required for their composition. Note that according to (1) there is such a notion of substructures.

Basic check-functions ccl_1 can be *uniformly defined* in all message algebras. However, ccl_1 is restricted to sets S where the items cannot be obtained by *merging*. The problem with merging is that there are *infinitely many* items allowing a derivation of a given secret. This problem can be solved by basic check-functions ccl_3 that extend ccl_1 with additional checks to integrate merging. In particular, we consider *only* the relevant merging derivations permitted by the protocol. The corresponding items are provided as an additional argument of ccl_3.

(B) The check-functions discussed above cover structured secrets with known structure but cannot be (directly) used to exclude that certain infinite sets of arbitrary nested messages can be *generated* by the attacker. For an infinite set of messages of same type given by a description D it suffices to show that messages s satisfying $D(s)$

1. cannot be *extracted* from elements $m \in ik$ even if we assume that all additional items necessary for the extraction are *public*, and
2. cannot be *composed* without using *substructures* s' that can be shown to be protected according to (A) or (B).

Again we use a recursive function $ccl_2(s,m)$ to check property (1). The reduction in (2) is given by a theorem that is generic for each type of messages, except for those that can be obtained by *merging*.

Since infinitely many items may be involved in a *generation* by merging, we took a different route. We use a protocol-specific invariant I to provide a *complete* case distinction on messages m derivable from ik that are of the type given by D. The conditions provided by I and the description D are then used to show that none of the derivable messages m satisfies $D(m)$. The use of protocol-specific invariants is another reduction technique, since parts of the conditions can be shown using the technique of (A).

(3) Our formalization of indistinguishability properties uses an enumeration of generic *derivation trees* representing the derivation strategies δ and δ' introduced in Sec. 1.1.2.3. This enumeration is consistent with the sub-tree relation. The reduction to trace properties is based on a *simulation relation*, given by a finite set of pairs that is then extended by composition. A generic central theorem permits to show that the simulation relation holds between the corresponding nodes of instantiated trees representing $\delta(kb), \delta(kb')$ and $\delta'(kb), \delta'(kb')$. The premises of the theorem state the necessary and sufficient conditions for the indistinguishability property to hold. The conditions include constraints on the initial set of pairs relative to the intruder knowledge derivable from kb and kb'. They can be shown by help of typical trace properties on different types of derivable messages and the protection of their substructures.

The proof of the central theorem is by induction on indices. Moreover it uses two lemmata to ensure that the simulation relation is *bijective* between (and restricted to) $DY(kb)$ and $DY(kb')$ where crucial structural properties are preserved by composition of messages.

In all three cases our approach is applicable to a wide range of algebraically specified protocols. General aspects of our contributions are strongly emphasized and discussed. Apart from that there are two fully worked out instantiations of the general theory. PACE and TC-AMP are real world protocols (to be) applied in the context of the German (electronic) identity cards. The investigation whose results are presented in this thesis started with a successful inductive verification of PACE. From that starting point it took quite some time and effort to generalize the underlying theory so that it could be applied to TC-AMP. To the best of our knowledge TC-AMP is among the most complex algebraically specified protocols that have been formally verified. In particular we do not know of any approaches to apply inductive proof techniques to comparable cases.

1.2.2 Outline of the Thesis

The thesis is described in two parts, where PACE and TC-AMP serve as concrete instantiations of the general approach. Both case studies are described in Chap. 2: An *adjusted* "Alice-and-Bob" protocol notation is used for the specification of PACE and TC-AMP and for the introduction of their message algebras (cryptographic primitives and required equations). Additionally, the protocol properties are discussed in the context of the application scenarios.

1.2.2.1 Part II: Handling Trace Properties

In this section, we elaborate on the first and second major contribution and outline the corresponding chapters.

Chap. 3–5 include the theoretical foundations and the used techniques to obtain the sound extension of equational specifications with appropriate axioms on message structures and operations:

- Chap. 3: The required notions (initial models, equational derivations, quotient algebras and complete (modular) rewriting systems) are introduced to prove that rewriting-based models of equational specifications (Σ, E) are initial. Then, *message algebra specifications* are defined by extending equational specifications with an enumerable set of atomic messages, and their models are defined using the rewriting-based models of the corresponding equational specifications. The chapter ends with the definition of *DY knowledge*, i.e. of the closure operator $DY(.)$.

- Chap. 4: The generation of the axioms about the structure of messages is preceded by the completion of the equations in E to a corresponding complete (modular) rewriting system R/A, where R includes the rewrite rules and A the non-orientable equations. Since available completion tools do not permit to deal with all non-orientable equations in PACE and TC-AMP, we describe in Chap. 4 an ad hoc completion algorithm applicable for typical non-orientable equations in algebraic protocol specifications, [42]. Here, the sets A are restricted to *permutative* equations on same function symbols f, [26], which we call *permutative functions*. This restriction not only guarantees finite A-equivalence classes, [94], but also allowed us to replace the superposition on equations in A by superposing appropriately extended rewrite rules. The completion procedure is as well modular in A, as the extensions to the rewrite rules and the decompositions in unification are A-specific. It was applied by hand for the completion of the equations in PACE and TC-AMP.

- Chap. 5: In the rewriting-based model given by a complete modulo rewriting system R/A, the objects are R-reduced A-equivalence classes. The operations are defined by the possible transformations to R/A-normal forms after the application of function symbols to arbitrary terms from R-reduced A-equivalence classes. In case A is defined on permutative functions, the R-reduced A-equivalence classes have nice properties that permit not only for a *uniform* generation of the axioms about the types of messages and their substructures, but also for a relatively simple definition of an appropriate measure function. We use predicates obj^f to specify the types of composed messages as f-objects (also f-messages), meaning that the terms in the corresponding R-reduced A-equivalence classes possess f as top-symbols. For f with arity n, the predicate obj^f has arity $n+1$ and is defined on f-objects together with their direct substructures, which permit for a derivation of these f-objects by a constructor-type application of f. We specify the predicates obj^f in axioms about the so-called *basic operations*. Besides the constructor-type case, an axiom includes all other cases where the structures of objects resulting by the application of f are specified relative to the structures of used arguments given by the possible rewrite-based transformations. The structural conditions are formalized using obj^f predicates so that the ensemble of the axioms provides a mutual recursive specification of these predicates. To justify (and illustrate) how these axioms cover exactly the basic operations induced by R and A, we use an intermediate representation of these operations in so-called operation schemata. They are similar to rewrite rules enriched with labels to distinguish top-irreducible from arbitrary occurrences of function symbols. The intermediate representation in operation schemata is also used for a *uniform generation* of the axioms, as demonstrated for PACE and TC-AMP.

As mentioned above, the structure provided by the new axioms permits for the definition of the recursive check functions used in our proof techniques to deal with the algebraic intruder reasoning in the verification of trace properties. Chap. 6–9 are mainly reserved for

the description of these proof techniques, their implementation in the VSE tool and for their application in the verification of the trace properties of PACE and TC-AMP:

- Chap. 6: We describe our technique to show required protocol verification arguments about items that *cannot* be obtained by the attacker from immediate observable messages *ik*. As introduced above in (2-A), the basic check-functions *test* that messages in *ik* are not *critical* for given finite sets S of secrets. They permit to prove confidentiality (using appropriately selected S) just by checking the messages of *ik* when they are guaranteed to be correct. This necessitates to adequately embed the effects of the basic operations in the definition of the basic check-functions and/or in conditions on the sets S as part of the correctness theorems. Therefore, we characterize seven types of operations by analyzing the (combined) effects of basic operations in PACE and TC-AMP. Furthermore, we qualify the constructor-type operations and the elementary (not composed) extraction operations to be *canonical*, as these kinds of operations can be embedded in one *uniform* basic check-function ccl_1. Not only the definition of this so-called canonical basic check-function but also the required condition on S for the correctness theorem are formulated according to the same schemata in all message algebras. We illustrate this for ccl_1 in PACE and TC-AMP.

  The check-functions (ccl_2) used together with generic reduction theorems as introduced above in (2-B) are also canonical, because they are obtained by a uniform adaptation of ccl_1. We illustrate this for ccl_2 in PACE and TC-AMP and describe the four typical kinds of the generic reduction theorems, which cover all relevant types of messages in PACE and TC-AMP.

  For $\oplus$-messages in TC-AMP, which are derivable by merging operations, i.e. using items different from their substructures, the reduction technique based on ccl_2 does not work. This permits us to exemplify how to formulate and prove a protocol-specific invariant providing a complete case distinction on the derivable $\oplus$-messages and to use this to show confidentiality. In particular, we describe how the function obtained by an adaptation of ccl_2 to express generalized occurrence of $\oplus$-messages is used in the invariant and supports its proof.

  The use of ccl_2 and protocol-specific invariants are complementary to basic check-functions, required to show the confidentiality of secrets inside protocol messages. Since the canonical basic check-function is not applicable with sets S including secrets derivable by merging, we describe a second basic check-function (ccl_3) for this kind of application scenario. For that purpose, we use an example key distribution protocol in the TC-AMP algebra. We designed the protocol so that the confidentiality of the session key is shown using sets S that include $\oplus$-messages. This way, the corresponding proof serves as a running example to describe the adaptation of ccl_1 to ccl_3 and the additional conditions in the correctness proof of ccl_3.

- Chap. 7: The implemented VSE framework for the inductive verification of cryptographic protocols based on message algebras is described in this chapter. It includes specification facilities in form of abstract data type (ADT) theories and proof construction facilities in form of proof heuristics. The ADT theories originate partly from the VSE implementation of the inductive method in the simple *enc-dec* scenario. The axioms about the underlying message structures and many auxiliary notions are in algebra-specific ADT theories, which can be adapted to new message algebras. We explain in this chapter how protocols and their trace properties are specified. Additionally, we describe the common proof scheme for protocol properties and the proof heuristics that automatize a very large part of the proof construction steps.

- Chap. 8: The specification of PACE and the verification of its trace properties in the VSE system are described.

- Chap. 9: The specification of TC-AMP and the verification of its trace properties are described. The provided verification details are quite sufficient to obtain corresponding machine-checked proofs in VSE.

1.2.2.2 Part III: Handling Indistinguishability Properties

In this section, we elaborate on the third major contribution and outline the corresponding chapters.

- Chap. 10: We motivate the formal definition of indistinguishability properties, describe how they are shown with the help of a simulation relation and we provide our axiomatization for derivation trees that induce an appropriate (level-wise) enumeration of their nodes: We first explain that different kinds of indistinguishability properties are eventually about simultaneously considered message sequences kb and kb' that the attacker may not distinguish through offline testing. Then, we motivate the restriction of testing algorithms to those built from generic derivations and equality checks. This justifies the definition of resistance against offline guessing attacks by (1.2) in Sec. 1.1.2.3 and the use of an analog definition for other indistinguishability properties as discussed in Sec. 1.1.2.4.

  For tool-supported verification (in VSE), we formalize generic derivations as derivation trees and provide an appropriate axiomatization that enumerates all the derivation tree nodes. Starting with the leaf nodes, i.e. with the items in the *first level* kb (respectively kb'), the enumeration continues successively with the subsequent levels consisting of parent nodes. The axiomatization associates the parent nodes with labels identifying (applied) function symbols and the positions of the child nodes from the previous levels whose respective results correspond to the arguments for the function applications.

- Chap. 11: We present our proof technique for indistinguishability properties using the resistance of PACE against offline password testing as a running example. The knowledge bases kb, kb' to be shown indistinguishable are defined relative to relevant protocol traces and property-specific parameters $\overline{x}$. Indistinguishability for kb, kb' holds according to Chap. 10, if appropriate simulation relations are provided for all kb, kb' fixed relative to relevant protocol traces. As described above in (3), we use finite sets of message pairs (as basis relations) and extend them uniformly by composition to simulation relations. The basis relations are required to be the smallest sets that fulfill certain inclusion rules relative to the regular messages in ik and to the property-specific parameters $\overline{x}$. These inclusion rules need to be defined carefully as part of the proof, ensuring that the basis relations are complete. Besides the so-called basis simulation relation lemma, where we show by trace induction that finite sets satisfying the inclusion rules without violating minimality exist, we apply the central indistinguishability theorem that provides sufficient and necessary conditions for the uniform extension of the considered basis relations to be an appropriate simulation relation. We describe how these conditions can be shown using regularity properties of the protocol on different types of derivable messages and the protection of their substructures.

  We identify the conditions of the central indistinguishability theorem in the PACE algebra during the description of its proof and the used lemmata: The so-called domain restriction lemma guarantees that simulation relations are subsets of $DY(kb) \times DY(kb')$ and the so-called structural mapping lemma ensures that simulation relations are bijective between $DY(kb)$ and $DY(kb')$ and respect crucial structural properties.

- Chap. 12: In this chapter, we apply the proof technique from Chap. 11 to show the resistance of PACE against offline password testing. We provide the inclusion rules for

the definition of the basis relations. We formalize the required regularity properties and explain how they are shown by trace induction. Then, we describe the proof of the basis simulation relation lemma and of the proof obligations resulting from the conditions of the central indistinguishability theorem.

- Chap. 13: We provide a similar central indistinguishability theorem as in Chap. 11 by adapting the theorem conditions to the TC-AMP algebra. We also describe its proof, using practically the same domain restriction lemma and a similar structural mapping lemma. The latter is obtained by adapting the structural properties to the new derivations by composition and in particular by the non-canonical operations.
- Chap. 14: In this chapter, we show the resistance of TC-AMP against offline password testing describing the same details as in Chap. 12. Clearly, we handle here the conditions of the central indistinguishability theorem identified in Chap. 13.

Finally, we conclude the thesis in Chap. 15 by a brief summary of the main results and related future work.

Last but not least, it should be noted that except of the meta-theory justifying the extension to equational specifications (see Chap. 3, 4 and Sec. 5.1.1), all presented theorems and verified protocol properties are (respectively can be) proven in the VSE tool and checked by its kernel inference machine. To make clear where the VSE tool is used, we label the corresponding definitions, axioms, theorems and lemmata with **VSE** if counterparts already exist in VSE. We also use the (gray) label VSE if counterparts can be generated in VSE straightforwardly by simple adaptations and/or if minor proof checking effort is required.

Chapter 2

Protocols Based on Message Algebras

Cryptographic primitives with algebraic properties beyond the simple *enc-dec* scenario are generally used in protocols that achieve *application-specific* security objectives. In this chapter we sketch this issue for our case studies, two protocols that are developed by the (German) Federal Office for Information Security (BSI) for use in the German ID card and machine readable travel documents (MRTDs): PACE is deployed since November 2010 to establish a secure connection between an inspection terminal and the RFID chip implanted in an ID card (some years later, the RFID chip implanted in a MRTD) using a *password*. It aims at the *strongest* security objectives in the field of password protocols, including the resistance against offline password testing attacks.

In parallel to the development of PACE, BSI has worked on a second protocol as fall-back solution for PACE. This work consists in adapting the TP-AMP (Three-Pass Authenticated key agreement via Memorable Passwords) protocol, [73], to the ID card application scenario. The result is a protocol called TC-AMP, for Terminal-Card AMP, [101]. TC-AMP is shorter than PACE, since it exploits algebraic properties of elliptic curve cryptography. For this reason, the TC-AMP algebra is far more involved than the PACE algebra.

After the introduction of an *adjusted* "Alice-and-Bob" protocol notation for protocols using cryptographic primitives beyond the simple *enc-dec* scenario, we describe PACE and then TC-AMP, discussing their security objectives and the impact of the message algebras.

2.1 Protocol Notation

In the enormous number of publications about (the verification of) cryptographic protocols there are a couple of similar abstract notations that make use of "Alice" and "Bob" to distinguish the *roles* of protocol participants in two-party protocols: Alice is typically the *initiator* and Bob the *responder*. For this reason, these notations are referred to in general by the "Alice-and-Bob" notation. In this section, we describe a (variation of the) "Alice-and-Bob" notation that allows us to express both the sender's view and the receiver's view on exchanged messages.

2.1.1 Steps, Roles and Participants

Generally, cryptographic protocols are composed out of n *steps* (where $n > 1$). The i-th protocol step is represented in our notation according to the following schema:

$$\text{i. } Role \longrightarrow Role' \ : \ Msg \,\%\, Msg' \,\&\, Diseqs. \tag{2.1}$$

"*Role*" and "*Role′*" are place holders for the *protocol role* of the sender and for that of the intended receiver, respectively. "*Msg*" and "*Msg′*" are place holders for *message terms* that represent the *i*-th message from the *view* of the sender and from that of the receiver, respectively; The sender's view shows how the message is computed and the receiver's view how it is *matched* (see Sec. 2.1.2). "*Diseqs*" is a place holder for inequality checks; The receiver accepts the message *only if* it matches "*Msg′*" and the inequalities given by "*Diseqs*" hold.

Obviously, "&" is simplified in steps where no inequality check is required. Furthermore, the separator "%" and the receiver's view "*Msg′*" are simplified in steps where a single message term "*Msg*" represents both the sender's view as well as the receiver's view.

The *roles* in a protocol correspond to the chronological *local* actions of each protocol participant (see Sec. 2.1.2). Note, that roles are equivalent to strands, [55], in case of protocols with a fixed number of steps and without inequality checks.

As introduced above, we distinguish in two-party protocols, e.g., in PACE and TC-AMP, two roles: The initiator Alice (abbreviated as A) is the sender in the *odd* steps and the receiver in the *even* steps; The responder Bob (abbreviated as B) is the receiver in the *odd* steps and the sender in the *even* steps.

Actually, the role names in abstract protocol specifications are place holders for *identifiers* (names, or addresses) of protocol participants who are able to run the protocol in the corresponding roles: For each role, the protocol assumes the availability of certain initial knowledge that is needed during the protocol run. The initial knowledge includes for instance different kinds of long-term keys.

Basically, the required initial knowledge is made available to protocol participants according to the organizational and technical settings of the IT application where the protocol is deployed. In this context, we distinguish for two-party protocols two alternative models:

- All participants possess the required initial knowledge to run the protocol in roles A and B. In this case, we denote participant identifiers by $ag_1, ag_2, \ldots$.
- The participants are separated in two groups: A group of participants who possess *just* the required initial knowledge to run the protocol in role A. We denote them by $ag_{A1}, ag_{A2}, \ldots$. The remaining participants possess *just* the required initial knowledge to run the protocol in role B. We denote them by $ag_{B1}, ag_{B2}, \ldots$.

2.1.2 Messages and Constituents

The message terms in our notation represent how the protocol messages are *computed* by the senders and *handled* by the receivers. They contain (protocol) variables as place holders for message parts that are determined successively in the protocol run. The variables are denoted with capital letters, to distinguish them from the function symbols (cryptographic primitives) and the constants, if any. Constants are employed generally as *keywords*, to distinguish different protocol steps with similar messages.

Before we explain how the instances for protocol variables are determined, we recall that protocols propagate certain *structures* in the exchanged messages. This includes concatenations (or pairs), which are represented in "Alice-and-Bob" notations with the help of commas between the parts.

Message structures provide means to encode purposeful information in distinct message parts, which occur typically in our protocol notation as different kinds of variables. We distinguish two groups of variables: (i) place holders for *atomic* message parts and (ii) place holders for *arbitrary* message parts that fit into receiver's views.

2.1.2.1 Atomic Message Parts

We introduce foremost those variables that are place holders for the mostly used kinds of *atomic* message parts.

- Role Names are instantiated by *participant identifiers* (see Sec. 2.1.1). They are generally used in messages to determine the communication partner(s). This could be necessary for the receiver to select the right initial knowledge in the computation of the reply message.

  The instances for role names are determined according to the following conventions:

  - Each participant is assumed to be aware of her role in the protocol: On the side of the sender (resp. receiver) the corresponding role name is instantiated by the own identifier.
  - The role names that occur first in a sender's view are generally instantiated in a *preliminary phase* of the protocol run.
  - The first occurrence of a role name in a receiver's view yields to the *binding* of this role name with the participant identifier in the received message.

- Nonce Variables are denoted by $N, N_1, N_2, \ldots$. They are place holders for random *n*umbers, which are assumed in protocol design to be used only *once*. *Nonces* are used to identify runs, in order to prevent replay attacks where old messages are re-used. We denote them by $nc_1, nc_2, \ldots$.

  The instances for nonce variables are determined according to the following conventions:

  - A variable that occurs first in a sender's view is instantiated by a *newly* generated nonce.
  - The first occurrence of a variable in a receiver's view yields to the *binding* of this variable with the corresponding nonce in the received message.

- Parameter Variables are place holders for certain kinds of parameters that belong typically to the initial knowledge. For instance, PACE and TC-AMP make use of *static generators*; Their place holders are denoted by G, G_1, and G_2 (see below).

  These variables are assumed to be instantiated in a *preliminary phase* of the protocol run. The instances are assumed to be (encoded in) *numbers* that differ from nonces. We denote them by $nb_1, nb_2, \ldots$.

Beside participant identifiers, nonces, numbers and also constants, we consider *long-term keys* to be atomic. Often, the long-term keys belong to the initial knowledge and are used in protocol runs for encryption or decryption. They are attributed to the corresponding participants with the help of certain cryptographic primitives. For instance, we could use $sk(ag)$ to denote the private keys of participants ag.

Nevertheless, there are (relatively few) protocols where long-term keys occur in the clear-text content of messages. This is the case when certain long-term keys are distributed or accessed to during the run. Typical examples are keys that are transferred as parts of *certificates* and hence handled as the public keys of given participants. Note that the attribution of such a long-term key to the corresponding participant is reached *indirectly* through the certificate (signature). In such protocols we would use Key Variables, which we denote by $K, K_1, K_2, \ldots$, and atomic keys $ky_1, ky_2, \ldots$ as their possible instances.

2.1.2.2 Arbitrary Message Parts

So far, we mentioned all instantiation conventions for the protocol variables that occur first in a sender's view. The remaining protocol variables, which occur first in a receiver's view, are denoted by $M_1, M_2, \ldots, M_{1,1}, M_{1,2}, \ldots, M_{2,1}, M_{2,2}, \ldots$. They can be bound to *arbitrary message parts*, which are determined while the receiver's view is matched to a received message:

- A receiver's view represented by a single variable is *directly* bound to any received message.
- The matching of a receiver's view represented by a concatenation starts by the binding of the not yet instantiated variables to corresponding message parts in the received message. Afterwards, the rest of the concatenation is instantiated and checked for equality with the corresponding received message parts. This equality check includes the application of the algebraic equations about the cryptographic primitives.

Note that the receiver's view can be a single message term where all variables are bound in previous steps. Here, matching corresponds simply to equality check using the algebraic equations.

Remark 1.

- Confidential nonces can be used as atomic session keys or as key materials in the computation of composed session keys.
- Our notation can be extended straightforwardly with more kinds of atomic messages, e.g., application data, misuse counters, or time stamps.

2.2 Usage of PACE/TC-AMP

Before we describe PACE (in Sec. 2.3) and TC-AMP (in Sec. 2.4) we introduce their application scenarios and their security objectives.

2.2.1 Application Scenarios

While MRTDs are generally used for border control, eIDs can be used in much more application scenarios, [18]. They can be even used for multiple online functions such as electronic contracts. Many application scenarios require access to sensitive (personal) data on the RF chip. For that purpose a password authenticated connection must be established between the chip and an inspection terminal:

1. First the password is computed by or entered in the terminal: According to the application scenario, the terminal computes the password from optical data on the MRZ or the card bearer enters a 6-digits PIN. The obtained data is handled as the (long-term) password $pwd(A)$ attributed to the chip A of the eID/MRTD.
2. A authentication protocol (PACE, or TC-AMP in case of a fall-back) is run between the terminal and the chip based on the password $pwd(A)$. If the password used by the terminal matches the password stored on the chip, the protocol run succeeds yielding a new session key.
3. The established session key is used afterwards to protect the integrity and the confidentiality of the transmitted application data from the chip to the terminal.

Running PACE (TC-AMP in case of a fall-back) achieves primarily the following guarantees:

- From the perspective of the bearer, the sensitive data on the chip can be only accessed after handing in the card/MRTD to allow for optical reading of the MRZ or after entering the right PIN.
- From the perspective of the terminal, the optically read MRZ or the entered PIN belongs indeed to the RF chip implanted in the card/MRTD.

2.2.2 Security Objectives

The above mentioned security guarantees rely on the following security objectives by the password protocol deployed in phase 2.

1. Mutual Authentication: The exchanged messages allow for A and B to check that the communication partner acts currently in the run and makes use of A's password.
2. Confidential Session Key: The exchanged messages allow for A and B to compute a confidential session key.
3. Forward Secrecy of Session Keys: A successfully guessed password does not reveal the session keys that had been generated in previous runs.
4. Resistance Against Offline Password Testing: Assuming that the attacker possesses a finite (relatively small) set of all passwords, the messages that she is able to gather cannot be exploited *offline* to identify any password used in any protocol run.

Note that the password protocol deployed in phase 2 is required to satisfy the *strongest* form of resistance, where the attacker may use (in her offline test attempts) accidentally disclosed session keys in addition to the exchanged messages. Together with the forward secrecy, this form of resistance property exhibits the highest security level of password protocols, as it was required for the German eIDs. This also explains why the password protocol deployed in eIDs is proposed for use in MRTDs instead of the Basis Access Control (BAC) Protocol, [59]. BAC does guarantee neither forward secrecy nor resistance against offline password testing.

Summing up, PACE and TC-AMP are required to establish password-authenticated session keys, for which forward secrecy is guaranteed, and to protect at the same time the used passwords from offline testing attacks. This heavily influences their design and the choice of the cryptographic primitives.

2.3 PACE

We foremost describe the protocol idea and steps, then motivate the message algebra, and we finally discuss the security objectives in this context.

2.3.1 PACE Steps

PACE aims at the establishment of a password-authenticated session key between A and B. The main idea in its design is to use the password $pwd(A)$ in the *encryption* (by *enc*) of a nonce N_1 that is necessary for the computation of the session key. In order to prevent offline testing attacks, N_1 is packed (on B's side after decryption by *dec*) together with other secrets in a *fresh* generator using practically irreversible functions *dh* and *gen*. The secrets are generated during a first DH exchange based on a static generator G. The obtained fresh generator *must* be used together with additional secrets in a second DH exchange, for the computation of the session key again by *dh*. This way, the obtained session key is bound to the password with the intermediate of N_1 used in the computation of the fresh generator.

In the following, we consider the specification of PACE in our notation from Sec. 2.1:

1. $A \longrightarrow B \ : \ enc(pwd(A), N_1) \ \% \ M_1$

2. $B \longrightarrow A \ : \ dh(G, N_2) \ \% \ M_2$

3. $A \longrightarrow B \ : \ dh(G, N_3) \ \% \ M_3$

4. $B \longrightarrow A \ : \ dh(gen(dh(G, dec(pwd(A), M_1)), dh(M_3, N_2)), N_4) \ \% \ M_4$

5. $A \longrightarrow B \ : \ dh(gen(dh(G, N_1), dh(M_2, N_3)), N_5) \ \% \ M_5$
   $\& \ M_5 \neq dh(gen(dh(G, dec(pwd(A), M_1)), dh(M_3, N_2)), N_4)$

6. $B \longrightarrow A \ : \ mac(dh(M_5, N_4), M_5)$
   $\% \ mac(dh(M_4, N_5), dh(gen(dh(G, N_1), dh(M_2, N_3)), N_5))$
   $\& \ M_4 \neq dh(gen(dh(G, N_1), dh(M_2, N_3)), N_5)$

7. $A \longrightarrow B \ : \ mac(dh(M_4, N_5), M_4)$
   $\% \ mac(dh(M_5, N_4), dh(gen(dh(G, dec(pwd(A), M_1)), dh(M_3, N_2)), N_4))$

The protocol can be divided in three phases:

- Steps 1–3 permit the exchange of the data necessary for the computation of a *fresh generator* that is bound to the password. In particular, the DH exchange in steps 2 and 3 serves to guarantee the freshness of this generator for B, who brings nonce N_2 into effect.

- Steps 4–5 constitute the second DH exchange, where the fresh generator is used. This allows A and B, who generate the nonces N_4 and N_5, respectively, to compute a common session key.

- Steps 6–7 are necessary to complete the authentication, where each participant verifies that the peer computes the same session key. This is achieved through message authentication codes (MACs) by *mac*. Note that sender and receiver compute the MACs in different ways, as given by their respective views.

Before we introduce the message algebra required to run the protocol, we comment on the inequality checks in steps 5 and 6.

Remark 2.

- A Previous version of PACE, [101], does not include the inequality checks of steps 5 and 6.

- The inequality checks allow to handle one of the anomalies that we pointed out earlier in a formal analysis of the first version of PACE. Without inequalities, there can be successful runs without the participation of a card. The terminal B would then detect this situation first during the transmission of the application data (see Sec. 2.2.1). In checking the inequality the terminal detects such a problem in step 5.

- We claim that the inequality check by A (above in step 6) is redundant, since A generates the public DH value (in step 5) after accepting M_4 and the equality of both values is very unlikely. Furthermore, such a very improbable situation would be detected due to the inequality check by B. For this reason, we will simplify the inequality check by A in the rest of the thesis.

2.3.2 PACE Algebra

For a successful run, both A and B need to compute the same fresh generator by the one-way function gen. Since gen is injective, the decryption of $M_1 = enc(pwd(A), N_1)$ by B in step 4 must result in N_1, i.e. $dec(pwd(A), enc(pwd(A), N_1)) = N_1$. Additionally, the DH values $dh(M_3, N_2)$ and $dh(M_2, N_3)$, respectively, used as the second argument of gen by B in step 4 and by A in step 5, respectively, must be equal. This means for $M_2 = dh(G, N_2)$ and $M_3 = dh(G, N_3)$, we have $dh(dh(G, N_3), N_2) = dh(dh(G, N_2), N_3)$.

Consequently, the following equations on the cryptographic primitives dec, enc and dh are necessary to run PACE:

$$\begin{aligned} \mathbf{e_1}: &\quad dec(x, enc(x, y)) = x \\ \mathbf{e_2}: &\quad dh(dh(x, y), z) = dh(dh(x, z), y) \end{aligned}$$

Equation $\mathbf{e_1}$ is the standard property for *symmetric* encryption, where each key equals its inverse.

Property $\mathbf{e_2}$ is the essential equation for a DH exchange, since it allows each participant to compute the common DH value using the public DH value received from the peer and the own private DH value. This happens twice in PACE: in the first DH exchange of steps 2 and 3 to compute the second argument of the fresh generator and then in the second DH exchange of steps 4 and 5 to compute the common DH value used (in steps 6 and 7) as MAC key.

In addition to the primitives described in Sec. 2.3.1, i.e. enc, dec, dh, gen, and mac, three more primitives will be used in the formal specification:

- $pair(.,.)$ for the construction of message pairs that are used to represent initial knowledge,
- and $fst(.)$, respectively $snd(.)$, to select the first, respectively second pair component.

Except of fst and snd which are of arity 1, all other operators are of arity 2.

The operational meaning of fst and snd is given by the following equations:

$$\begin{aligned} \mathbf{e_3}: &\quad fst(pair(x, y)) = x \\ \mathbf{e_4}: &\quad snd(pair(x, y)) = y \end{aligned}$$

Furthermore, the following equation is necessary for resistance against offline password testing.

$$\mathbf{e_5}: \quad enc(x, dec(x, y)) = y$$

The presence of this equation with the decryption equation $\mathbf{e_1}$ prevents offline tests by decrypting the first message and then encrypting the result. $\mathbf{e_5}$ guarantees that the outcome of the tests is (independent of the used password) in all cases identical with the used first message.

2.3.3 PACE Security

The security of PACE is based both on the *binding* of the fresh generator $\widehat{g}$ to the password of A and on the use of *local* secrets. The way how $\widehat{g}$ is generated necessitates to employ the password of A and to participate in the first DH exchange (in steps 2 and 3). The protection of this password guarantees the confidentiality of $\widehat{g}$ and the use of a common DH value relative to a static generator g guarantees that $\widehat{g}$ is cryptographically as strong as g.

The confidentiality of $\widehat{g}$ is essential for the authentication guarantee. For the sender of a DH value $dh(\widehat{g}, nc)$, the receiving of a message composed with the help of a MAC-key $dh(M, nc)$ after having received M implies that M has been generated using the same

generator $\widehat{g}$. This is necessary, since the local secret nc cannot be extracted from $dh(\widehat{g},nc)$. Its occurrence in $dh(M,nc)$ necessitates to generate M by $dh(\widehat{g},nc')$ using $\widehat{g}$ with a second local secret nc', and then to compute $dh(M,nc)$ by the application of dh to $dh(\widehat{g},nc)$ and nc'. That is, the peer must be also in possession of $\widehat{g}$ and thus of the password of A, which is necessary for the computation of $\widehat{g}$.

The necessity to employ one of the local secrets nc or nc' in the computation of the MAC-key explains the forward secrecy of the session key, which is generated using the same confidential key material. The disclosure of the password of A does not allow to generate $dh(M,nc) = dh(dh(\widehat{g},nc'),nc)$, as long as nc and nc' remain confidential.

The resistance against offline password testing relies on a similar argument. The local secrecy of nc and nc' prevents the attacker to verify any password candidate through the decryption of the first message, the use of the result to re-compute a fresh generator, and finally the use of this generator to compute a DH value for comparison with $dh(\widehat{g},nc)$ or $dh(\widehat{g},nc')$.

2.4 TC-AMP

We first describe the protocol idea and steps, then motivate the message algebra, and we finally discuss the security objectives in this context.

2.4.1 TC-AMP Steps

TC-AMP aims at the establishment of a password-authenticated session key between A and B. It is carried out in three steps according to the challenge and response principle. The basic idea in its design is to compose each challenge using the password $pwd(A)$ together with local secrets N_1 and N_2. The protection of the local secrets is crucial to prevent offline testing attacks.

In the following, we consider the specification of TC-AMP in our notation from Sec. 2.1:

1. $B \longrightarrow A \;:\; \oplus(*(N_1,G_1),\ominus(*(pwd(A),G_2)))\ \%\ M_1$

2. $A \longrightarrow B \;:\; \langle *(N_2,*(pwd(A),G_1)),$
$h_1(M_1,\ *(N_2,*(pwd(A),G_1)),$
$*\,(N_2,\oplus(\oplus(M_1,*(pwd(A),G_2)),*(M_1,G_1)))))\rangle\ \%$
$\langle M_{2,1},$
$h_1(\oplus(*(N_1,G_1),\ominus(*(pwd(A),G_2))),\ M_{2,1},$
$\oplus\,(*(inv(pwd(A)),*(N_1,M_{2,1})),$
$*\,(inv(pwd(A)),*(\oplus(*(N_1,G_1),\ominus(*(pwd(A),G_2))),M_{2,1})))))\rangle$

3. $B \longrightarrow A \;:\; h_2(\oplus(*(N_1,G_1),\ominus(*(pwd(A),G_2))),\ M_{2,1},$
$\oplus\,(*(inv(pwd(A)),*(N_1,M_{2,1})),$
$*\,(inv(pwd(A)),*(\oplus(*(N_1,G_1),\ominus(*(pwd(A),G_2))),M_{2,1}))))\ \%$
$h_2(M_1,\ *(N_2,*(pwd(A),G_1)),$
$*\,(N_2,\oplus(\oplus(M_1,*(pwd(A),G_2)),*(M_1,G_1))))$

For technical reasons, the terminal B initiates the run. In terms of elliptic curve cryptography, the challenge of the terminal $*(N_1,G_1)$ is generated by the multiplication of a scalar (a nonce N_1) with a base point G_1. The binding to the password of A is reached through composition by ($\oplus$ for) point addition with $\ominus(*(pwd(A),G_2))$, where the password $pwd(A)$ is multiplied with a second base point G_2.

In step 2, the message of the card A includes its challenge $*(N_2,G_1)$ and its response to the challenge of B. The challenge of A is generated similarly by the multiplication of a new scalar (nonce N_2) with the same base point G_1. Contrarily to the first step, the binding to the password in the second step is reached directly through scalar multiplication. The

response to the challenge of B in the second message part cannot be generated without the extraction of $*(N_1,G_1)$ from the first message. That extraction necessitates to employ $*(pwd(A),G_2)$ that cannot be generated without possessing the password $pwd(A)$. This allows B to verify that the response originates from the owner of this password, i.e. from A.

The third message serves as the response by B to the challenge of A. It allows A to verify that the peer has extracted $*(N_2,G_1)$ using $inv(pwd(A))$, which requires to possess $pwd(A)$.

Finally, we notice that G_2 is assumed to be linearly independent from G_1. Otherwise, the resistance against offline testing attacks is violated. The requirement is added to counter such an attack in a prior version of TC-AMP, where the equality of G_2 and G_1 is exploited.

2.4.2 TC-AMP Algebra

For a successful run, both A and B need to compute the same hash values by h_1 and h_2 in steps 2 and 3, respectively. In particular, they need to use a same third argument. Provided M_1 is, as expected by A of the form $\oplus(*(N_1,G_1),\ominus(*(pwd(A),G_2)))$, the computation of this argument by A includes the following algebraic transformations:

1. $*(N_2,\oplus(\oplus(\oplus(*(N_1,G_1),\ominus(*(pwd(A),G_2))),*(pwd(A),G_2)),$
   $*(\oplus(*(N_1,G_1),\ominus(*(pwd(A),G_2))),G_1)))$ equals
   $*(N_2,\oplus(\oplus(*(N_1,G_1),*(\oplus(*(N_1,G_1),\ominus(*(pwd(A),G_2))),G_1)),$
   $\oplus(\ominus(*(pwd(A),G_2)),*(pwd(A),G_2))))$,
2. which can be simplified to
   $*(N_2,\oplus(*(N_1,G_1),*(\oplus(*(N_1,G_1),\ominus(*(pwd(A),G_2))),G_1)))$.

These transformations necessitate to apply the equations of an abelian group, where we use ∞ as identity element:

$$\begin{aligned}
\mathbf{e_6}: &\quad \oplus(x,y) = \oplus(y,x)\\
\mathbf{e_7}: &\quad \oplus(x,\oplus(y,z)) = \oplus(\oplus(x,y),z)\\
\mathbf{e_8}: &\quad \oplus(x,\ominus(x)) = \infty\\
\mathbf{e_9}: &\quad \oplus(x,\infty) = x
\end{aligned}$$

From the point of view of the terminal B the message part $M_{2,1}$ is expected to be of the form $*(N_2,*(pwd(A),G_1))$ and the computation of the third argument used to obtain the hash values includes the following algebraic transformations:

1. $\oplus(*(inv(pwd(A)),*(N_1,*(N_2,*(pwd(A),G_1)))),$
   $*(inv(pwd(A)),*(\oplus(*(N_1,G_1),\ominus(*(pwd(A),G_2))),*(N_2,*(pwd(A),G_1)))))$
   equals
   $\oplus(*(pwd(A),*(inv(pwd(A)),*(N_1,*(N_2,G_1)))),$
   $*(pwd(A),*(inv(pwd(A)),*(\oplus(*(N_1,G_1),\ominus(*(pwd(A),G_2))),*(N_2,G_1)))))$,
2. which can be simplified to
   $\oplus(*(N_1,*(N_2,G_1)),*(\oplus(*(N_1,G_1),\ominus(*(pwd(A),G_2))),*(N_2,G_1)))$.

These transformations necessitate to use the following equations:

$$\begin{aligned}
\mathbf{e_{10}}: &\quad *(x,*(y,z)) = *(y,*(x,z))\\
\mathbf{e_{11}}: &\quad *(x,*(inv(x),y)) = y\\
\mathbf{e_{12}}: &\quad inv(inv(x)) = x
\end{aligned}$$

For the results of A and B (given above in 2) to be equal we need additionally the following distributivity equation:

$$\mathbf{e_{13}}: \quad *(x,\oplus(y,z)) = \oplus(*(x,y),*(x,z))$$

2.4.3 TC-AMP Security

The security of TC-AMP is based both on the *binding* of the challenges $*(nc_1,g_1)$ (for A) and $*(nc_2,g_1)$ (for B) to the password π_A of A and on the use of *local* secrets nc_1 and nc_2. The way how the responses to these challenges are generated necessitates to employ the password π_A and the protection of this password guarantees their confidentiality.

For the sender of the challenge $*(nc_1,g_1)$ as part of the first message, the receiving of a response computed with the help of $*(nc_1,M_2)$ using the attached message part M_2 implies that the same base point g_1 has been employed to generate M_2. This is necessary, since the local secret nc_1 cannot be extracted from $*(nc_1,g_1)$. Its occurrence in $*(nc_1,M_2)$ necessitates to generate M_2 by $*$ using g_1 with a second local secret nc_2, and then to compute $*(nc_1,M_2)$ also by $*$ using $*(nc_1,g_1)$ and nc_2. This must be preceded by the extraction of $*(nc_1,g_1)$ from the first message $\oplus(*(nc_1,g_1),\ominus(*(\pi_A,g_2)))$. That is, the peer must be also in possession of the password π_A, which is necessary to compute $*(\pi_A,g_2)$ required for the extraction.

The authentication guarantee for the sender of the challenge $*(nc_2,g_1)$ is similar. This message part is sent in $*(\pi_A,*(nc_2,g_1))$ attached with the response to the challenge of B in step 2. From the point of view of A, the occurrence of the local secret nc_2 in the received response implies that the peer possesses the password π_A. It is necessary to compute its inverse required for the extraction of $*(nc_2,g_1)$ from $*(\pi_A,*(nc_2,g_1))$.

The necessity to employ the local secrets nc_1 and nc_2 in the computation of the (same) third argument employed to obtain the hash values in steps 2 and 3 explains the forward secrecy of the session key. The third argument of h_1 and h_2 constitutes the confidential key material required for its computation. The disclosure of the password of A would allow to derive $\oplus(*(inv(\pi_A),*(nc_1,g_1)),\ominus(g_2))$, $*(nc_1,g_1)$ and $*(nc_2,g_1)$. But, these do not allow to compute $*(nc_2,\oplus(*(nc_1,g_1),*(M_1,g_1)))$, as long as nc_1 and nc_2 remain confidential.

The resistance against offline password testing relies on a similar argument. The local secrecy of the nonces nc_1 and nc_2 prevents the attacker to verify any password candidate through derivations by $*$ using the first message $\oplus(*(nc_1,g_1),\ominus(*(\pi_A,g_2)))$, the message part $*(\pi_A,*(nc_2,g_1))$ in step 2 and the accidentally disclosed key material $*(nc_2,\oplus(*(nc_1,g_1),*(M_1,g_1)))$. Any test using one of the obtained results necessitates to employ nc_1 or nc_2.

So far, we discussed informally how potential attacks are thwarted in the design of PACE and TC-AMP. In particular, our discussion was guided implicitly through certain attacks, only. A thorough verification necessitates the exclusion of all possible attacks and this requires a formal attacker model that incorporates the computation capabilities given by the message algebra. This will be described in the rest of the thesis.

Part II

Handling Trace Properties

Chapter 3

Equational Reasoning and Algebraic Intruder Models

Nowadays, DY-models are represented as algebraic theories, where function symbols are used for the cryptographic primitives and the equations for their properties. This is in particular important when dealing with indistinguishability properties, as it is noticed by Santiago, Escobar, Meadows and Meseguer, [88]:

> When a Dolev-Yao tool is used to check for subtle properties such as indistinguishability, it is important that it offers as detailed a picture of the properties of the cryptographic operations as possible. This is done by including information about their algebraic properties, that is, the equations obeyed by the function symbols.

After a general introduction of algebraic specifications and their semantics of interest (in Sec. 3.1 and 3.2), we define message algebras and the corresponding notion of DY intruder knowledge (in Sec. 3.3).

3.1 Equational Theories

We start with the definition of equational specifications.

Definition 1 (Signature, Terms, Equational Specification)**.**

1. A *signature* Σ is a set of function symbols associated with their arities from the set $\{0,\ldots,\max_\Sigma\}$ of natural numbers. $\Sigma\langle n\rangle$ denotes the set of function symbols that are associated with arity n.

2. For a signature Σ and a set V of variables, the set of *terms* $Ter(\Sigma,V)$ is the smallest set satisfying

   (a) $\Sigma\langle 0\rangle, V \subseteq Ter(\Sigma,V)$ and

   (b) $\{f(t_0,\ldots,t_{n-1}) \mid t_0,\ldots,t_{n-1} \in Ter(\Sigma,V)\} \subseteq Ter(\Sigma,V)$, where $f \in \Sigma\langle n\rangle$ and $n > 0$.

   For $s,t \in Ter(\Sigma,V)$, we say s is a *sub-term* of t iff s equals t or t equals $f(t_0,\ldots,t_{n-1})$ and s is a *sub-term* of t_i for $0 \leq i < n$.

   Sub-term positions are sequences of natural numbers, where ϵ denotes the empty sequence for the top (sub-term) position and $i \cdot p$ denotes a nested (sub-term) position. The sub-term of t at position p, denoted by $t|p$, is defined by

   (a) $t|p$ equals t, if p equals ϵ, and
   (b) $t|p$ equals $t_i|q$, if t equals $f(t_0,\ldots,t_{n-1})$ for $f \in \Sigma\langle n\rangle$ with $n > 0$ and if p equals $i \cdot q$ for $0 \leq i < n$.

   For a term t with a sub-term position p, we use $t[s]_p$ to denote the term that results by the replacement of the sub-term $t|p$ with a term s.

   A *(term) substitution* σ is a function $V \to Ter(\Sigma, V)$ and $t\sigma$ denotes the term that results by replacing every variable v in t with $\sigma(v)$.

   We use $Ter_0(\Sigma)$ for the set of *ground terms*, i.e. the terms in $Ter(\Sigma, V)$ that have no sub-terms in V.

3. An *equational specification* is a pair (Σ, E), where
   (a) Σ is a *signature*, and
   (b) E is a finite set of *equations* $t = s$ where $t, s \in Ter(\Sigma, V)$ for a set V of variables.

Next, we define corresponding semantics.

Definition 2 (Interpretation of Equational Specification)**.**
Let (Σ, E) be an equational specification with a given signature Σ and a given set E of equations $t = s$ where $t, s \in Ter(\Sigma, V)$ for a given set V of variables.

1. A Σ-algebra $\mathcal{A}$ is a non-empty set A together with functions $f^{\mathcal{A}} : A^n \to A$ for every function symbol $f \in \Sigma\langle n\rangle$ with $0 \leq n \leq \max_\Sigma$. A is the carrier set and the functions $f^{\mathcal{A}}$ are the interpretation of the symbols f.

2. An *assignment* for the variables in V to a (Σ-algebra $\mathcal{A}$ with) carrier set A is a function $\theta : V \to A$. It determines a meaning in $\mathcal{A}$ for every term $t \in Ter(\Sigma, V)$ by:

$$\begin{array}{rcl} [\![v]\!]^{\mathcal{A},\theta} & = & \theta(v), \text{ for } v \in V, \text{ and} \\ [\![f(t_0,\ldots,t_{n-1})]\!]^{\mathcal{A},\theta} & = & f^{\mathcal{A}}([\![t_0]\!]^{\mathcal{A},\theta},\ldots,[\![t_{n-1}]\!]^{\mathcal{A},\theta}), \text{ for } f \in \Sigma\langle n\rangle. \end{array}$$

3. A Σ-algebra $\mathcal{A} = (A, \{f^{\mathcal{A}} : A^n \to A\})$ *satisfies* an equation $s = t$ for $s, t \in Ter(\Sigma, V)$, if for every assignment $\theta : V \to A$ we have $[\![s]\!]^{\mathcal{A},\theta} = [\![t]\!]^{\mathcal{A},\theta}$. This is denoted by $\mathcal{A} \models s = t$. If $\mathcal{A}$ satisfies every equation in E we call $\mathcal{A}$ *a model of* E and denote this by $\mathcal{A} \models E$.

We want to consider the initial models, where the set of satisfied equations is minimal.

Definition 3 (Initial Models)**.**

1. Let $\mathcal{A} = (A, \{f^{\mathcal{A}} \mid f \in \Sigma\})$ and $\mathcal{B} = (B, \{f^{\mathcal{B}} \mid f \in \Sigma\})$ be Σ-algebras. $h : A \to B$ is a Σ-*homomorphism from* $\mathcal{A}$ *to* $\mathcal{B}$ if for each $f \in \Sigma\langle n\rangle$, with $0 \leq n \leq \max_\Sigma$, and all $a_0,\ldots,a_{n-1} \in A$ we have

$$h(f^{\mathcal{A}}(a_0,\ldots,a_{n-1})) = f^{\mathcal{B}}(h(a_0),\ldots,h(a_{n-1})).$$

   If h is bijective, we say $\mathcal{A}$ and $\mathcal{B}$ are *isomorphic* and call h a Σ-*isomorphism*.

2. Let $\mathcal{A} \models E$ for some set E of equations. $\mathcal{A}$ is said to be *initial in* (the class of all models of) E if for every model $\mathcal{B}$ of E, there exists a *unique* Σ-homomorphism from $\mathcal{A}$ to $\mathcal{B}$.

Initial models have in common the feature that they satisfy *exactly* those equations that are derivable from E.

Definition 4 (Equational Derivation)**.**
Let (Σ, E) be an equational specification and let $t, s \in Ter(\Sigma, V)$.

1. *t rewrites according to the term pair (l,r) at the sub-term position p to s*, which we denote by $t \rightarrow^{p}_{(l,r)} s$, if
   - (i) l matches the sub-term of t at position p, i.e. there is a substitution σ such that $t|p = l\sigma$, and
   - (ii) $s = t[r\sigma]_p$, i.e. s results from t by the replacement of $t|p$ with $r\sigma$.
2. For equation sets E, we define $\overrightarrow{E} := \{(l,r) \mid l = r \in E\}$ and $\overleftarrow{E} := \{(r,l) \mid l = r \in E\}$.
3. We define the binary relation $=_E$ on terms by: $t =_E s$ *iff*
   - *either* $t \equiv s$, i.e. t and s are syntactically equal,
   - *or* we have $t_1 \rightarrow^{p_1}_{(l_1,r_1)} t_2,\ \ldots,\ t_n \rightarrow^{p_n}_{(l_n,r_n)} s$ with
     - $t_1 \equiv t$
     - p_i is a sub-term position of the term t_i and $(l_i, r_i) \in \overrightarrow{E} \cup \overleftarrow{E}$, for $1 \leq i \leq n$.

   We say the equation $t = s$ is *derivable from* E, if $t =_E s$ holds.

$=_E$ is an equivalence relation that is a congruence.

Definition 5 (Congruence; Quotient Algebra)**.** Let Σ be a signature and $\mathcal{A}$ be a Σ-algebra. An equivalence relation $\sim$ on the carrier set A is a Σ-congruence, if for each $f \in \Sigma\langle n \rangle$ with $0 \leq n \leq \max_\Sigma$ and any $a_0, \ldots, a_{n-1}, a'_0, \ldots, a'_{n-1} \in A$ we have

$$a_0 \sim a'_0,\ \ldots,\ a_{n-1} \sim a'_{n-1} \ \Rightarrow\ f^{\mathcal{A}}(a_0, \ldots, a_{n-1}) \sim f^{\mathcal{A}}(a'_0, \ldots, a'_{n-1})\ .$$

The corresponding *quotient algebra* $\mathcal{A}/_\sim$ is then given by

- (i) the carrier set $A/_\sim$, consisting of the equivalence classes $[a_i]_\sim$, and
- (ii) by functions $f^{\mathcal{A}/_\sim}$ where $f^{\mathcal{A}/_\sim}([a_0]_\sim, \ldots, [a_{n-1}]_\sim) = [f^{\mathcal{A}}(a_0, \ldots, a_{n-1})]_\sim$

The congruence $=_E$ provides us with a partition of $Ter_0(\Sigma)$ into subsets of semantically equivalent terms.

Theorem 6. *Let (Σ, E) be an equational specification.*

1. *Then $=_E$ is an equivalence relation on the set $Ter_0(\Sigma)$ and is a Σ-congruence.*
2. *The algebra $\mathcal{I} = (Ter_0(\Sigma)/_{=_E}, \{f^{\mathcal{I}} \mid f \in \Sigma\})$ where*

$$f^{\mathcal{I}}([t_0]_{=_E}, \ldots, [t_{n-1}]_{=_E}) = [f(t_0, \ldots, t_{n-1})]_{=_E}$$

   is an initial model of E.

Proof: First, we prove that $=_E$ is an equivalence relation on $Ter_0(\Sigma)$ by showing straightforwardly that $=_E$ is reflexive, symmetric and transitive.

We know that $Ter_0(\Sigma)$ is the carrier set of the ground term algebra $\mathcal{T} = (Ter_0(\Sigma), \Sigma)$, which is clearly a Σ-algebra. The proof that $=_E$ is a Σ-congruence consists in reusing the equational derivations for $t_0 =_E s_0, \ldots, t_{n-1} =_E s_{n-1}$ to obtain an equational derivation for $f(t_0, \ldots, t_{n-1}) =_E f(s_0, \ldots, s_{n-1})$.

Let $\mathcal{B} = (B, \{f^{\mathcal{B}} \mid f \in \Sigma\})$ be an arbitrary model of E and let $h : Ter_0(\Sigma)/_{=_E} \to B$ defined by $h([t]_{=_E}) = [\![t]\!]^{\mathcal{B}}$. We first prove that h is a Σ-homomorphism from $\mathcal{I}$ to $\mathcal{B}$:

Applying the definition of $f^{\mathcal{I}}$ and then that of h, we obtain

$$\begin{aligned} h(f^{\mathcal{I}}([t_0]_{=_E}, \ldots, [t_{n-1}]_{=_E})) \quad &= h([f(t_0, \ldots, t_{n-1})]_{=_E}) \\ &= [\![f(t_0, \ldots, t_{n-1})]\!]^{\mathcal{B}}. \end{aligned}$$

Applying the definition of h (n-times), we get

$$f^{\mathcal{B}}(h([t_0]_{=_E}), \ldots, h([t_{n-1}]_{=_E})) \quad = f^{\mathcal{B}}([\![t_0]\!]^{\mathcal{B}}, \ldots, [\![t_{n-1}]\!]^{\mathcal{B}}).$$

As $\mathcal{B}$ is a model of E, we have $[\![f(t_0, \ldots, t_{n-1})]\!]^{\mathcal{B}} = f^{\mathcal{B}}([\![t_0]\!]^{\mathcal{B}}, \ldots, [\![t_{n-1}]\!]^{\mathcal{B}})$ and thus h is a Σ-homomorphism from $\mathcal{I}$ to $\mathcal{B}$.

It remains to prove that h is the unique Σ-homomorphism from $\mathcal{I}$ to $\mathcal{B}$: Assume that h' is another Σ-homomorphism from $\mathcal{I}$ to $\mathcal{B}$. Then there must be some $t \in Ter_0(\Sigma)$ such that $h([t]_{=_E}) \neq h'([t]_{=_E})$. Let us consider such a term t so that t does not have any proper sub-term s where $h([s]_{=_E}) \neq h'([s]_{=_E})$.

If $t = f$ and $f \in \Sigma\langle 0 \rangle$, then we have $h([t]_{=_E}) = h([f]_{=_E}) = [\![f]\!]^{\mathcal{B}}$. Since h' is a Σ-homomorphism, we get $h'([t]_{=_E}) = h'(f^{\mathcal{I}}()) = f^{\mathcal{B}}() = [\![f]\!]^{\mathcal{B}}$. Thus, we obtain a contradiction with $h([t]_{=_E}) \neq h'([t]_{=_E})$.

When $t = f(t_0, \ldots, t_{n-1})$ for some $f \in \Sigma\langle n \rangle$ and $n > 0$, then $h([t_i]_{=_E})$ and $h'([t_i]_{=_E})$ must be equal for all $i \in \{0, \ldots, n-1\}$. As h' is a Σ-homomorphism, we obtain

$$\begin{aligned} h'([t]_{=_E}) \quad &= h'([f(t_0, \ldots, t_{n-1})]_{=_E}) \\ &= h'(f^{\mathcal{I}}([t_0]_{=_E}, \ldots, [t_{n-1}]_{=_E})) \\ &= f^{\mathcal{B}}(h'([t_0]_{=_E}), \ldots, h'([t_{n-1}]_{=_E})) \\ &= f^{\mathcal{B}}(h([t_0]_{=_E}), \ldots, h([t_{n-1}]_{=_E})) \\ &= f^{\mathcal{B}}([\![t_0]\!]^{\mathcal{B}}, \ldots, [\![t_{n-1}]\!]^{\mathcal{B}}) \\ &= [\![f(t_0, \ldots, t_{n-1})]\!]^{\mathcal{B}} \\ &= h([f(t_0, \ldots, t_{n-1})]_{=_E}) \\ &= h([t]_{=_E}). \end{aligned}$$

Thus, we got again a contradiction with $h([t]_{=_E}) \neq h'([t]_{=_E})$. □

In the rest of the thesis we will use $[t]_E$ and often $[t]$ instead of $[t]_{=_E}$.

For our inductive approach we need recursive definitions of auxiliary functions and predicates. These have to be based on semantic concepts like "size" and "substructure". The initial model $\mathcal{I}$ above gives no hint on how these may be obtained. Nevertheless in more simple cases as the PACE algebra (see Sec. 2.3.2), the necessary structure can be "guessed" simply from the equations. A solution for the more complex TC-AMP algebra became only possible[1] by a systematic analysis of an isomorphic semantics that is based on rewriting.

[1] after quite some time

In contrast to the approach of Bergstra and Tucker, [19], where the focus is on computability results, the rewriting-based algebra is used in our approach to analyze (reduced) objects and operations on these objects and thus acquire the required structure. Based on the semantic notions introduced object level axioms are then derived (see Chap. 5).

3.2 Rewriting-based Models

We introduce first the basic notions of (term) rewriting.

Definition 7 (Term Rewriting System)**.**

1. Let $\mathcal{B}$ be a binary relation on some set S and let $\mathcal{B}^*$ be the corresponding transitive-reflexive closure.
   (a) Some $x \in S$ is *$\mathcal{B}$-reducible* (*a $\mathcal{B}$-redex*), if $(x,y) \in \mathcal{B}$ for some $y \in S$. We say x is a *$\mathcal{B}$-normalform*, if it is not reducible. We call *y a $\mathcal{B}$-normalform of x*, if $(x,y) \in \mathcal{B}^*$ and y is a normalform. A *unique* $\mathcal{B}$-normalform of x is called *the* $\mathcal{B}$-normalform of x.
   (b) The elements x and y are *$\mathcal{B}$-joinable*, if there is some z such that (x,z), $(y,z) \in \mathcal{B}^*$.
   (c) $\mathcal{B}$ is *local-confluent*, if for all $x,y_1,y_2 \in S$ with (x,y_1), $(x,y_2) \in \mathcal{B}$ we have y_1 and y_2 are $\mathcal{B}$-joinable.
   (d) $\mathcal{B}$ is *terminating*, if for all $x \in S$ there is no infinite sequence (x,y_1), $(y_1,y_2),\ldots$ of pairs in $\mathcal{B}$.
   (e) $\mathcal{B}$ is *complete*, if $\mathcal{B}$ is terminating and local-confluent.
2. A *rewrite rule* $l \to r$ consists of a non-variable term l and a second term r whose variables occur in l.
3. We call a set R of rewrite rules a *term rewriting system* (*TRS*). R induces a binary relation on terms, denoted by $t \to_R s$; We have $t \to_R s$ iff $t \to^p_{(l,r)} s$ for a position p of a sub-term in t and for some $l \to r \in R$.
   We say, R is local-confluent (terminating/complete) if $\to_R$ is local-confluent (terminating/complete).
4. For a TRS R we define $R^= := \{l = r \mid l \to r \in R\}$.

A complete TRS R allows us to obtain for each term s a *unique* t such that

- $s \to^*_R t$ and
- t is not R-reducible.

t is the R-normalform of s, which we denote by $s\downarrow_R$.

If a TRS R results by the completion of a given set E of equations, then we may assume that the completion algorithm guarantees that the binary relations $=_E$ and $=_{R^=}$ are identical. For an equational specification (Σ, E) the initial model $\mathcal{I}$ of E in Theorem 6 corresponds *then* to $(Ter_0(\Sigma)/_{=_{R^=}}, \{f^{\mathcal{I}} \mid f \in \Sigma\})$ and we have

$$f^{\mathcal{I}}([t_0]_{=_{R^=}},\ldots,[t_{n-1}]_{=_{R^=}}) = [f(t_0,\ldots,t_{n-1})]_{=_{R^=}}.$$

That is, any rewriting-based algebra that is isomorphic to the initial model of $R^=$ is clearly an initial model of E, too.

Theorem 8. *Let R be a complete TRS and $(\Sigma, R^=)$ be an equational specification. Then the algebra $\mathcal{R} = (\{t\downarrow_R \mid t \in Ter_0(\Sigma)\}, \{f^{\mathcal{R}} \mid f \in \Sigma\})$ where*

$$f^{\mathcal{R}}(t_0\downarrow_R, \ldots, t_{n-1}\downarrow_R) = f(t_0, \ldots, t_{n-1})\downarrow_R$$

is isomorphic to the initial model $\mathcal{I}$ given in Theorem 6.

Proof: We use the function $h : Ter_0(\Sigma)/_{=_{R^=}} \to \{t\downarrow_R \mid t \in Ter_0(\Sigma)\}$ defined by $h([t]_{R^=}) := t\downarrow_R$. It is easy to show that h is a Σ-homomorphism from $\mathcal{I}$ to $\mathcal{R}$ and that h is bijective. □

Generally, certain equations, such as the commutativity of $\oplus$ in TC-AMP, are not orientable without signature extension. This means, the introduction of new function symbols and the definition of operations, such as sorting of lists. All these would change message algebras by concepts at an implementation level. Instead of rewriting relations as defined above, we use modular rewriting, [64, 84, 71]. Rewriting will be modulo equivalence classes induced by the non-orientable equations.

Definition 9 (Modular Term Rewriting System)**.**

1. Let R be a TRS and let A be a set of equations. For $s, t \in Ter(\Sigma, V)$ we define $s \to_{R/A} t$ iff for some s' and t' such that $s =_A s'$ and $t =_A t'$ we have $s' \to_R t'$. We call R/A a *modular* TRS.
2. For $[s], [t] \in Ter(\Sigma, V)/_{=_A}$ we define $[s] \to_{R_A} [t]$ iff for some $s' \in [s]$ and $t' \in [t]$ we have $s' \to_R t'$. Clearly, $\to_{R_A}$ is a binary relation on A equivalence classes that determines a binary relation on terms corresponding to $\to_{R/A}$ and vice versa.
3. We say that R/A is a local-confluent/terminating/complete *modular* TRS iff $\to_{R_A}$ is local-confluent/terminating/complete.

A complete modular TRS R/A does not guarantee the existence of unique R/A-normalforms of terms s. But, it allows us to obtain for each term s a *unique* equivalence class $[t]_A$ such that

- $[s]_A \to^*_{R_A} [t]_A$ and
- $[t]_A$ is not R_A-reducible.

$[t]_A$ is the R_A-normalform of $[s]_A$, which we denote by $[s]_A\downarrow_{R_A}$.

If a modular TRS R/A results by the completion of a given set $A \cup E$ of equations, then we assume like above that the equality of the binary relations $=_{A\cup E}$ and $=_{A\cup R^=}$ is guaranteed by the completion algorithm. For an equational specification $(\Sigma, A \cup E)$ the initial model $\mathcal{I}$ of $A \cup E$ in Theorem 6 corresponds *then* to $(Ter_0(\Sigma)/_{=_{A\cup R^=}}, \{f^{\mathcal{I}} \mid f \in \Sigma\})$ and we have

$$f^{\mathcal{I}}([t_0]_{=_{A\cup R^=}}, \ldots, [t_{n-1}]_{=_{A\cup R^=}}) = [f(t_0, \ldots, t_{n-1})]_{=_{A\cup R^=}}.$$

That is, any rewriting-based algebra that is isomorphic to the initial model of $A \cup R^=$ is clearly an initial model of $A \cup E$, too. This holds also for the following algebra that is based on a modular TRS R/A.

Theorem 10. *Let R/A be a complete modular TRS and $(\Sigma, A \cup R^=)$ be an equational specification. Then the algebra $\mathcal{R}_A = (\{[t]\downarrow_{R_A} \mid [t] \in Ter_0(\Sigma)/A\}, \{f^{\mathcal{R}_A} \mid f \in \Sigma\})$ where*

$$f^{\mathcal{R}_A}([t_0]_A\downarrow_{R_A}, \ldots, [t_{n-1}]_A\downarrow_{R_A}) = [f(t_0, \ldots, t_{n-1})]_A\downarrow_{R_A}$$

is isomorphic to the initial model $\mathcal{I}$ given in Theorem 6.

Proof: We prove that the function $h : Ter_0(\Sigma)/_{A\cup R^=} \to \{[t]\downarrow_{R_A} \mid [t] \in Ter_0(\Sigma)/A\}$ defined by $h([t]_{A\cup R^=}) := [t]_A\downarrow_{R_A}$ is a Σ-homomorphism from $\mathcal{I}$ to $\mathcal{R}_A$ and h is bijective.

Applying the definition of $f^{\mathcal{I}}$ and then that of h, we obtain

$$\begin{aligned} h(f^{\mathcal{I}}([t_0]_{A\cup R^=},\ldots,[t_{n-1}]_{A\cup R^=})) &= h([f(t_0,\ldots,t_{n-1})]_{A\cup R^=}) \\ &= [f(t_0,\ldots,t_{n-1})]_A\downarrow_{R_A}. \end{aligned}$$

Applying the definition of h (n-times) and then that of $f^{\mathcal{R}_A}$, we get

$$\begin{aligned} f^{\mathcal{R}_A}(h([t_0]_{A\cup R^=}),\ldots,h([t_{n-1}]_{A\cup R^=})) &= f^{\mathcal{R}_A}([t_0]_A\downarrow_{R_A},\ldots,[t_{n-1}]_A\downarrow_{R_A}) \\ &= [f(t_0,\ldots,t_{n-1})]_A\downarrow_{R_A}. \end{aligned}$$

That is, we have $h(f^{\mathcal{I}}([t_0]_{A\cup R^=},\ldots,[t_{n-1}]_{A\cup R^=})) = f^{\mathcal{R}_A}(h([t_0]_{A\cup R^=}),\ldots,h([t_{n-1}]_{A\cup R^=}))$ and thus h is a Σ-homomorphism from $\mathcal{I}$ to $\mathcal{R}_A$.

Next, we prove that h is injective: For $t,s \in Ter_0(\Sigma)$ with $h([t]_{A\cup R^=}) = h([s]_{A\cup R^=})$, we have (by the definition of h) $[t]_A\downarrow_{R_A} = [s]_A\downarrow_{R_A}$. Since R/A is a complete modular TRS, we obtain $t =_{A\cup R^=} s$ and thus $[t]_{A\cup R^=} = [s]_{A\cup R^=}$.

Finally, the proof that h is surjective is obvious: If we take any R_A-reduced A-class $[t]_A\downarrow_{R_A}$, we know that $[t]_A\downarrow_{R_A}$ belongs to $Ter_0(\Sigma)/_{A\cup R^=}$ and that $h([t]_A\downarrow_{R_A})$ equals $[t]_A\downarrow_{R_A}$. □

If A induces finite equivalence classes, the initial algebra $\mathcal{R}_A$ is constructive. For our purposes (protocol verification) we may consider sets A as disjoint partitions of *permutative* equations on single function symbols, [94]. These do not only induce finite equivalence classes but additionally allow for simpler definitions of concepts like size (of an object) and (immediate) substructures.

3.3 Message Algebras and Intruder Knowledge

Inductive approaches to protocol verification consider arbitrary many protocol runs with arbitrary many protocol agents. This means that we need an infinite repertoire of items like nonces, passwords, and keys, (see Chap. 7). Therefore we add an enumerable infinite set of atomic messages At to the basic constructs. As an alternative we might add notions for numbers as additional message terms. However, this would complicate the message algebra(s) and cause significant overhead in the proofs. Note that messages are not sorted in any way. In the actual VSE axiomatisation auxiliary notions, like lists, sets, and numbers will be given by separate sorts so that there is no interference with messages (see Chap. 7).

Definition 11 (Message Terms)**.** Let Σ be a signature, V a set of variables and At be an enumerable infinite set of *atomic messages*, which are disjoint with the function symbols in Σ and the variables in V. The set of *message terms* $Mes(\Sigma, At, V)$ is the smallest set satisfying

1. $\Sigma\langle 0\rangle, At, V \subseteq Mes(\Sigma, At, V)$ and
2. $\{f(m_0,\ldots,m_{n-1}) \mid m_0,\ldots,m_{n-1} \in Mes(\Sigma, At, V)\} \subseteq Mes(\Sigma, At, V)$, where $f \in \Sigma\langle n\rangle$ and $n > 0$.

We use $Mes(\Sigma, V)$ for message terms without atomic messages and $Mes_0(\Sigma, At)$ for closed message terms.

In our setting atomic messages as opposed to elements of $\Sigma\langle 0\rangle$ will not satisfy any particular algebraic properties.

Definition 12 (Message Algebra Specification)**.** Let Σ be a signature, V a set of variables, At an enumerable infinite set of *atomic messages* and let E be a set of equations $m_0 = m_1$ where $m_0, m_1 \in Mes(\Sigma, V)$. Then we call $((\Sigma, At), E)$ a *message algebra specification*.

A message algebra specification $((\Sigma, At), A \cup R^{=})$ where R/A is a complete modular TRS allows us to define

1. the set $C = \{[m]_A \downarrow_{R_A} \mid [m]_A \in Mes_0(\Sigma, At)/=_A\}$, consisting of *the messages*, and
2. the set $\{f^{\mathcal{C}} \mid f \in \Sigma \setminus \Sigma\langle 0\rangle\}$, consisting of *the basic operations* defined by

$$f^{\mathcal{C}}([m_0]_A \downarrow_{R_A}, \ldots, [m_{n-1}]_A \downarrow_{R_A}) = [f(m_0, \ldots, m_{n-1})]_A \downarrow_{R_A} .$$

In this context, we interpret the message terms

- $m \in At \cup \Sigma\langle 0\rangle$ by $[\![m]\!] = \{m\}$, and
- $f(m_0, \ldots, m_{n-1}) \in Mes_0(\Sigma, At) \setminus (At \cup \Sigma\langle 0\rangle)$ by

$$[\![f(m_0, \ldots, m_{n-1})]\!] = f^{\mathcal{C}}([\![m_0]\!], \ldots, [\![m_{n-1}]\!]).$$

For all messages $m, m' \in Mes_0(\Sigma, At)$ we have $[\![m]\!] = [\![m']\!] \Leftrightarrow m =_{A \cup R^{=}} m'$: After fixing m and m', it is possible to extend Σ to Σ' such that

1. $\Sigma'\langle 0\rangle$ extends $\Sigma\langle 0\rangle$ with the atomic messages that occur in m and/or m', and
2. $\Sigma'\langle i\rangle = \Sigma\langle i\rangle$ for all $i > 0$.

It is clear that $[\![m]\!] = [\![m]\!]^{\mathcal{R}_A}$ and $[\![m']\!] = [\![m']\!]^{\mathcal{R}_A}$ for the initial model $\mathcal{R}_A$ of the equational specification $(\Sigma', A \cup R^{=})$. For that reason, we may employ $[\![m]\!]^{\mathcal{R}_A} = [\![m']\!]^{\mathcal{R}_A} \Leftrightarrow m =_{A \cup R^{=}} m'$ to get the above equivalence.

For an intruder only a subset $Op \subseteq (\Sigma \setminus \Sigma\langle 0\rangle)$ of function symbols might be available, which restricts her basic *operations* to the elements of $\{f^{\mathcal{C}} \mid f \in Op\}$. The knowledge an intruder is able to gain from an arbitrary but *finite* set of immediate observations $ik \subset Mes_0(\Sigma, At)$ is then given straightforwardly by an inductive definition.

Definition$^{\mathbf{VSE}}$ 13 (DY-Knowledge (Intruder Knowledge)):
Let $Op \subseteq (\Sigma \setminus \Sigma\langle 0\rangle)$ and let $ik \subset Mes_0(\Sigma, At)$. The *DY-knowledge* obtainable from ik using Op, $DY_{Op}(ik)$, is defined by

$$m \in DY_{Op}(ik) \Leftrightarrow (\exists n \in \mathbb{N} : m \in DYl_{Op}(ik, n)), \text{ where}$$

1. $DYl_{Op}(ik, 0) = ik$ and
2. $DYl_{Op}(ik, i+1) = DYl_{Op}(ik, i) \cup \{f(m_0, \ldots, m_{n-1}) \mid f \in Op \cap \Sigma\langle n\rangle, m_{j(<n)} \in DYl_{Op}(ik, i)\}.$

This definition is *generic*, as it depends only on the considered function symbols in Op.

Note that Op is disjoint with $\Sigma\langle 0\rangle$ and this guarantees that $DY_{Op}(\emptyset) = \emptyset$, i.e. the intruder is not able to generate any message from scratch.

A major problem in the inductive approach is to show (by induction) that some secret s does not belong to $DY_{Op}(ik)$ for all observations ik produced by the protocol. For that purpose, we want to "check" elements of ik whether they are *critical* for secrets $\{s, \ldots\}$. From "$\forall m \in ik : m$ is not critical for $\{s, \ldots\}$" should then follow that $\{s, \ldots\} \cap DY_{Op}(ik) = \emptyset$. The "check" function will be specified *recursively* on m (see Chap. 6). In order to obtain such consistent (and suitable) extensions, a thorough analysis of the initial models is indispensable (see Chap. 5). This is based on a complete modular TRS R/A, which we obtain from the equations as described in the next chapter.

Chapter 4

From Equations to a Complete (Modular) TRS

In this chapter we describe how to obtain appropriate complete rewriting relations R/A for given equations E of message algebra specifications. This consists principally in

1. fixing a set $A \subseteq E$ of non-orientable equations,
2. choosing an appropriate orientation of the remaining equations to obtain a first set R_0 of rewrite rules, and
3. looking for a final set R of rewrite rules such that R/A is a complete modular TRS and $=_{A \cup R^=}$ equals $=_{A \cup R_0^=}$.

Since $=_E$ equals $=_{A \cup R_0^=}$, we guarantee in (3) that $=_E$ and $=_{A \cup R^=}$ are equal as required in Sec. 3.2.

The third step is called completion (of R_0) modulo equations (A). The known approaches are based on algorithms for unification modulo A and expect these algorithms to return finite unifier sets, [71, 9, 104]. The accessible tools support to the best of our knowledge only completion modulo associativity and commutativity (AC). For this reason, we propose a completion approach that works for other kinds of modulo theories used in message algebras (see Sec. 4.1– 4.4). This allows us to obtain the rewriting relations for the PACE- and the TC-AMP-algebra (see Sec. 4.5 and 4.6).

4.1 Message Algebra Equations and Their Orientation

We start by defining the features of the usual equations in message algebras, [42].

Definition 14.

1. Let E be a set of equations. We call $l = r \in E$ *a cancellation equation in* E, if the right-hand side r is *either* a sub-term of the left-hand side l *or* $r \in \Sigma\langle 0 \rangle$.
2. We call a set of equations a *FEC-theory*, if it induces finite equivalence classes. The elements of this set are said to be *FEC-equations*.
3. An equation $l = r$ is *permutative*, if the number of occurrences of the symbols in the left-hand side l is the same as in the right-hand side r. A set of permutative equations is called a *permutative theory*.
4. We call a function symbol f *permutative*, if f satisfies a permutative equation $l = r$ where l and r contain no other function symbols.

Permutative theories are FEC-theories, [94].

Cancellation equations are inherently rewrite rules. They allow us for instance to reverse the application of other function symbols, as it is required for decryption of encrypted messages.

For a given equation set E we determine the sets A and R_0 according to the following principle:

- The set E_C of the cancellation equations in E provides us with a first subset $\vec{E_C}$ of R_0. Here, we obtain a first partial order $<_{E_C}$ on terms given by the transitive closure of $\vec{E_C}$ (denoted by $(\vec{E_C})^+$), i.e. we have
$$t <_{E_C} s \text{ iff } s \to^+_{\vec{E_C}} t.$$
- The remaining set $E \setminus E_C$ contains generally equations that are known to be non-orientable. A comprehensive review of equations used in cryptographic protocols shows that we have to deal with two categories of non-orientable equations:
  1. Equations $f(f(x,y),z) = f(x,f(y,z))$ and $f(x,y) = f(y,x)$ specify that f is associative and commutative (AC).
  2. Equations $f(f(x,y),z) = f(f(x,z),y)$ (resp. $f(x,f(y,z)) = f(y,f(x,z))$), specify that f allows for the permutation of the sub-terms at the positions $1 \cdot 2$ and 2 (resp. $2 \cdot 1$ and 1).

  We include all equations of categories (1) and (2) in A.
- Finally, we appropriately orient the remaining equations in $E \setminus (E_C \cup A)$ to get the complementary subset of R_0. Hereby, we obtain a partial order $<_{R_0}$ on terms given (as well) by the transitive closure of R_0/A, i.e. we have
$$t <_{R_0} s \text{ iff } s \to^+_{R_0/A} t.$$
  Obviously, we have $<_{E_C} \subseteq <_{R_0}$.

The rewriting relation R_0/A is not necessarily complete. So, we need to check this property and adapt or extend the rewrite rules if needed. For this purpose we developed a completion approach where nice features of the considered permutative equations are exploited: We first introduce the benefit of these features for rewriting modulo the considered equations (Sec. 4.2), then provide an analysis of local confluence (Sec. 4.3), and we finally describe the completion procedure (Sec. 4.4).

4.2 Rewriting modulo Specific Permutative Equations

In the rest of this chapter we assume that the permutative theories A are on permutative function symbols, where equations are defined on single function symbols. Permutative theories on different function symbols of this kind can be seen as independent. This allows us to separate the application of the permutative equations in modular rewriting into *linked* and *decoupled* equational derivations. We may thus abstract away from the decoupled ones during the analysis of local confluence (see Sec. 4.3).

Before we provide the theorem that allows us to separate linked and decoupled equational derivations, we first introduce underlying notions.

Definition 15.

1. We use $p \bullet q$ to denote the concatenation of two subterm positions: For $p = i_0 \cdot \ldots \cdot i_{N(i)-1} \cdot \epsilon$ and $q = j_0 \cdot \ldots \cdot j_{N(j)-1} \cdot \epsilon$, we have $p \bullet q = i_0 \cdot \ldots \cdot i_{N(i)-1} \cdot j_0 \cdot \ldots \cdot j_{N(j)-1} \cdot \epsilon$.

   We use $\mathbf{Pos}(t)$ to denote the set of the sub-term positions in t.

   We define $\mathbf{LPos}(t) := \{p \mid p \in \mathbf{Pos}(t) \text{ and } t|p \text{ is a constant or a variable}\}$, which consists of the positions of the *atomic* sub-terms of t.

   We define $\mathbf{FPos}(t) := \mathbf{Pos}(t) \setminus \mathbf{LPos}(t)$, which consists of the positions of the *composed* sub-terms of t.

   For $p, q \in \mathbf{Pos}(t)$, we say p is *below* q iff $p = q \bullet (i_0 \cdot \ldots \cdot i_n)$, i.e. $t|p$ occurs as a proper subterm in $t|q$. We say p and q are *non-overlapping* iff $p \neq q$, p is not below q and q is not below p.

2. Let f be a function symbol in $\Sigma\langle n \rangle$ with $n > 0$ and let t be an arbitrary term. We define the *f-structure* of t, denoted $Struct_f(t)$, by:

   (a) $Struct_f(t) = \emptyset$, if t is a constant (resp. a variable) or the top-symbol of t differs from f,

   (b) and $Struct_f(t) = \{\epsilon\} \cup \{1 \bullet p \mid p \in Struct_f(t|1)\} \cup \ldots \cup \{n \bullet p \mid p \in Struct_f(t|n)\}$, when f is the top-symbol of t.

   We say t is a *f-term*, if $Struct_f(t) \neq \emptyset$.

   The subterms of t at positions $q \in \{i \bullet p \mid 0 < i \leq n \text{ and } p \in Struct_f(t)\}$ are called the *f-subterms* of t.

   Every f-subterm that is not a f-term is called to be *basic*.

3. Let p in $\mathbf{FPos}(t)$ satisfy

   - $t|p$ has a permutative function symbol f as its top-symbol,
   - and there is no $q \in \mathbf{FPos}(t)$ such that $q \cdot i = p$ and $t|q$ has f as top-symbol.

   We call $\{p \bullet q \mid q \in Struct_f(t|p)\}$ *a non-rigid pattern* in t and p its *top-position*.

   We call $q \in \mathbf{FPos}(t)$ *a rigid occurrence* of a function symbol if q does not belong to any non-rigid pattern.

   For $q \in \mathbf{FPos}(t)$, we use $\hat{q}$ to denote the top-position of the non-rigid pattern q belongs to, if any, and q itself, otherwise.

As for example, the set A in the TC-AMP algebra will contain equations on '$*$' and on '$\oplus$' (see Sec. 4.6). A term $t = fst(*(c,*(inv(b),*(a,*(b,v)))))$ contains one non-rigid pattern (having $*$ as its permutative function symbol). It is given by $\{1, 1 \cdot 2, 1 \cdot 2 \cdot 2, 1 \cdot 2 \cdot 2 \cdot 2\}$, where 1 is the corresponding top-position in t. This induces $\widehat{1 \cdot 2} = \widehat{1 \cdot 2 \cdot 2} = 1 \cdot \widehat{2 \cdot 2 \cdot 2} = 1$. The sub-term positions ϵ and $1 \cdot 2 \cdot 1$ are rigid occurrences.

We have defined non-rigid patterns relative to the permutative function symbols (occurring in A), because the application of equations in A (preceding the rule matching) in a modular rewrite step *permutes* sub-terms only at positions below the top-positions of these patterns. Note that the number of function symbol occurrences (the rigid occurrences as well as in non-rigid patterns) is preserved.

In every permutative equation $l = r$, l includes one non-rigid pattern that transforms to the sole non-rigid pattern of r. There is no rigid occurrence of function symbols.

Since $\mathbf{Pos}(l)$ and $\mathbf{Pos}(r)$ have the same cardinality, and all symbols in l have the same number of occurrences in r, there must be a *bijective* mapping $\hbar[l,r] \subset \mathbf{Pos}(l) \times \mathbf{Pos}(r)$ that satisfies the following conditions:

1. $(\epsilon,\epsilon) \in \hbar[l,r]$ and
2. $(p,q) \in \hbar[l,r] \Leftrightarrow$
   $((p \in \mathbf{FPos}(l) \wedge q \in \mathbf{FPos}(r)) \vee (p \in \mathbf{LPos}(l) \wedge q \in \mathbf{LPos}(r) \wedge l|p = r|q)$.

We associate every permutative equation $l = r$ with such a *bijective* mapping that we denote $\hbar[l,r]$. When $l = r$ is defined on a single function symbol, $\hbar[l,r]$ permits to *characterize* any equational derivation step $t \rightarrow^{p}_{(l,r)} s$ as explained below. The inverse mapping $(\hbar[l,r])^{-1}$, which corresponds clearly to a bijective mapping $\hbar[r,l] \subset \mathbf{Pos}(r) \times \mathbf{Pos}(l)$ satisfying as well conditions (1) and (2), permits to characterize similarly any equational derivation step $t \rightarrow^{p}_{(r,l)} s$.

Definition 16. Let $l = r$ be some permutative equation (on a single function symbol) and let t be an arbitrary term having a sub-term that matches l. Using the bijective mapping $\hbar[l,r] \subset \mathbf{Pos}(l) \times \mathbf{Pos}(r)$ associated with $l = r$, we define the following function $\hbar[t,(l,r)]$ from $\{p \mid p \in \mathbf{FPos}(t)$ and $t|p$ matches $l\} \times \mathbf{Pos}(t)$ to a finite set of term positions by:

$$\hbar[t,(l,r)](p,q) = \begin{cases} p \bullet \hbar[l,r](p_l) & : \quad q = p \bullet p_l \text{ and } p_l \in \mathbf{Pos}(l) \\ p \bullet \hbar[l,r](p_l) \bullet p_t & : \quad q = p \bullet p_l \bullet p_t \text{ and } p_l \in \mathbf{LPos}(l) \\ q & : \quad \text{otherwise} \end{cases}$$

If we fix p for some $t|p = l\sigma$ and we put the top-symbol of every sub-term $t|q$ at the position $\hbar[t,(l,r)](p,q)$, we obtain the term $t[r\sigma]_p$. As $t[r\sigma]_p$ satisfies $t \rightarrow^{p}_{(l,r)} t[r\sigma]_p$, we say that the function $\hbar[t,(l,r)]$ characterizes any transformation of t by a left-right application of $l = r$ (or by an equational derivation step using (l,r)). In fact, the function $q \mapsto \hbar[t,(l,r)](p,q)$ is a *bijective* mapping from $\mathbf{Pos}(t)$ to $\mathbf{Pos}(t[r\sigma]_p)$, which fulfills the following properties.

Theorem 17. *Let A be a set of permutative equations on same function symbols, $(l,r) \in \overrightarrow{A} \cup \overleftarrow{A}$ and let t and s satisfy $t \rightarrow^{p}_{(l,r)} s$. Then the mapping $\hbar[t,(l,r)]$ satisfies the following properties:*

1. *For $q_1, q_2 \in \mathbf{FPos}(t)$, we have*
   (a) *$\hbar[t,(l,r)](p,\hat{q_2})$ is below $\hbar[t,(l,r)](p,\hat{q_1})$ if $\hat{q_2}$ is below $\hat{q_1}$, and*
   (b) *$\hbar[t,(l,r)](p,\hat{q_1})$ and $\hbar[t,(l,r)](p,\hat{q_2})$ are non-overlapping if $\hat{q_1}$ and $\hat{q_2}$ are non-overlapping.*
2. *For all $q \in \mathbf{FPos}(t)$, we have $t|\hat{q} =_A s|\hbar[t,(l,r)](p,\hat{q})$.*
3. *For $q_1, q_2 \in \mathbf{FPos}(t)$ such that $\hat{q_1}$ and $\hat{q_2}$ are non-overlapping, we have $t[.,.]_{\hat{q_1},\hat{q_2}} =_A s[.,.]_{\hbar[t,(l,r)](p,\hat{q_1}),\hbar[t,(l,r)](p,\hat{q_2})}$.*

Proof: We start with property (1-a): Since $\hat{q_2}$ is below $\hat{q_1}$, we have $\hat{q_2} = \hat{q_1} \bullet q_3$ for $q_3 \neq \epsilon$. According to the definition of $\hbar[t,(l,r)]$, we distinguish the following cases:

1. $\hat{q_1} = p \bullet p_l$ for $p_l \in \mathbf{Pos}(l)$: If $p_l \in \mathbf{FPos}(l)$, $\hat{q_1}$ and p belong to a same non-rigid pattern. As $\hat{q_1}$ is the top-position of this pattern, we obtain $p_l = \epsilon$, $\hat{q_1} = p$ and $\hbar[t,(l,r)](\hat{q_1}) = \hat{q_1} = p$. This means, $\hat{q_2} = p \bullet q_3$ is of the form $p \bullet p_l$ or $p \bullet p_l \bullet p_t$. That is, $\hbar[t,(l,r)](\hat{q_2})$ equals $p \bullet \hbar[l,r](p_l)$ or $p \bullet \hbar[l,r](p_l) \bullet p_t$. Hence, $\hbar[t,(l,r)](\hat{q_2})$ is below $\hbar[t,(l,r)](\hat{q_1})$, in both cases.

   Otherwise, $p_l \in \mathbf{LPos}(l)$ implies $\hbar[t,(l,r)](\hat{q_1}) = p \bullet \hbar[l,r](p_l)$, $\hat{q_2} = \hat{q_1} \bullet q_3$ is of the form $p \bullet p_l \bullet q_3$ and $\hbar[t,(l,r)](\hat{q_2}) = p \bullet \hbar[l,r](p_l) \bullet q_3$. Hence, $\hbar[t,(l,r)](\hat{q_2})$ is below $\hbar[t,(l,r)](\hat{q_1})$.

2. $\hat{q_1} = p \bullet p_l \bullet p_t$ for $p_l \in \mathbf{LPos}(l)$: This implies $\hbar[t,(l,r)](\hat{q_1}) = p \bullet \hbar[l,r](p_l) \bullet p_t$, $\hat{q_2} = \hat{q_1} \bullet q_3$ is of the form $p \bullet p_l \bullet p_t \bullet q_3$ and $\hbar[t,(l,r)](\hat{q_2}) = p \bullet \hbar[l,r](p_l) \bullet p_t \bullet q_3$. Hence, $\hbar[t,(l,r)](\hat{q_2})$ is below $\hbar[t,(l,r)](\hat{q_1})$.

3. $\hat{q_1} \neq p \bullet p_l \bullet p_t$ for $p_l \in \mathbf{Pos}(l)$: Here, $\hbar[t,(l,r)](\hat{q_1}) = \hat{q_1}$.

   If $\hat{q_2} = \hat{q_1} \bullet q_3$ is of the form $p \bullet p_l$ or $p \bullet p_l \bullet p_t$, we obtain p is below $\hat{q_1}$. This implies $\hbar[t,(l,r)](\hat{q_2}) = p \bullet \hbar[l,r](p_l)$ or $\hbar[t,(l,r)](\hat{q_2}) = p \bullet \hbar[l,r](p_l) \bullet p_t$. Hence, $\hbar[t,(l,r)](\hat{q_2})$ is below $\hat{q_1}$, i.e. $\hbar[t,(l,r)](\hat{q_1})$, as required.

   Otherwise, $\hbar[t,(l,r)](\hat{q_2}) = \hat{q_2}$ and thus $\hbar[t,(l,r)](\hat{q_2})$ is trivially below $\hbar[t,(l,r)](\hat{q_1})$

Next, we prove property (1-b) by contraposition: Assuming that $\hbar[t,(l,r)](p,\hat{q_1})$ and $\hbar[t,(l,r)](p,\hat{q_2})$ are not non-overlapping, yields to the following cases:

1. $\hbar[t,(l,r)](p,\hat{q_1}) = \hbar[t,(l,r)](p,\hat{q_2})$: Since the function $q \mapsto \hbar[t,(l,r)](p,q)$ is bijective, we obtain $\hat{q_1} = \hat{q_2}$. Hence, $\hat{q_1}$ and $\hat{q_2}$ are not non-overlapping.

2. $\hbar[t,(l,r)](p,\hat{q_1})$ is below $\hbar[t,(l,r)](p,\hat{q_2})$: Focusing on $\hbar[t,(l,r)](p,\hat{q_2})$, the definition of $\hbar[t,(l,r)]$ provides three cases:

   (a) $\hat{q_2} = p \bullet p_l$, $p_l \in \mathbf{Pos}(l)$ and $\hbar[t,(l,r)](p,\hat{q_2}) = p \bullet \hbar[l,r](p_l)$: Similar to the proof of (1-a), we obtain two cases:

   - $\hat{q_2} = p$ and $\hbar[t,(l,r)](p,\hat{q_2}) = p$: Since the position $\hbar[t,(l,r)](p,\hat{q_1})$ is below $\hbar[t,(l,r)](p,\hat{q_2})$, we obtain $\hbar[t,(l,r)](p,\hat{q_1})$ is of the form $p \bullet q_3$ for $q_3 \neq \epsilon$. According to the definition of $\hbar[t,(l,r)]$, we get either (i) $\hat{q_1} = p \bullet p'_l \bullet p'_t$ for $p'_l \in \mathbf{Pos}(l)$ or (ii) $\hat{q_1} = p \bullet q_3$. In both cases, $\hat{q_1}$ is below $\hat{q_2}$. Hence, $\hat{q_1}$ and $\hat{q_2}$ are not non-overlapping.
   - $p_l \in \mathbf{LPos}(l)$ implies $\hbar[l,r](p_l) = p_r$ for $p_r \in \mathbf{LPos}(r)$: Since $\hbar[t,(l,r)](p,\hat{q_1})$ is below $\hbar[t,(l,r)](p,\hat{q_2})$, we obtain $\hbar[t,(l,r)](p,\hat{q_1})$ is of the form $p \bullet p_r \bullet q_3$ for $q_3 \neq \epsilon$. According to the definition of $\hbar[t,(l,r)]$, $\hat{q_1}$ must be of the form $p \bullet p_l \bullet q_3$. This means that $\hat{q_1}$ is below $\hat{q_2}$. Hence, $\hat{q_1}$ and $\hat{q_2}$ are not non-overlapping.

   (b) $\hat{q_2} = p \bullet p_l \bullet p_t$, $p_l \in \mathbf{LPos}(l)$ and $\hbar[t,(l,r)](p,\hat{q_2}) = p \bullet \hbar[l,r](p_l) \bullet p_t$: As in the previous case, we have $\hbar[l,r](p_l) = p_r$ for $p_r \in \mathbf{LPos}(r)$ and this yields $\hbar[t,(l,r)](p,\hat{q_1}) = p \bullet p_r \bullet p_t \bullet q_3$ for $q_3 \neq \epsilon$. Similarly, we obtain $\hat{q_1} = p \bullet p_l \bullet p_t \bullet q_3$. This means that $\hat{q_1}$ is below $\hat{q_2}$. Hence, $\hat{q_1}$ and $\hat{q_2}$ are not non-overlapping.

   (c) $\hat{q_2} \neq p \bullet p_l \bullet p_t$ for $p_l \in \mathbf{Pos}(l)$ and $\hbar[t,(l,r)](p,\hat{q_2}) = \hat{q_2}$: Since $\hbar[t,(l,r)](p,\hat{q_1})$ is below $\hbar[t,(l,r)](p,\hat{q_2})$, we obtain $\hbar[t,(l,r)](p,\hat{q_1})$ is of the form $\hat{q_2} \bullet q_3$ for $q_3 \neq \epsilon$. According to the definition of $\hbar[t,(l,r)]$, we distinguish the following cases:

   - $\hat{q_1} = p$ and $\hbar[t,(l,r)](p,\hat{q_1}) = p$: That is, $\hat{q_1} = \hat{q_2} \bullet q_3$. This means that $\hat{q_1}$ is below $\hat{q_2}$. Hence, $\hat{q_1}$ and $\hat{q_2}$ are not non-overlapping.
   - $\hat{q_1} = p \bullet p_l$, $p_l \in \mathbf{LPos}(l)$ and $\hbar[t,(l,r)](p,\hat{q_1}) = p \bullet \hbar[l,r](p_l)$: That is, $p \bullet \hbar[l,r](p_l) = \hat{q_2} \bullet q_3$. Since $\hat{q_2} \neq p \bullet p_l \bullet p_t$, p must be below $\hat{q_2}$. This means that $\hat{q_1}$ is below $\hat{q_2}$. Hence, $\hat{q_1}$ and $\hat{q_2}$ are not non-overlapping.
   - $\hat{q_1} = p \bullet p_l \bullet p_t$, $p_l \in \mathbf{LPos}(l)$ and $\hbar[t,(l,r)](p,\hat{q_1}) = p \bullet \hbar[l,r](p_l) \bullet p_t$: That is, $p \bullet \hbar[l,r](p_l) \bullet p_t = \hat{q_2} \bullet q_3$. Since $\hat{q_2} \neq p \bullet p_l \bullet p_t$, p must be below $\hat{q_2}$. This means that $\hat{q_1}$ is below $\hat{q_2}$. Hence, $\hat{q_1}$ and $\hat{q_2}$ are not non-overlapping.
   - $\hat{q_1} \neq p \bullet p_l \bullet p_t$ for $p_l \in \mathbf{Pos}(l)$ and $\hbar[t,(l,r)](p,\hat{q_1}) = \hat{q_1}$: That is, $\hat{q_1} = \hat{q_2} \bullet q_3$. This means that $\hat{q_1}$ is below $\hat{q_2}$. Hence, $\hat{q_1}$ and $\hat{q_2}$ are not non-overlapping.

3. $\hbar[t,(l,r)](p,\hat{q_2})$ is below $\hbar[t,(l,r)](p,\hat{q_1})$: It is similar to case (2).

Regarding property (2), the definition of $\hbar[t,(l,r)]$ implies $t|q' = s|\hbar[t,(l,r)](q')$ for all q' different from $p \bullet p_l$ for $p_l \in \mathbf{FPos}(l)$. This means, we have $t|\hat{q} = s|\hbar[t,(l,r)](\hat{q})$ if $\hat{q} \neq p \bullet p_l$ for $p_l \in \mathbf{FPos}(l)$.

In the complementary case, i.e. $\hat{q} \neq p \bullet p_l$ for $p_l \in \mathbf{FPos}(l)$, we obtain $\hat{q} = p$ as $\hat{q}$ and p belong to the same non-rigid pattern. That is, we have $\hbar[t,(l,r)](\hat{q}) = p$, $t|\hat{q} = l\sigma$ and $s|\hbar[t,(l,r)](\hat{q}) = r\sigma$. Hence, it follows $t|\hat{q} =_A s|\hbar[t,(l,r)](\hat{q})$, as required.

Finally, we prove property (3) by the following case distinction:

1. $\hat{q_1}$ equals p or p is below $\hat{q_1}$: Here, we have $\hbar[t,(l,r)](\hat{q_1}) = \hat{q_1}$, $t|\hat{q_1} = c_1[l\sigma]$ and $s|\hat{q_1} = c_1[r\sigma]$.

   Since $\hat{q_2}$ is non-overlapping with $\hat{q_1}$, $\hat{q_2}$ is also non-overlapping with p. This means, $\hbar[t,(l,r)](\hat{q_2}) = \hat{q_2}$ and $t|\hat{q_2} = s|\hat{q_2}$.

   For $c_2 = t|\hat{q_2}$, we obtain $t[c_1[l\sigma],c_2]_{\hat{q_1},\hat{q_2}} =_A t[c_1[r\sigma],c_2]_{\hat{q_1},\hat{q_2}} = s[c_1[r\sigma],c_2]_{\hat{q_1},\hat{q_2}}$. Hence, it follows $t[.,.]_{\hat{q_1},\hat{q_2}} =_A s[.,.]_{\hbar[t,(l,r)](\hat{q_1}),\hbar[t,(l,r)](\hat{q_2})}$.

2. $\hat{q_1}$ is below p with $\hat{q_1} = p \bullet p_{l1} \bullet p_{t1}$ for $p_{l1} \in \mathbf{LPos}(l)$: Here, we have $\hbar[t,(l,r)](\hat{q_1}) = p \bullet \hbar[l,r](p_{l1}) \bullet p_{t1}$ and $(l\sigma)|p_{l1} = c_1[t|\hat{q_1}]_{p_{t1}} = (r\sigma)|\hbar[l,r](p_{l1})$.

   Since $\hat{q_2}$ is non-overlapping with $\hat{q_1}$, p may not be equal or below $\hat{q_2}$, which yields two cases:

   (a) The position $\hat{q_2}$ is below p with $\hat{q_2} = p \bullet p_{l2} \bullet p_{t2}$ for $p_{l2} \in \mathbf{LPos}(l)$: This means, $\hbar[t,(l,r)](\hat{q_2}) = p \bullet \hbar[l,r](p_{l2}) \bullet p_{t2}$ and $(l\sigma)|p_{l2} = c_2[t|\hat{q_2}]_{p_{t2}} = (r\sigma)|\hbar[l,r](p_{l2})$. That is, we have

   $$\begin{aligned} t[t|\hat{q_1},t|\hat{q_2}]_{\hat{q_1},\hat{q_2}} &= t[(l\sigma)[c_1[t|\hat{q_1}]_{p_{t1}},c_2[t|\hat{q_2}]_{p_{t2}}]_{p_{l1},p_{l2}}]_p \\ &=_A t[(r\sigma)[c_1[t|\hat{q_1}]_{p_{t1}},c_2[t|\hat{q_2}]_{p_{t2}}]_{\hbar[l,r](p_{l1}),\hbar[l,r](p_{l2})}]_p \\ &= s[s|\hbar[t,(l,r)](\hat{q_1}),s|\hbar[t,(l,r)](\hat{q_2})]_{\hbar[t,(l,r)](\hat{q_1}),\hbar[t,(l,r)](\hat{q_2})}. \end{aligned}$$

   (b) p and $\hat{q_2}$ are non-overlapping: Like in (1), we have $\hbar[t,(l,r)](\hat{q_2}) = \hat{q_2}$ and $t|\hat{q_2} = s|\hat{q_2}$. For $c_2 = t|\hat{q_2}$, we obtain

   $$\begin{aligned} t[t|\hat{q_1},c_2]_{\hat{q_1},\hat{q_2}} &= t[(l\sigma)[c_1[t|\hat{q_1}]_{p_{t1}}]_{p_{l1}},c_2]_{p,\hat{q_2}} \\ &=_A t[(r\sigma)[c_1[t|\hat{q_1}]_{p_{t1}}]_{\hbar[l,r](p_{l1})},c_2]_{p,\hat{q_2}} \\ &= s[s|\hbar[t,(l,r)](\hat{q_1}),s|\hbar[t,(l,r)](\hat{q_2})]_{\hbar[t,(l,r)](\hat{q_1}),\hbar[t,(l,r)](\hat{q_2})}. \end{aligned}$$

   Hence, it follows $t[.,.]_{\hat{q_1},\hat{q_2}} =_A s[.,.]_{\hbar[t,(l,r)](\hat{q_1}),\hbar[t,(l,r)](\hat{q_2})}$ in (a) and (b).

3. $\hat{q_1}$ and $\hat{q_2}$ are non-overlapping with p: Like in (1), we have $\hbar[t,(l,r)](\hat{q_1}) = \hat{q_1}$, $t|\hat{q_1} = s|\hat{q_1}$, $\hbar[t,(l,r)](\hat{q_2}) = \hat{q_2}$ and $t|\hat{q_2} = s|\hat{q_2}$. For $c_1 = t|\hat{q_1}$ and $c_2 = t|\hat{q_2}$, we obtain $t[c_1,c_2]_{\hat{q_1},\hat{q_2}} =_A s[c_1,c_2]_{\hat{q_1},\hat{q_2}}$. Hence, it follows $t[.,.]_{\hat{q_1},\hat{q_2}} =_A s[.,.]_{\hbar[t,(l,r)](\hat{q_1}),\hbar[t,(l,r)](\hat{q_2})}$. □

The properties in theorem 17 can be transferred (by induction) to equational derivations with several steps. For arbitrary $t =_A s$, we obtain thus a *bijective* mapping from $\mathbf{Pos}(t)$ to $\mathbf{Pos}(s)$ that characterizes the transformation of t to s and fulfills the same properties like $\hbar[t,(l,r)]$.

Theorem 18. *Let A be a set of permutative equations on same function symbols and let t and s be two syntactically different terms with $t =_A s$. Then there is a bijective mapping $\hbar[t,s]$ from $\mathbf{Pos}(t)$ to $\mathbf{Pos}(s)$ fulfilling the following properties:*

1. *For all $q \in \mathbf{FPos}(t)$, we have $\widehat{\hbar[t,s](\hat{q})} = \hbar[t,s](\hat{q})$.*

2. *For $q_1, q_2 \in \mathbf{FPos}(t)$, we have*
   (a) *$\hbar[t,s](\hat{q_2})$ is below $\hbar[t,s](\hat{q_1})$ if $\hat{q_2}$ is below $\hat{q_1}$, and*
   (b) *$\hbar[t,s](\hat{q_1})$ and $\hbar[t,s](\hat{q_2})$ are non-overlapping if $\hat{q_1}$ and $\hat{q_2}$ are non-overlapping.*
3. *For all $q \in \mathbf{FPos}(t)$, we have $t|\hat{q} =_A s|\hbar[t,s](\hat{q})$.*
4. *For $q_1, q_2 \in \mathbf{FPos}(t)$ such that $\hat{q_1}$ and $\hat{q_2}$ are non-overlapping, we have $t[.,.]_{\hat{q_1},\hat{q_2}} =_A s[.,.]_{\hbar[t,s](\hat{q_1}),\hbar[t,s](\hat{q_2})}$.*

Proof: The proof is by induction on the number of the equational derivation steps that transform t to s.

In the base case, we have $t \rightarrow^{p_1}_{(l_1,r_1)} s$. Here, we define $\hbar[t,s] : q \mapsto \hbar[t,(l_1,r_1)](p_1,q)$. Properties (2)–(4) hold due to theorem 17. Property (1) is shown by the following case distinction:

1. $\hat{q} = p \bullet p_l$ for $p_l \in \mathbf{FPos}(l_1)$: This holds only when $\hat{q} = p$ and if $p = p_1 \cdot i_1$ then $t|p$ and $t|p_1$ have different top-symbols. Here, we have $\hbar[t,(l_1,r_1)](p,\hat{q}) = p = \hat{q}$ and $\hbar[t,\widehat{(l_1,r_1)}](p,\hat{q}) = \hat{p} = \hat{q}$, as required.
2. $\hat{q} = p \bullet p_l$ for $p_l \in \mathbf{LPos}(l_1)$: This holds only when the top-symbol of $t|\hat{q}$ differs from the top-symbol of l_1. Here, we have $\hbar[t,(l_1,r_1)](p,\hat{q}) = p \bullet \hbar[l_1,r_1](p_l)$. Since $l_1|p_l = r_1|\hbar[l_1,r_1](p_l)$, $t|\hat{q}$ and $s|\hbar[t,(l_1,r_1)](p,\hat{q})$ have the same top-symbol. Furthermore, if $\hbar[l_1,r_1](p_l) = p'_l \cdot i'$ then $s|(p \bullet p'_l)$ has the same top-symbol like l_1. Hence, $\hbar[t,\widehat{(l_1,r_1)}](p,\hat{q}) = \hbar[t,(l_1,r_1)](p,\hat{q})$, as required.
3. $\hat{q} = p \bullet p_l \bullet p_t$ for $p_l \in \mathbf{LPos}(l_1)$: The proof is similar to the previous case.
4. $\hat{q} \neq p \bullet p_l \bullet p_t$ for $p_l \in \mathbf{Pos}(l_1)$: Here, we have $\hbar[t,(l_1,r_1)](p,\hat{q}) = \hat{q}$, which trivially implies $\hbar[t,\widehat{(l_1,r_1)}](p,\hat{q}) = \hbar[t,(l_1,r_1)](p,\hat{q})$.

In the step case, we have $t \rightarrow^{p_1}_{(l_1,r_1)} t_2, \ \ldots, \ t_n \rightarrow^{p_n}_{(l_n,r_n)} s$ for $n > 1$. The induction hypothesis yields $\hbar[t,t_n]$ with the required properties, which we use to define $\hbar[t,s] : q \mapsto \hbar[t_n,(l_n,r_n)](p_n,\hbar[t,t_n](q))$, where $q \mapsto \hbar[t_n,(l_n,r_n)](p_n,q)$ characterizes the last derivation step. The required properties of $\hbar[t,s]$ are shown (also based on theorem 17) as follows:

- To show $\widehat{\hbar[t,s]}(\hat{q}) = \hbar[t,s](\hat{q})$ for property (1), we use $\widehat{\hbar[t,t_n]}(\hat{q}) = \hbar[t,t_n](\hat{q})$ (provided by the induction hypothesis). This permits to reduce the proof goal to the equality $\hbar[t_n,(l_n,r_n)]\widehat{(p_n,\hbar[t,t_n]}(\hat{q})) = \hbar[t_n,(l_n,r_n)](p_n,\widehat{\hbar[t,t_n]}(\hat{q}))$. This proof goal can be shown as in the base case.
- To show property (2-a), we consider arbitrary $q_1, q_2 \in \mathbf{FPos}(t)$ such that $\hat{q_2}$ is below $\hat{q_1}$. By the induction hypothesis, $\hbar[t,t_n](\hat{q_2})$ must be below $\hbar[t,t_n](\hat{q_1})$. Then, theorem 17 and property (1) imply that $\hbar[t_n,(l_n,r_n)](p_n,\hbar[t,t_n](\hat{q_2}))$ is below $\hbar[t_n,(l_n,r_n)](p_n,\hbar[t,t_n](\hat{q_1}))$. Hence, $\hbar[t,s](\hat{q_2})$ is below $\hbar[t,s](\hat{q_1})$, as required.
- Property (2-b) is shown according to the same principle as in the proof of (2-a).
- To show property (3), we use $t|\hat{q} =_A t_n|\hbar[t,t_n](\hat{q})$ (provided by the induction hypothesis) and $t_n|\hbar[t,t_n](\hat{q}) =_A s|\hbar[t_n,(l_n,r_n)](p_n,\hbar[t,t_n](\hat{q}))$ (provided by theorem 17 and property (1)). This trivially yields $t|\hat{q} =_A s|\hbar[t_n,(l_n,r_n)](p_n,\hbar[t,t_n](\hat{q})) = s|\hbar[t,s](\hat{q})$, as required.
- To show property (4), we use

(a) $t[.,.]_{\hat{q_1},\hat{q_2}} =_A t_n[.,.]_{\hbar[t,t_n](\hat{q_1}),\hbar[t,t_n](\hat{q_2})}$

(b) and $t_n[.,.]_{\hbar[t,t_n](\hat{q_1}),\hbar[t,t_n](\hat{q_2})} =_A s[.,.]_{\hbar[t_n,(l_n,r_n)](p_n,\hbar[t,t_n](\hat{q_1})),\hbar[t_n,(l_n,r_n)](p_n,\hbar[t,t_n](\hat{q_2}))}$.

(a) is provided by the induction hypothesis and (b) by theorem 17 together with property (1). This trivially yields $t[.,.]_{\hat{q_1},\hat{q_2}} =_A s[.,.]_{\hbar[t,s](\hat{q_1}),\hbar[t,s](\hat{q_2})}$, as required. □

The inverse equational derivation of t from s can be characterized the same way with a bijective mapping $\hbar[s,t] \subset \mathbf{Pos}(s) \times \mathbf{Pos}(t)$ corresponding to the *inverse* function of $\hbar[t,s]$. This means, $\hbar[s,t] = (\hbar[t,s])^{-1}$.

Theorem 18 allows us to *localize* the scope of the rule application in every rewriting step $t \to_{R/A} s$, by identifying a sub-term of t where the application of equations from A might influence the application of the rule from R.

Theorem 19. *Let A be a set of permutative equations on same function symbols, $l \to r$ a rewrite rule, and let t,t',s,s' be terms such that $t =_A t'$, $t' \to^p_{l\to r} s'$, $s' =_A s$. Then we have*

1. $t'|\hat{p} =_A t|\hbar[t',t](\hat{p})$
2. $p = \hat{p} \bullet q$ *(with q equals ϵ or a position of a proper subterm in $t'|\hat{p}$) and there is a substitution σ with $(t'|\hat{p})|q = l\sigma$, $(s'|\hat{p})|q = r\sigma$ and $(t'|\hat{p})[.]_q = (s'|\hat{p})[.]_q$.*

   If $q = q_1 \ldots q_n$, i.e. q is a position of a proper subterm in $t'|\hat{p}$, then $\hat{p} \bullet (q_1 \cdot \ldots \cdot q_j)$ is in $\mathbf{FPos}(t'|\hat{p})$ for $1 \leq j \leq n$. This means, all subterms $t'|\hat{p}$, $t'|\hat{p} \bullet (q_1)$, …, $t'|\hat{p} \bullet (q_1 \ldots q_{n-1})$ and $t'|p$ have the same top-symbol, which equals the top-symbol of l.
3. $s'[.]_{\hat{p}} = t'[.]_{\hat{p}} =_A t[.]_{\hbar[t',t](\hat{p})}$, *and*

   $s'[.]_{\hat{p}} =_A s[.]_{\hbar[s',s](\hat{p})}$

Proof: Properties (1)–(3) can be easily proven using our definition of a rewriting step (cp. Def. 4-(1)) and the results of Theorem 18. □

Theorem 19 provides us with a term context ($t[.]_{\hbar[t',t](\hat{p})} =_A t'[.]_{\hat{p}} = s'[.]_{\hat{p}} =_A s[.]_{\hbar[s',s](\hat{p})}$) where equality modulo A is *decoupled* from the rewrite step. Contrarily, the rewrite step in the corresponding subterm $t'|\hat{p} =_A t|\hbar[t',t](\hat{p})$ could be influenced by equality modulo A. This means, the rewrite step and equality modulo A are *linked* in this subterm, which we call *the application scope of the rewrite rule*. It is given by the top-symbol of the rule left-hand side and the subterm position of the (pure) reduction step: When the rewrite rule is applied to t' at position p for $t =_A t'$, the application scope of this rewrite rule in t is at position $\hbar[t',t](\hat{p})$.

According to property (2), (pure) rewriting replaces a subterm of the application scope and preserves the corresponding context. We call the former (part of the application scope) *the rewrite scope* and the latter *the equational context*, which can be clearly empty. In this case, the application scope and the rewrite scope coincide.

Recall the above example $t = fst(*(c,*(inv(b),*(a,*(b,v)))))$ from the TC-AMP algebra. We will see that the subterm $*(inv(b),*(a,*(b,v)))$ at position $1 \cdot 2$ is reducible with a rule $*(x,*(inv(x),y)) \to y$ to $*(a,v)$. The scope of this rule application corresponds to the subterm $*(c,*(inv(b),*(a,*(b,v))))$, which determines the context $fst(.)$ where the application of any equation from A can be decoupled from the considered rule application.

The example makes clear that the application scope of a rule is not limited to the replaced subterm (the rewrite scope $*(inv(b),*(a,*(b,v))) =_A *(b,*(inv(b),*(a,v)))$). If A contains equations on the top-symbol of l, the application scope can have additionally a non-empty equational context. It corresponds to $*(c,.)$ in our example.

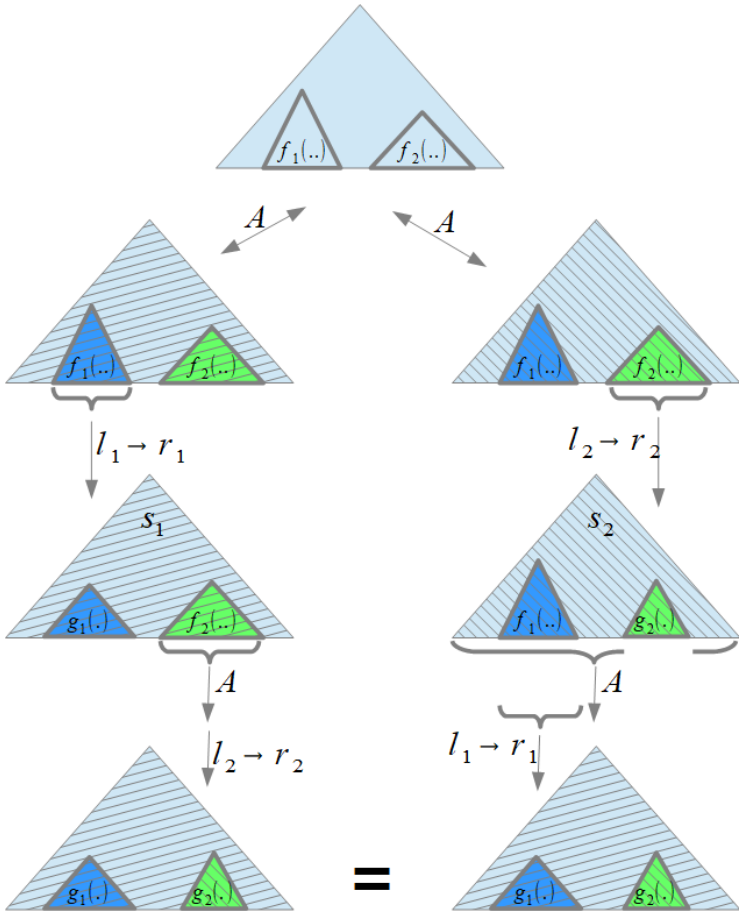

Figure 4.1: Separated Application Scopes

4.3 Analysis of Local Confluence

Based on the results of Sec. 4.2, we want to determine the joining requirements that we may *focus on* in our completion procedure.

We consider in the rest of this section two arbitrary reduction steps $t \to_{R/A} s_1$ and $t \to_{R/A} s_2$. They provide us with (intermediate) terms t_1, s_1', t_2, s_2', rules $l_1 \to r_1, l_2 \to r_2$, non-variable subterm positions p_1, p_2, and with substitutions σ_1, σ_2 such that

(i) $t =_A t_1$, $t_1 \to^{p_1}_{l_1 \to r_1} s_1'$, $s_1' =_A s_1$, $t_1|p_1 = l_1\sigma_1$ and

(ii) $t =_A t_2$, $t_2 \to^{p_2}_{l_2 \to r_2} s_2'$, $s_2' =_A s_2$, $t_2|p_2 = l_2\sigma_2$.

We describe how to join s_1 and s_2 in case the application scopes are separated (see Sec. 4.3.1) and in case of an overlap at a variable position (see Sec. 4.3.3). Additionally, we analyze all possible situations for overlapping application scopes (see Sec. 4.3.2) and identify corresponding joining requirements to focus on in our completion algorithm.

4.3.1 Separated Application Scopes:

The application scopes of $l_1 \to r_1$ and $l_2 \to r_2$ are separated, if the subterm positions $\hat{p_1}$ and $\hbar[t_2, t_1](\hat{p_2})$ are non-overlapping. Here, s_1 and s_2 can be joined as depicted in Fig 4.1:

We show below how s_1 and s_2 can be joined based on the following facts:

1. The rewrite step (i) provides $s_1 =_A s_1'[s_1'|\hat{p_1}]_{\hat{p_1}} = t_1[s_1'|\hat{p_1}]_{\hat{p_1}}$. Since $\hat{p_1}$ and $\hbar[t_2,t_1](\hat{p_2})$ are non-overlapping, we obtain $s_1 =_A t_1[s_1'|\hat{p_1}, t_1|\hbar[t_2,t_1](\hat{p_2})]_{\hat{p_1},\hbar[t_2,t_1](\hat{p_2})}$.
2. $t_1|\hbar[t_2,t_1](\hat{p_2}) =_A t_2|\hat{p_2}$ follows from $t_1 =_A t_2$.
3. In rewrite step (2) we have $p_2 = \hat{p_2} \bullet q$ and this yields $t_2|\hat{p_2} \rightarrow^{q_2}_{l_2\rightarrow r_2} s_2'|\hat{p_2}$.

By rewriting the subterm $t_1|\hbar[t_2,t_1](\hat{p_2})$ ($f_2(..)$ in Fig. 4.1) of s_1 with $l_2 \rightarrow r_2$, we get $s_1 \rightarrow_{R/A} t_1[s_1'|\hat{p_1}, s_2'|\hat{p_2}]_{\hat{p_1},\hbar[t_2,t_1](\hat{p_2})}$.

Similarly, we apply $l_1 \rightarrow r_1$ to the subterm $t_2|\hbar[t_1,t_2](\hat{p_1})$ ($f_1(..)$ in Fig. 4.1) of s_2 and obtain $s_2 \rightarrow_{R/A} t_2[s_2'|\hat{p_2}, s_1'|\hat{p_1}]_{\hat{p_2},\hbar[t_1,t_2](\hat{p_1})}$.

Since $t_1 =_A t_2$ and positions $\hat{p_1}$ and $\hbar[t_2,t_1](\hat{p_2})$ are non-overlapping, theorem 18 allows us to use the equality $t_1[.,.]_{\hat{p_1},\hbar[t_2,t_1](\hat{p_2})} =_A t_2[.,.]_{\hbar[t_1,t_2](\hat{p_1}),\hbar[t_1,t_2](\hbar[t_2,t_1](\hat{p_2}))}$, which can be simplified to $t_1[.,.]_{\hat{p_1},\hbar[t_2,t_1](\hat{p_2})} =_A t_2[.,.]_{\hbar[t_1,t_2](\hat{p_1}),\hat{p_2}}$. This allows us to conclude with $t_1[s_1'|\hat{p_1}, s_2'|\hat{p_2}]_{\hat{p_1},\hbar[t_2,t_1](\hat{p_2})} =_A t_2[s_2'|\hat{p_2}, s_1'|\hat{p_1}]_{\hat{p_2},\hbar[t_1,t_2](\hat{p_1})}$.

4.3.2 Overlapping Application Scopes:

The application scopes of $l_1 \rightarrow r_1$ and $l_2 \rightarrow r_2$ are overlapping, if $\hat{p_1} = \hbar[t_2,t_1](\hat{p_2})$, $\hat{p_1}$ is below $\hbar[t_2,t_1](\hat{p_2})$ or vice versa. We want to analyze all possible situations and identify corresponding joining requirements.

4.3.2.1 Same Application Scope

We focus first on the case where $\hat{p_1} = \hbar[t_2,t_1](\hat{p_2})$. This means, $t_1|p_1$ and $t_2|p_2$ have a same top-symbol f, which must be as well the top-symbol of l_1 and l_2. In particular, $l_1 \rightarrow r_1$ and $l_2 \rightarrow r_2$ have the same application scope modulo A. Since $t_1[.]_{\hat{p_1}} =_A t_2[.]_{\hat{p_2}}$ and the sufficient conditions (below) for joining s_1 and s_2 do not depend on these term contexts, we assume w.l.o.g. that $t_1[.]_{\hat{p_1}} = t_2[.]_{\hat{p_2}} = [.]$, i.e. $\hat{p_1} = \hat{p_2} = \epsilon$, $t_1|\hat{p_1} = t_1$ and $t_2|\hat{p_2} = t_2$.

There are mainly four cases regarding the interference of the rewrite scopes of $l_1 \rightarrow r_1$ and $l_2 \rightarrow r_2$, i.e. of $l_1\sigma_{12}$ and $l_2\sigma_{12}$ for $\sigma_{12} = \sigma_1 \cup \sigma_2$:

1. We have two *separable* rewrite scopes, if $t_1 =_A c[l_1\sigma_{12}, l_2\sigma_{12}]$ and $t_2 =_A c[l_1\sigma_{12}, l_2\sigma_{12}]$.

   Here, we obtain $s_1 =_A c[r_1\sigma_{12}, l_2\sigma_{12}]$ and $s_2 =_A c[l_1\sigma_{12}, r_2\sigma_{12}]$. This allows us to join s_1 and s_2 similar to the case of separated application scopes (see Sec. 4.3.1).

2. The rewrite scopes of $l_1 \rightarrow r_1$ and $l_2 \rightarrow r_2$ are equal modulo A. That is, we have $l_1\sigma_{12} =_A l_2\sigma_{12}$ and there is a common context $c[.]_{p_1} = c[.]_{p_2}$ with $t_1 =_A c[l_1\sigma_{12}]_{p_1}$ and $t_2 =_A c[l_2\sigma_{12}]_{p_1}$. This means in particular that l_1 and l_2 are A-unifiable. Let $\mathcal{U}_A(l_1,l_2)$ be a complete, finite set of unifiers. If we make sure that $r_1\sigma$ and $r_2\sigma$ for all $\sigma \in \mathcal{U}_A(l_1,l_2)$ are join-able, then s_1 and s_2 are join-able, as well[1]: Since l_1 and l_2 are A-unifiable, the rewrite of $t_1 =_A c[l_1\sigma_{12}]_{p_1}$ to $s_1 =_A c[r_1\sigma_{12}]_{p_1}$ and of $t_2 =_A c[l_2\sigma_{12}]_{p_1}$ to $s_2 =_A c[r_2\sigma_{12}]_{p_1}$ includes an *overlap*, in the sense of [9]. It is the overlap of $l_2 \rightarrow r_2$ on the non-variable position ϵ of $l_1 \rightarrow r_1$ given by the common substitution σ_{12}. According to the extended critical pair lemma, [70, 9], there is $\sigma \in \mathcal{U}_A(l_1,l_2)$ and a second substitution ρ such that $x\sigma_{12} =_A (x\sigma)\rho$ for all variables x in $l_1 \rightarrow r_1$ or $l_2 \rightarrow r_2$. Ensuring that $r_1\sigma$ and $r_2\sigma$ are join-able allows us to replay the corresponding rewriting steps to join $r_1\sigma_{12}$ and $r_2\sigma_{12}$. Obviously, we only need to replace the variables x in $r_1\sigma$ and $r_2\sigma$ with their counterparts in the of ρ, i.e. with $x\rho$.

[1] The variables in l_1 are assumed to be distinct from those in l_2. In the general approach, this is obtained by renaming the variables in l_1 to obtain l_1'. Furthermore, the superposition of l_2 on l_1' is not necessarily at the top-position ϵ (as in our case), but it is at an arbitrary non-variable sub-term position μ of l_1'. So, the most general unifiers σ must belong to $\mathcal{U}_A(l_1'|\mu, l_2)$

3. The rewrite scope of $l_2 \to r_2$ is *included* in that of $l_1 \to r_1$, or vice versa.

   W.l.o.g., we focus on the former case, where $t_1 =_A c[l_1\sigma_{12}]$, $t_2 =_A c[c'[l_2\sigma_{12}]]$ and $l_1\sigma_{12} =_A c'[l_2\sigma_{12}]$ for a non-empty f-context $c'[.]$. Here, the basic f-subterms in $l_1\sigma_{12}$ are distributed to $c'[.]$ and $l_2\sigma_{12}$. Furthermore, we distinguish whether there is $l_1' =_A l_1$ and a sub-term position ν of a variable f-subterm in l_1' such that $l_1\sigma_{12} =_A l_1'\sigma_{12}[c''[l_2\sigma_{12}]]_\nu$ holds.

   (a) If it is the case, we have an overlap of $l_2 \to r_2$ on $l_1 \to r_1$ at a variable position. Here, the terms s_1 and s_2 can be joined canonically, as described in Sec. 4.3.3.

   (b) For the complementary case, we provide in Sec. 4.4.2 sufficient conditions to join $r_1\sigma_{12}$ and $c'[r_2\sigma_{12}]$, which extends clearly to $s_1 =_A c[r_1\sigma_{12}]$ and $s_2 =_A c[c'[r_2\sigma_{12}]]$.

4. The rewrite scopes of $l_1 \to r_1$ and $l_2 \to r_2$ are *partly overlapping*, if $t_1 =_A c[c_2[l_1\sigma_{12}]]$ and $t_2 =_A c[c_1[l_2\sigma_{12}]]$ for non-empty f-contexts $c_2[.]$ and respectively $c_1[.]$ that are composed from f-subterms in $l_2\sigma_{12}$ and $l_1\sigma_{12}$, respectively. This means, $l_1\sigma_{12}$ and respectively $l_2\sigma_{12}$ includes f-subterms (in $c_1[.]$ and respectively in $c_2[.]$) outside of $l_2\sigma_{12}$ and outside of $l_1\sigma_{12}$, respectively. They can be included into $l_1\sigma_{12}$ and respectively into $l_2\sigma_{12}$ *only* if f is permutative. Here, we say that we have an *equational overlap* between $l_1 \to r_1$ and $l_2 \to r_2$, since this overlap is caused by the equational contexts of these rules.

   In Sec. 4.4.2, we provide sufficient conditions to join $c_2[r_1\sigma_{12}]$ and $c_1[l_2\sigma_{12}]$, which extends clearly to $s_1 =_A c[c_2[r_1\sigma_{12}]]$ and $s_2 =_A c[c_1[r_2\sigma_{12}]]$.

Note that cases (1) and (4) are obsolete, if f is not permutative.

Recapitulating, the above analysis yields to the following cases to be handled in our completion algorithm:

- Case (2): For all $\sigma \in \mathcal{U}_A(l_1, l_2)$, we have to ensure that $r_1\sigma$ and $r_2\sigma$ are join-able.
- Case (3-b): There is an overlap of $l_2 \to r_2$ on $l_1 \to r_1$ at a non-variable position of l_1. In Sec. 4.4.2, we provide sufficient conditions to join $r_1\sigma_{12}$ and $c'[r_2\sigma_{12}]$ for $\sigma_{12} = \sigma_1 \cup \sigma_2$.
- Case (4): There is an equational overlap between $l_1 \to r_1$ and $l_2 \to r_2$, which is caused by the equational contexts of these rules. In Sec. 4.4.2, we provide sufficient conditions to join $c_2[r_1\sigma_{12}]$ and $c_1[l_2\sigma_{12}]$ for $\sigma_{12} = \sigma_1 \cup \sigma_2$.

4.3.2.2 Nested Application Scope

Next, we analyze the case where $\hbar[t_2, t_1](\hat{p_2})$ is below $\hat{p_1}$. Let f be the head symbol of l_1, then there is a non-variable basic f-subterm t_1^2 of $t_1|\hat{p_1}$ such that $t_1^2 =_A c_2[t_2|\hat{p_2}]$. Here, the sufficient conditions for joining s_1 and s_2 do not depend on the term context $t_1[.]_{\hat{p_1}}$. For this reason, we assume w.l.o.g. that $t_1[.]_{\hat{p_1}} = [.]$, i.e. $\hat{p_1} = \epsilon$ and $t_1|\hat{p_1} = t_1$.

In contrast to the case analyzed in Sec. 4.3.2.1, l_2 can have an arbitrary head symbol g. There are mainly two cases regarding the interference of the rewrite scopes of $l_1 \to r_1$ and $l_2 \to r_2$, i.e. of $l_1\sigma_{12}$ and $l_2\sigma_{12}$ for $\sigma_{12} = \sigma_1 \cup \sigma_2$:

1. We have two *separable* rewrite scopes, if $t_1 =_A c[l_1\sigma_{12}, l_2\sigma_{12}]$ and $t_2 =_A c[l_1\sigma_{12}, l_2\sigma_{12}]$.

   This is similar to case (1) in Sec. 4.3.2.1.

2. The rewrite scope of $l_2 \to r_2$ is *included* in that of $l_1 \to r_1$. This means, $t_1 =_A c[l_1\sigma_{12}]$, $t_2 =_A c[c'[l_2\sigma_{12}]]$ and $l_1\sigma_{12} =_A c'[l_2\sigma_{12}]$ for a non-empty context $c'[.]$. Here, we distinguish whether there is $l_1' =_A l_1$ and a sub-term position ν of a variable in l_1' such that $l_1\sigma_{12} =_A l_1'\sigma_{12}[c''[l_2\sigma_{12}]]_\nu$ holds.

(a) If it is the case, we have an overlap of $l_2 \to r_2$ on $l_1 \to r_1$ at a variable position. We proceed similar to case (3-a) in Sec. 4.3.2.1.

(b) Otherwise, there must be a position μ at or below the position of a basic f-subterm in l_1 such that $\hat{\mu} = \mu$, and $l_1|\mu$ has the same head symbol g as l_2. Regarding the occurrence of $l_2\sigma_{12}$, we distinguish again two alternatives:

i. $(l_1|\mu)\sigma_{12} =_A l_2\sigma_{12}$: Similar to case (2) in Sec. 4.3.2.1, we require that $r_1\sigma$ and $(l_1[r_2]_\mu)\sigma$ are join-able for all $\sigma \in \mathcal{U}_A((l_1|\mu), l_2)$. This is sufficient to join s_1 and s_2.

ii. $(l_1|\mu)\sigma_{12} =_A c''[l_2\sigma_{12}]$ for a non-empty g-context $c''[.]$: Similar to case (3-b), we provide in Sec. 4.4.2 sufficient conditions to join the terms $r_1\sigma_{12}$ and $l_1\sigma_{12}[c''[r_2\sigma_{12}]]_\mu$, which extends clearly to $s_1 =_A c[r_1\sigma_{12}]$ and $s_2 =_A c[c'[r_2\sigma_{12}]]$.

Note that the occurrence of $t_2|\hat{p_2}$ (the application scope of $l_2 \to r_2$) in a non-variable basic f-subterm t_1^2 of $t_1|\hat{p_1}$ (the application scope of $l_1 \to r_1$) prevents that f-subterms from $t_1|\hat{p_1}$ are permuted into $t_2|\hat{p_2}$. For that reason, case (4) in Sec. 4.3.2.1 is excluded in case of a nested application scope.

4.3.3 Overlap at a Variable Position:

In this section, we consider cases (3-a) of Sec. 4.3.2.1 and (2-a) of Sec. 4.3.2.2, where we have an overlap of $l_2 \to r_2$ on $l_1 \to r_1$ at a variable position. This means, $t_1|p_1 =_A c_{l_1}\sigma_{12}[l_2\sigma_{12}]$ for a context $c_{l_1}\sigma_{12}[.]$ that is given by a position ν of a variable in l_1. For $x = l_1|\nu$, we have $x\sigma_{12} =_A c''[l_2\sigma_{12}] =_A (l_1\sigma_{12})|\nu$ and $t_1[l_1\sigma_{12}]_{p_1} =_A t_1[l_1\sigma_{12}[c''[l_2\sigma_{12}]]_\nu]_{p_1} =_A t_2[l_2\sigma_{12}]_{p_2}$. This implies in particular that the contexts $t_1[l_1\sigma_{12}[c''[.]]_\nu]_{p_1}$ and $t_2[.]_{p_2}$ are equal modulo A. If x occurs n-times in l_1 for $n > 1$, there must be $n-1$ further positions $\nu_1, \ldots, \nu_{n-1}$ of x in l_1 such that $(l_1\sigma_{12})|\nu_1 = \ldots = (l_1\sigma_{12})|\nu_{n-1} =_A c''[l_2\sigma_{12}]$.

In the following, we prove that s_1 and s_2 are join-able, for arbitrary n and arbitrary m, the number of occurrence of x in r_1. The basic principle is depicted in Fig 4.2, where the shown transformations correspond to a $l_1 \to r_1$ with two occurrences of x in l_1 and one occurrence in r_1.

Let $\mu_1', \ldots, \mu_m'$ be the sub-term positions of the variable x in r_1. Then, we know that $s_1 =_A s_1'$ and $s_1' = t_1[r_1\sigma_{12}]_{p_1} =_A t_1[r_1\sigma_{12}[c''[l_2\sigma_{12}], \ldots, c''[l_2\sigma_{12}]]_{\mu_1', \ldots, \mu_m'}]_{p_1}$. This allows us to apply the rule $l_2 \to r_2$ below the positions $\mu_1', \ldots, \mu_m'$ and rewrite this way s_1 to $t_1[r_1\sigma_{12}[c''[r_2\sigma_{12}], \ldots, c''[r_2\sigma_{12}]]_{\mu_1', \ldots, \mu_m'}]_{p_1}$. The same term can be obtained by rewriting s_2 as follows:

1. Using $s_2 =_A s_2' = t_2[r_2\sigma_{12}]_{p_2}$ and $t_2[.]_{p_2} =_A t_1[l_1\sigma_{12}[c''[.]]_\nu]_{p_1}$, we deduce $s_2 =_A t_1[l_1\sigma_{12}[c''[r_2\sigma_{12}]]_\nu]_{p_1} =_A t_1[l_1\sigma_{12}[c''[r_2\sigma_{12}], c''[l_2\sigma_{12}], \ldots, c''[l_2\sigma_{12}]]_{\nu, \nu_1, \ldots, \nu_{n-1}}]_{p_1}$. This allows us to rewrite s_2 through $n-1$ applications of $l_2 \to r_2$ and obtain $t_1[l_1\sigma_{12}[c''[r_2\sigma_{12}], c''[r_2\sigma_{12}], \ldots, c''[r_2\sigma_{12}]]_{\nu, \nu_1, \ldots, \nu_{n-1}}]_{p_1}$.

2. It is clear that the term $l_1\sigma_{12}[c''[r_2\sigma_{12}], c''[r_2\sigma_{12}], \ldots, c''[r_2\sigma_{12}]]_{\nu, \nu_1, \ldots, \nu_{n-1}}$ matches l_1 modulo A. The resulting substitution σ_{12}' maps all variables v that differ from the variable x to $v\sigma_{12}$ and maps x to $c''[r_2\sigma_{12}]$ (instead of $c''[l_2\sigma_{12}]$). This allows us to rewrite $t_1[l_1\sigma_{12}[c''[r_2\sigma_{12}], c''[r_2\sigma_{12}], \ldots, c''[r_2\sigma_{12}]]_{\nu, \nu_1, \ldots, \nu_{n-1}}]_{p_1}$ at position p_1 to $t_1[r_1\sigma_{12}[c''[r_2\sigma_{12}], \ldots, c''[r_2\sigma_{12}]]_{\mu_1', \ldots, \mu_m'}]_{p_1}$.

4.4 Completion Procedure

In this section, we describe our algorithm on how to check the local confluence of rewrite rules used for rewriting modulo a set A of permutative equations. Based on the analysis in Sec. 4.3, we want to determine the pairs of terms that are needed to be joined, in order to

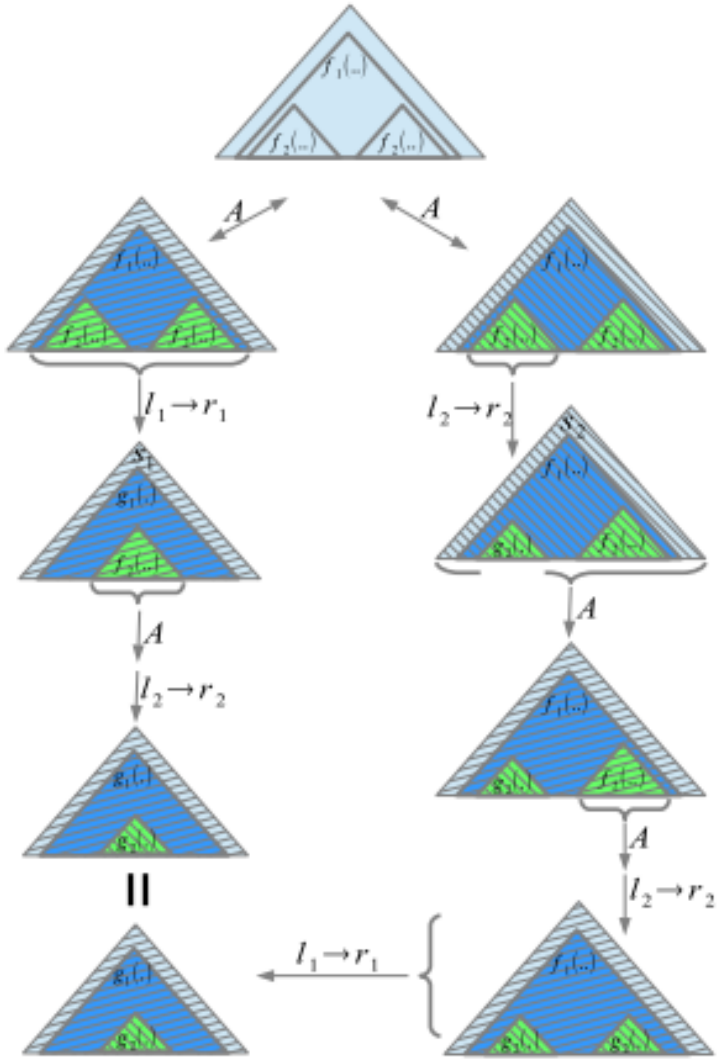

Figure 4.2: Overlap at a Variable Position

ensure the local confluence. Like in other completion approaches, certain pairs necessitate to include new rewrite rules.

For rewrite rules *without occurrence* of permutative function symbols, equational overlaps as discussed in case (4) of Sec. 4.3.2.1 are not relevant. Regarding the remaining cases discussed in Sec. 4.3.2.1 and 4.3.2.2, it is sufficient to ensure the following: For $l_2 \to r_2$ and $l_1 \to r_1$ where the top-symbol of l_2 occurs at position μ of l_1 and $l_2\sigma = (l_1|\mu)\sigma$ holds for a (*syntactic*) unifier σ, we need to join $r_1\sigma$ and $(l_1[r_2]_\mu)\sigma$.

In presence of rewrite rules with function symbols f occurring in A, the restriction to permutative equations allows us to reach our goal by solving mainly two tasks for everyone of the permutative theories A_f:

1. coming up with a *decomposition rule* that applies to term pairs $f(t_1,\ldots,t_n)$ and $f(s_1,\ldots,s_n)$ in a *uniform* unification algorithm, and

2. providing *uniformly* sufficient conditions to deal with cases (3-b) and (4) in Sec. 4.3.2.1

and with case (2-b-ii) in Sec. 4.3.2.2.

4.4.1 Unification by Decomposition

As in other completion approaches, we also relate on complete sets of unifiers modulo the equations in A. But, we do not use existing theory-specific unification algorithms. Instead of that, we compute the unifiers (required for our completion process) by a *uniform* algorithm that expands the search space by the application of theory-specific decomposition rules.

The decomposition decisions are based on a partition of the function symbols into

- F_A, which includes the function symbols occurring in A,
- and F', which includes the function symbols non-occurring in A.

The search space is an or-tree, since the decomposition yields in general more than one alternative. The nodes correspond to sets of term pairs or to $\bot$, which signals a failed search path. The root node contains generally one term pair, which represents the original unification problem. The leaf nodes that are different from $\bot$ are sets of the form $\{(v_0,t_0),\ldots,(v_{n-1},t_{n-1})\}$, where

- $v_0,\ldots,v_{n-1}$ are distinct variables, and
- $t_0,\ldots,t_{n-1}$ are terms that do not contain any variable v_i.

These leaf nodes are called to be in *solved form* and provide us with the required complete set of unifiers.

The expansion of the search space is carried out by interleaving two procedures:

1. SIMPLIFY brings the root node (if not yet) and the nodes resulting by decomposition into simplified forms:
   (a) It checks for clashes: If there is a term pair $(v,f(t_1,\ldots,t_n))$ with $f \in \Sigma\langle n\rangle$ and some t_i contains v, or there is a term pair $(f(t_1,\ldots,t_n),f'(s_1,\ldots,s_{n'}))$ for two different function symbols $f \in \Sigma\langle n\rangle, f' \in \Sigma\langle n'\rangle$, the node is replaced with $\bot$.
   (b) It applies intermediate substitutions: A node $\{(v,t)\} \cup S$, where v occurs in S and not in t, is replaced with $\{(v,t)\} \cup (S[t/v])$.
   (c) It eliminates trivial pairs (t,t).
2. DECOMPOSE selects a pair $(f(t_1,\ldots,t_n),f(s_1,\ldots,s_n))$ and generates the successor nodes in the search space according to the corresponding decomposition rule:
   - We first handle the cases where $f \in F'$: Such term pairs are decomposed as in syntactic unification, i.e. they are replaced by term pairs $(t_1,s_1),\ldots,(t_n,s_n)$.
   - In the complementary cases, i.e. $f \in F_A$, we apply corresponding *theory-specific* decomposition rules.

For our purpose, i.e. unification in permutative theories used in cryptographic protocols, we assume that each A_f is associated with a decomposition rule of the form

$$f(x_1,\ldots,x_n) = f(y_1,\ldots,y_n) \Rightarrow ((x_1 = y_1 \wedge \ldots \wedge x_n = y_n) \vee \psi_1^f \vee \ldots \vee \psi_{n_f}^f),$$

where $\psi_1^f,\ldots,\psi_{n_f}^f$ are equality constraints, i.e. (existentially quantified) conjunctions of equations. The implication is supposed to be satisfied in the initial models of A_f. It provides us clearly with a decomposition rule to apply for the unification of term pairs $(f(t_1,\ldots,t_n),f(s_1,\ldots,s_n))$.

In the completion of the equations of PACE and TC-AMP we employed decomposition rules that we obtained as follows:

1. For AC-theories A_f, similar to $A_\oplus$, we obtain the corresponding decomposition rule from

$$\begin{aligned}
&f(x_0,x_1) = f(x_2,x_3) \Rightarrow \\
&((x_0 = x_2 \wedge x_1 = x_3) \vee \\
&\ (x_0 = x_3 \wedge x_1 = x_2) \vee \\
&\ (\exists x_4 : x_0 = f(x_2,x_4) \wedge x_3 = f(x_4,x_1)) \vee \\
&\ (\exists x_4 : x_0 = f(x_3,x_4) \wedge x_2 = f(x_4,x_1)) \vee \\
&\ (\exists x_4 : x_1 = f(x_2,x_4) \wedge x_3 = f(x_4,x_0)) \vee \\
&\ (\exists x_4 : x_1 = f(x_3,x_4) \wedge x_2 = f(x_4,x_0)) \vee \\
&\ (\exists x_4,x_5,x_6,x_7 : x_0 = f(x_4,x_5) \wedge x_1 = f(x_6,x_7) \wedge \\
&\qquad\qquad x_2 = f(x_4,x_6) \wedge x_3 = f(x_5,x_7))).
\end{aligned}$$

2. For $A_f = \{f(f(x,y),z) = f(f(x,z),y)\}$, similar to A_{dh}, we obtain the corresponding decomposition rule from

$$\begin{aligned}
&f(x_0,x_1) = f(x_2,x_3) \Rightarrow \\
&((x_0 = x_2 \wedge x_1 = x_3) \vee \\
&\ (\exists x_4 : x_0 = f(x_4,x_3) \wedge x_2 = f(x_4,x_1))).
\end{aligned}$$

3. For $A_f = \{f(x,f(y,z)) = f(y,f(x,z))\}$, similar to A_*, we obtain the corresponding decomposition rule from

$$\begin{aligned}
&f(x_0,x_1) = f(x_2,x_3) \Rightarrow \\
&((x_0 = x_2 \wedge x_1 = x_3) \vee \\
&\ (\exists x_4 : x_1 = f(x_2,x_4) \wedge x_3 = f(x_0,x_4))).
\end{aligned}$$

Each decomposition rule is expected to reduce the unification of term pairs back to unification constraints on their subterms. This is necessary for the termination of the unification procedure, which we assume at least for the unification problems that arise in the completion process.

4.4.2 Dealing with Specific Overlaps

In this section, we provide sufficient joining requirements to deal with the overlaps of cases (3-b) and (4) in Sec. 4.3.2.1 and of case (2-b-ii) in Sec. 4.3.2.2. We first determine the requirements for the permutative theory $A_f = \{f(x,f(y,z)) = f(y,f(x,z))\}$ (see Sec. 4.4.2.1) and then for the AC theory (see Sec. 4.4.2.2). The approach in Sec. 4.4.2.1 can be straightforwardly adapted to the permutative theory $A_f = \{f(f(x,y),z) = f(f(x,z),y)\}$.

4.4.2.1 Permutative Function Symbols like $*$:

Let $A_f = \{f(x,f(y,z)) = f(y,f(x,z))\}$ be the permutative theory on $f \in \Sigma\langle 2\rangle$. We want to provide the joining requirements to deal with overlaps between two arbitrary rewrite rules $l_1 \to r_1$ and $l_2 \to r_2$ where l_2 is an f-term. We focus first on cases (3-b) and (4) in Sec. 4.3.2.1.

4.4.2.1.1 Joining Requirements for Cases (3-b) and (4): In these cases, l_1 must be an f-term, too. Here,

1. case (3-b) corresponds to an overlap of l_2 on l_1 at a position $\mu \in Struct_f(l_1) \setminus \{\epsilon\}$
2. and case (4) corresponds to an equational overlap between l_1 and l_2.

The joining requirements for case (4) make use of the set of context pairs $Ctxts_f(l_1,l_2)$, which we define as follows:

Definition 20. Let $l_1 = f(l_1^1,\ldots,f(l_{n_1}^1,l^1)\ldots)$ and $l_2 = f(l_1^2,\ldots,f(l_{n_2}^2,l^2)\ldots)$ for $n_1 \geq 1$, $n_2 \geq 1$ and basic f-subterms l^1 and l^2. Then the set $Ctxts_f(l_1,l_2)$ consists of all context pairs $(f(l_{j_1}^2,\ldots,f(l_{j_n}^2,.)\ldots),f(l_{i_1}^1,\ldots,f(l_{i_m}^1,.)\ldots))$, where $\{j_1,\ldots,j_n\}$ is a non-empty subset of $\{1,\ldots,n_2\}$ and $\{i_1,\ldots,i_m\}$ a non-empty subset of $\{1,\ldots,n_1\}$.

The overlaps in cases (3-b) and (4) are dealt with by the following joining requirements:

1. For all $l_1' =_A l_1$, $\mu \in Struct_f(l_1') \setminus \{\epsilon\}$ and $\sigma \in \mathcal{U}_A(l_1'|\mu,l_2)$, ensure that $r_1\sigma$ and $(l_1'[r_2]_\mu)\sigma$ are join-able. This addresses case (3-b).
2. For all $(c_2[.],c_1[.]) \in Ctxts_f(l_1,l_2)$ and $\sigma \in \mathcal{U}_A(c_2[l_1],c_1[l_2])$, ensure that $c_2[r_1]\sigma$ and $c_1[r_2]\sigma$ are join-able. This addresses case (4).
3. When l^1 is a variable, it can match an f-term with arbitrary many f-subterms not included in the rewrite scope of l_2. To ensure local confluence by our approach, we need to pop up these f-subterms outside the rewrite scope of $l_1 \to r_1$, too. This yields to the requirement that for all new variables x_1,x_2 the equalities $l_1\{f(x_1,x_2)/l^1\} =_A f(x_1,l_1\{x_2/l^1\})$ and $r_1\{f(x_1,x_2)/l^1\} =_A f(x_1,r_1\{x_2/l^1\})$ hold.
   Similarly, if l^2 is a variable, the equalities $l_2\{f(x_1,x_2)/l^2\} =_A f(x_1,l_2\{x_2/l^2\})$ and $r_2\{f(x_1,x_2)/l^2\} =_A f(x_1,r_2\{x_2/l^2\})$ must hold for new variables x_1,x_2.

The third requirement means intuitively that the f-subterms of any f-term that matches l^1 (or l^2) as a variable are preserved as f-subterms after the reduction step.

4.4.2.1.2 Justification of Joining Requirements for Cases (3-b) and (4): In order to justify that the given joining requirements are sufficient, we prove the following theorem about the application scope of $l_1 \to r_1$, given above. Here, $\overline{f}(L,x)$ is defined by $\overline{f}(\epsilon,x) = x$ and $\overline{f}(y{\bullet}L',x) = f(y,\overline{f}(L',x))$. Furthermore, we use $e_l = e_r$ to abbreviate $f(x,f(y,z)) = f(y,f(x,z))$ in A_f and their associated mappings $\hbar[e_l,e_r] \subset \mathbf{Pos}(e_l) \times \mathbf{Pos}(e_r)$ and $\hbar[e_r,e_l] \subset \mathbf{Pos}(e_r) \times \mathbf{Pos}(e_l)$ correspond to

$$\hbar[e_l,e_r] = \hbar[e_r,e_l] = \{(\epsilon,\epsilon),(2,2),(2\cdot 2,2\cdot 2),(1,2\cdot 1),(2\cdot 1,1)\}.$$

The induced functions $\hbar[t,(e_l,e_r)]$ and $\hbar[t,(e_r,e_l)]$ according to Def. 16 correspond to:

$$(p,q) \mapsto \begin{cases} p \bullet (2\cdot 1) \bullet r & : \quad q = p \bullet (1) \bullet r \\ p \bullet (1) \bullet r & : \quad q = p \bullet (2\cdot 1) \bullet r \\ q & : \quad \text{otherwise} \end{cases}$$

We denote this mapping simply by $\hbar[t,\{e_l,e_r\}]$.

Theorem 21. *Let t and s be two f-terms and let p_1 be in $Struct_f(t)$. If $t|p_1 = l_1\sigma_1$ and $t =_A s$, then there is q_1 with $\hbar[t,s](p_1) = q_1 \bullet (\overbrace{2\cdot\ldots 2}^{n\times})$ for $0 \leq n$ and $s|q_1$ is of the form $\overline{f}(L_0,f(\tau_{i_1},\overline{f}(L_1,\ldots,\overline{f}(L_{n_1-1},f(\tau_{i_{n_1}},\overline{f}(L_{n_1},\tau^1)))\ldots)))$, where τ^1 is not an f-term, $\tau_j =_A l_j^1\sigma_1$ for $1 \leq j \leq n_1$, and*

1. *either l^1 is a variable and $l^1\sigma_1 =_A \overline{f}(L^1,\tau^1)$ for a list L^1 consisting of elements from $L_0\#\ldots\#L_{n_1}$*
2. *or l^1 is not a variable, $l^1\sigma_1 =_A \tau^1$, and L_0 is empty.*

Proof: The proof is by induction on the number of the equational derivation steps to obtain s from t.

In the base case, where $t = s$, the proof is trivial: We set $q_1 = p_1$, $L_0, \ldots, L_{n_1-1}$ are empty and $\overline{f}(L_{n_1}, \tau^1) = l^1\sigma_1$, where L_{n_1} can be not empty only if l^1 is a variable.

In the step case, we have $t =_A t'$ and $t' \rightarrow^p_{(l,r)} s$ for $(l,r) \in \overrightarrow{A} \cup \overleftarrow{A}$. Using the induction hypothesis, we obtain q_1 with $\hbar[t,t'](p_1) = q_1 \bullet (\overbrace{2 \cdot \ldots 2}^{n\times})$ for $0 \leq n$ and $t'|q_1$ is of the form $\overline{f}(L_0, f(\tau_{i_1}, \overline{f}(L_1, \ldots, \overline{f}(L_{n_1-1}, f(\tau_{i_{n_1}}, \overline{f}(L_{n_1}, \tau^1))) \ldots)))$

If $(l,r) \notin \{(e_l,e_r),(e_r,e_l)\}$, the form of $t'|q_1$ is preserved, because the last derivation step is performed inside a basic f-subterm of t': First, the last derivation step preserves the positions of the f-subterms of t'. Furthermore, there must be a basic f-subterm $\tau = t'|(q \cdot 1)$ (resp. $\tau = t'|(q \cdot 2)$) for $q \in Struct_f(t')$ and p is below or at $(q \cdot 1)$ (resp. $(q \cdot 2)$). This basic f-subterm τ corresponds to $c[l\sigma_l]$ and rewrites to $c[r\sigma_l]$ that occurs as a basic f-subterm of s at the same position, i.e. at $(q \cdot 1)$ (resp. $(q \cdot 2)$). All other basic f-subterms of t' remain unchanged. This means, $\hbar[t',s](q_1) = q_1$ and $s|q_1$ preserves the same form as $t'|q_1$. Only if $t'|p$ occurs inside one L_i, we merely need to replace this with L'_i obtained by replacing just one element in L_i with its corresponding pendant (that is equal modulo $l = r$).

When $(l,r) \in \{(e_l,e_r),(e_r,e_l)\}$ and $p \notin Struct_f(t')$, the position p must be below the position of a basic f-subterm of t' and we have a similar situation as in case $(l,r) \notin \{(e_l,e_r),(e_r,e_l)\}$.

When $(l,r) \in \{(e_l,e_r),(e_r,e_l)\}$ and $p \in Struct_f(t')$, we distinguish the following cases:

- If $p = q_1 \bullet (\overbrace{2 \cdot \ldots 2}^{n_p\times})$ for $0 \leq n_p$, the equation application yields according to $\hbar[t', \{e_l,e_r\}]$ to one of the following cases:
  1. Two elements in L_i switch the positions: Here, we replace $L_i = L_i^l \# [\tau'_1, \tau'_2] \# L_i^r$ with $L'_i = L_i^l \# [\tau'_2, \tau'_1] \# L_i^r$.
  2. Some τ_{i_j} switches the position with the last element in L_{j-1}: Here, we replace $L_{j-1} = L_{j-1}^l \# [\tau']$ with $L'_{j-1} = L_{j-1}^l$ and replace L_j with $L'_j = [\tau'] \# L_j$.
  3. Some τ_{i_j} switches the position with the first element in L_j: Here, we replace $L_j = [\tau'] \# L_j^r$ with $L'_j = L_j^r$ and replace L_{j-1} with $L'_{j-1} = L_{j-1} \# [\tau']$, when $j - 1 \neq 0$ (case 3-a). In case $j - 1 = 0$, we preserve L_0 as an empty list, if l^1 is not a variable (case 3-b), and we replace L_0 with $L'_0 = L_0 \# [\tau']$, otherwise (case 3-a).
  4. Some τ_{i_j} switches the position with $\tau_{i_{j+1}}$: This is only possible when L_j is empty.

  Except of case (3-b), where τ' is pulled out, the whole structure of $t'|q_1$ is preserved in all other cases. Thus, we set $q'_1 = q_1 \bullet (2)$ in case (3-b), and $q'_1 = q_1$ in the other cases. Clearly, $s|q'_1$ has the required form (corresponding to the form of $t'|(q_1 \bullet (2))$ in case (3-b) and of $t'|q_1$ in the other cases).

- If there is $q_2 \in Struct_f(t')$ with $q_1 = q_2 \bullet (2)$ and $p = q_2$, the equation application yields according to $\hbar[t, \{e_l,e_r\}]$ to one of the following cases:
  1. The sub-term $\tau'_p = t'|(p \bullet (1))$ switches the position with the first element in L_0: Here, we replace $L_0 = [\tau'] \# L_0^r$ with $L'_0 = [\tau', \tau'_p] \# L_0^r$.
  2. The sub-term $\tau'_p = t'|(p \bullet (1))$ switches the position with τ_{i_1}, which holds when L_0 is empty: Here, we replace L_1 with $L'_1 = [\tau'_p] \# L_1$.

  In both cases, we set $q'_1 = p$. Clearly, $s|q'_1$ has the required form, where L_0 is non-empty in case (1) and empty in case (2).

- In the complementary case, i.e. $p \neq q_1 \bullet (\overbrace{2 \cdot \ldots 2}^{n_p \times})$ for $0 \leq n_p$ and $q_1 = q_2 \bullet (2)$ implies $p \neq q_2$, the replacement of $t'|p = l\sigma_l$ with $s'|p = r\sigma_l$ does not change the order of the elements in $L_0\#[\tau_{i_1}]\#\ldots\#L_{n_1}\#[\tau^1]$. This allows us to set $q'_1 = q_1$ and obtain the required form for $s|q'_1$. □

We want to justify that the above given joining requirements are sufficient to deal with the overlaps in cases (3-b) and (4). For that purpose, we assume arbitrary f-terms t_1, t_2, s_1 and s_2 satisfying $t_1 =_A t_2$, $t_1 \to^{p_1}_{(l_1,r_1)} s_1$ and $t_2 \to^{p_2}_{(l_2,r_2)} s_2$ for $p_1 \in Struct_f(t_1)$, to explain how s_1 and s_2 can be joined in cases (3-b) and (4).

Using $t_1|p_1 = l_1\sigma_1$ and $t_1 =_A t_2$, theorem 21 yields q_1 with $\hbar[t_1,t_2](p_1) = q_1 \bullet (\overbrace{2 \cdot \ldots 2}^{n \times})$ for $0 \leq n$ and $t_2|q_1$ is of the form $\overline{f}(L_0, f(\tau_{i_1}, \overline{f}(L_1, \ldots, \overline{f}(L_{n_1-1}, f(\tau_{i_{n_1}}, \overline{f}(L_{n_1}, \tau^1)))\ldots))))$, where τ^1 is not an f-term, $\tau_j =_A l^1_j\sigma_1$ for $1 \leq j \leq n_1$, and

1. either l^1 is a variable and $l^1\sigma_1 =_A \overline{f}(L^1, \tau^1)$ for a list L^1 consisting of elements from $L_0\#\ldots\#L_{n_1}$
2. or l^1 is not a variable, $l^1\sigma_1 =_A \tau^1$, and L_0 is empty.

For the positions p_2 (with $t_2|p_2 = l_2\sigma_2$) and q_1 of t_2, we distinguish the following cases:

- p_2 is below q_1 and $l_2\sigma_2$ occurs in $L_0\#\ldots\#L_{n_1}$: Either there is no overlap between l_1 and l_2 or the overlap of l_2 on l_1 is at or below the position of l^1, provided l^1 is a variable.
- p_2 is below q_1 and $l_2\sigma_2$ occurs in τ_{i_j} for $1 \leq j \leq n_1$: Here, we distinguish two cases:
  1. $l^1_{i_j}$ is a f-term and $l_2\sigma_2$ occurs at a position $\mu' \in Struct_f(l^1_{i_j})$: This corresponds to case (3-b), as we have $l'_1 =_A l_1$, $\mu \in Struct_f(l'_1) \setminus \{\epsilon\}$, $(l'_1|\mu)\sigma_1 =_A l_2\sigma_2$, $s_1 = t_1[r_1\sigma_1]_{p_1}$ and $s_2 = t_2[r_2\sigma_2]_{p_2} =_A t_1[l'_1\sigma_1[r_2\sigma_2]_\mu]_{p_1}$. Here, joining requirement (1) is sufficient to join $l'_1\sigma_1[r_2\sigma_2]_\mu$ and $r_1\sigma_1$. Hence, s_1 and s_2 are join-able, too.
  2. Otherwise, $l_2\sigma_2$ occurs below a position $\mu \in Struct_f(l_1)$: The overlap of l_2 on l_1 is at a variable position or corresponds to case (2-b-ii) in Sec. 4.3.2.2.
- p_2 is below q_1 and $l_2\sigma_2$ occurs in τ^1: Here, the overlap of l_2 on l_1 is at a variable position or corresponds to case (2-b-ii) in Sec. 4.3.2.2.
- p_2 equals or is below q_1 and p_2 is of the form $q_1 \bullet (\overbrace{2 \cdot \ldots 2}^{n_2 \times})$: Regarding the overlapping between the rewrite scopes of l_1 and l_2, we need to take the following issues into consideration.
  1. In case $l^1\sigma_1$ is an f-term, we distinguish whether there is f-subterms of $l^1\sigma_1$ out of the rewrite scope of l_2. This is for instance the case when $l^1\sigma_1 =_A \overline{f}(L_0, \tau^1)$, $L_0 = [\tau'_1]$, $L_1\#\ldots\#L_{n_1} = \epsilon$ and $l_2\sigma_2 = f(\tau_{i_1}, \ldots, f(\tau_{i_{n_1}}, \tau^1)\ldots)$. Here, $\sigma_1 = \sigma'_1 \cup \{f(\tau'_1, \tau^1)/l^1\}$, $s_1 = t_1[r_1\sigma_1]_{p_1}$ and $s_2 =_A t_1[f(\tau'_1, r_2\sigma_2)]_{p_1}$. Since $l_1 \to r_1$ must satisfy joining requirement (3), we have $r_1\sigma_1 =_A f(\tau'_1, r_1(\sigma'_1 \cup \{\tau^1/l^1\}))$. Hence, joining $r_1\sigma$ and $r_2\sigma$ for all $\sigma \in \mathcal{U}_A(l_1, l_2)$ permits to join s_1 and s_2, in this case.

     W.l.o.g., we assume in the following cases that there is no f-subterms of $l^1\sigma_1$ out of the rewrite scope of l_2.

2. In case $l^2\sigma_2$ is an f-term, we distinguish whether there is f-subterms of $l^2\sigma_2$ out of the rewrite scope of l_1. This is for instance the case when $l^2\sigma_2 =_A \overline{f}(L_{n_1},\tau^1)$, $L_{n_1} = [\tau'_2]$, $L_0\#\ldots\#L_{n_1-1} = \epsilon$, $l_2\sigma_2 = f(\tau_{i_1},\ldots,f(\tau_{i_{n_1}},f(\tau'_2,\tau^1))\ldots)$ and $l_1\sigma_1 =_A f(\tau_{i_1},\ldots,f(\tau_{i_{n_1}},\tau^1)\ldots)$. Here, $\sigma_2 = \sigma'_2 \cup \{f(\tau'_2,\tau^1)/l^2\}$, $s_2 = t_2[r_2\sigma_2]_{p_2}$ and $s_1 =_A t_2[f(\tau'_2,r_1\sigma_1)]_{p_2}$. Since $l_2 \to r_2$ must satisfy joining requirement (3), we have $r_2\sigma_2 =_A f(\tau'_2,r_2(\sigma'_2 \cup \{\tau^1/l^2\}))$. Hence, joining $r_1\sigma$ and $r_2\sigma$ for all $\sigma \in \mathcal{U}_A(l_1,l_2)$ permits to join s_1 and s_2, in this case.
   W.l.o.g., we assume in the following cases that there is no f-subterms of $l^2\sigma_2$ out of the rewrite scope of l_1.
3. We distinguish whether there is f-subterms of $l^1_j\sigma_1$ out of the rewrite scope of l_2. This is for instance the case when $l_2\sigma_2 =_A \overline{f}(L,f(\tau_{i_2},\ldots,f(\tau_{i_{n_1}},\tau^1)\ldots))$ and $l_1\sigma_1 =_A f(\tau_{i_1},\ldots,f(\tau_{i_{n_1}},\tau^1)\ldots)$.
   If L is empty, $l_2\sigma_2$ occurs at a position $\mu \in Struct_f(l'_1) \setminus \{\epsilon\}$ with $l'_1 =_A l_1$, and $(l'_1|\mu)\sigma_1 =_A l_2\sigma_2$. This corresponds to a case (3-b) that can be handled based on joining requirement (1).
   Otherwise, L must include f-subterms of l_2. For instance, $L = [l^2_1\sigma_2]$ and $t_2|q_1 = f(\tau_{i_1},\overline{f}(L,f(\tau_{i_2},\ldots,f(\tau_{i_{n_1}},\tau^1)\ldots)))$ correspond to $t_1|\hbar[t_2,t_1](q_1) = f(l^2_1\sigma_2,l_1\sigma_1)$ and $t_2|q_1 =_A f(l^1_{i_1}\sigma_1,l_2\sigma_2)$. This yields $s_1 = t_1[f(l^2_1\sigma_2,r_1\sigma_1)]_{\hbar[t_2,t_1](q_1)}$ and $s_2 =_A t_2[f(l^1_{i_1}\sigma_1,r_2\sigma_2)]_{q_1}$. Here, s_1 and s_2 can be clearly joined based on joining requirement (2).
4. In the complementary case to (1), (2) and (3), the overlap of l_1 on l_2 is at a position $\mu \in Struct_f(l_2)$. This corresponds either to case (2) in Sec. 4.3.2.1 or to case (3-b) where l_1 and l_2 are switched.

- q_1 is below p_2: Here, the overlap of l_1 on l_2 is at a position $\mu \in Struct_f(l_2) \setminus \{\epsilon\}$ or at or below the position of l^2, provided l^2 is a variable.
- q_1 and p_2 are non-overlapping: Here, there is no overlap between l_1 and l_2.

4.4.2.1.3 Joining Requirements for Case (2-b-ii): Finally, we provide the joining requirements that are sufficient to deal with the overlaps of case (2-b-ii) in Sec. 4.3.2.2. For a g-term l_1 and a f-term $l_1|\mu$ given by a subterm position μ below the position of a basic g-subterm in l_1 where $\hat{\mu} = \mu$, there can be a nested overlap of $l_2 \to r_2$ on $l_1 \to r_1$ at μ only if l_2 is a f-term. Since $\hat{\mu}$ is below the position of a basic g-subterm in l_1, we just need to adapt joining requirement (1): For all $l' =_A l_1|\mu$, $\mu' \in Struct_f(l')$ and $\sigma \in \mathcal{U}_A(l'|\mu',l_2)$, ensure that $r_1\sigma$ and $(l_1[l'[r_2]_{\mu'}]_\mu)\sigma$ are join-able.

4.4.2.2 AC Function Symbols:

Let $f \in \Sigma\langle 2\rangle$ be a AC function symbol. Focusing on the overlaps introduced in Sec. 4.3.2.1, we consider arbitrary rewrite rules $l_1 \to r_1$ and $l_2 \to r_2$ where l_1 and l_2 are f-terms.

- It is easy to prove that the equational overlaps can be covered by extending l_1 and l_2 to $f(x_2,l_1)$ and respectively $f(x_1,l_2)$ for two new variables x_1 and x_2. The AC property allows us to gather all basic f-subterms of l_2 and l_1 that cause the equational overlap into x_2 and respectively x_1.
  To deal with case (4) in Sec. 4.3.2.1, we require that $f(x_2,r_1)\sigma$ and $f(x_1,r_2)\sigma$ are join-able for all $\sigma \in \mathcal{U}_A(f(x_2,l_1),f(x_1,l_2))$.
- Similarly, the overlaps of l_2 on l_1 according to case (3-b) in Sec. 4.3.2.1 can be covered by extending l_2 to $f(x_1,l_2)$ for a new variable x_1. The AC property allows us to gather all basic f-subterms of l_1 outside of the rewrite scope of l_2 into x_1.

Accordingly, we require that $r_1\sigma$ and $f(x_1,r_2)\sigma$ are join-able for all unifiers σ in $\mathcal{U}_A(l_1,f(x_1,l_2))$.

Regarding the overlaps introduced in Sec. 4.3.2.2, we assume that f is the head symbol of l_2 and of a nested sub-term $l_1|\mu$ of l_1 for μ below the position of a basic g-subterm in l_1.

- To cover overlaps of l_2 on l_1 at position μ, we require that $r_1\sigma$ and $(l_1[r_2]_\mu)\sigma$ are join-able for all $\sigma \in \mathcal{U}_A(l_1|\mu,l_2)$.
- To cover overlaps of l_2 on l_1 at position $\mu' \in Struct_f(l_1|\mu) \setminus \{\epsilon\}$, we require that $r_1\sigma$ and $(l_1[f(x_1,r_2)]_\mu)\sigma$ are join-able for all $\sigma \in \mathcal{U}_A(l_1|\mu,f(x_1,l_2))$, where x_1 is a new variable.

4.5 PACE Algebra

We describe how we obtained a complete rewriting relation R/A in case of the PACE algebra.

Recall the set of equations $Eq = \{\mathbf{e_1},\mathbf{e_2},\mathbf{e_3},\mathbf{e_4},\mathbf{e_5}\}$, which we introduced in Sec. 2.3.2:

$$\begin{aligned}
\mathbf{e_1}: &\quad dec(x,enc(x,y)) = x\\
\mathbf{e_2}: &\quad dh(dh(x,y),z) = dh(dh(x,z),y)\\
\mathbf{e_3}: &\quad fst(pair(x,y)) = x\\
\mathbf{e_4}: &\quad snd(pair(x,y)) = y\\
\mathbf{e_5}: &\quad enc(x,dec(x,y)) = y
\end{aligned}$$

Except of $\mathbf{e_2}$, i.e. a permutative equation to permute subterms of dh-terms at their subterm positions $1\cdot 2$ and 2, the rest are cancellation equations. We have thus $A = \{\mathbf{e_2}\}$ and $R_0 = \{r_1,r_2,r_3,r_4\}$, where

$$\begin{aligned}
r_1: &\quad dec(x,enc(x,y)) \to x\\
r_2: &\quad enc(x,dec(x,y)) \to y\\
r_3: &\quad fst(pair(x,y)) \to x\\
r_4: &\quad snd(pair(x,y)) \to y
\end{aligned}$$

Since dh does not occur in R_0, we need only syntactic unification in the completion process. Here, we consider merely two critical pairs of terms:

1. In the superposition of r_1 on r_2 at subterm position 2 we unify $dec(v_1,enc(v_1,v_2))$ and $dec(v_3,v_4)$ and obtain the substitution $\sigma = [v_4 \mapsto enc(v_1,v_2), v_3 \mapsto v_1]$. This yields $r_2\sigma = enc(v_1,v_2)$ and $enc(v_1,r_1\sigma) = enc(v_1,v_2)$, which are obviously join-able.
2. In the superposition of r_2 on r_1 at subterm position 2 we unify $enc(v_3,dec(v_3,v_4))$ and $enc(v_1,v_2)$ and obtain the substitution $\sigma = [v_2 \mapsto dec(v_3,v_4), v_1 \mapsto v_3]$. This yields $r_1\sigma = dec(v_3,v_4)$ and $dec(v_3,r_2\sigma) = dec(v_3,v_4)$, which are obviously join-able.

Consequently, $R/A = R_0/A$ is local confluent. The induced reduction ordering implies that it is terminating and thus complete.

4.6 TC-AMP Algebra

We describe how we obtained a complete rewriting relation R/A in case of the TC-AMP algebra.

Recall the set of equations $Eq = \{\mathbf{e_3}, \mathbf{e_4}, \mathbf{e_6}, \mathbf{e_7}, \mathbf{e_8}, \mathbf{e_9}, \mathbf{e_{10}}, \mathbf{e_{11}}, \mathbf{e_{12}}, \mathbf{e_{13}}\}$, which we introduced in Sec. 2.4.2:

$$\begin{aligned}
\mathbf{e_3}: &\quad fst(pair(x,y)) = x\\
\mathbf{e_4}: &\quad snd(pair(x,y)) = y\\
\mathbf{e_6}: &\quad \oplus(x,y) = \oplus(y,x)\\
\mathbf{e_7}: &\quad \oplus(x,\oplus(y,z)) = \oplus(\oplus(x,y),z)\\
\mathbf{e_8}: &\quad \oplus(x,\ominus(x)) = \infty\\
\mathbf{e_9}: &\quad \oplus(x,\infty) = x\\
\mathbf{e_{10}}: &\quad *(x,*(y,z)) = *(y,*(x,z))\\
\mathbf{e_{11}}: &\quad *(x,*(inv(x),y)) = y\\
\mathbf{e_{12}}: &\quad inv(inv(x)) = x\\
\mathbf{e_{13}}: &\quad *(x,\oplus(y,z)) = \oplus(*(x,y),*(x,z))
\end{aligned}$$

The equations in $A_\oplus = \{\mathbf{e_6}, \mathbf{e_7}\}$ corresponds to an AC theory for $\oplus$. In addition we obtain a second permutative theory $A_* = \{\mathbf{e_{10}}\}$, which allows us to permute subterms of $*$-terms at the subterm positions $2 \cdot 1$ and 1. Hence, $A = A_\oplus \uplus A_*$. Except of $\mathbf{e_{13}}$, the rest in $Eq \setminus A$ are cancellation equations.

We chose to orient $\mathbf{e_{13}}$ from left to right. So, R_0 consists of the following rewrite rules:

$$\begin{aligned}
r_1: &\quad fst(pair(x,y)) \to x\\
r_2: &\quad snd(pair(x,y)) \to y\\
r_3: &\quad \oplus(x,\ominus(x)) \to \infty\\
r_4: &\quad \oplus(x,\infty) \to x\\
r_5: &\quad *(x,*(inv(x),y)) \to y\\
r_6: &\quad inv(inv(x)) \to x\\
r_7: &\quad *(x,\oplus(y,z)) \to \oplus(*(x,y),*(x,z))
\end{aligned}$$

Before we applied our completion procedure in Sec. 4.4, we added three rules (r_8, r_9 and r_{10}) known from rewriting systems for abelian groups, [39].

$$\begin{aligned}
r_8: &\quad \ominus(\infty) \to \infty\\
r_9: &\quad \ominus(\ominus(x)) \to x\\
r_{10}: &\quad \ominus(\oplus(x,y)) \to \oplus(\ominus(x),\ominus(y))
\end{aligned}$$

Afterwards, we checked the local confluence of the rules r_1–r_{10} (see appendix A). According to the obtained joining requirements, it is necessary to add the following rules

$$\begin{aligned}
r_{11}: &\quad *(x,\infty) \to \infty\\
r_{12}: &\quad *(x,\ominus(y)) \to \ominus(*(x,y))
\end{aligned}$$

The final set R of rewrite rules and the set A provide us with a confluent (modular) rewriting relation R/A. The induced reduction ordering implies that it is terminating and thus complete.

Chapter 5

Message Objects and Operations

As discussed in Sec. 1.1.3.2, it is unclear whether the core structure of a given message algebra like in TC-AMP can be modeled as a freely generated data type permitting appropriate definitions of recursive functions as required for inductive protocol verification. Instead of that, we propose another approach: For a given message algebra specification $((\Sigma, At), E)$,

1. we first gain a complete (modular) rewriting system R/A equivalent to the equations in E, to focus on the rewriting-based model $\mathcal{R}_A = (C, \{f^{\mathcal{C}} \mid f \in \Sigma \setminus \Sigma\langle 0\rangle\})$, as defined in Sec. 3.3 by
   - $C = \{[m]_A \downarrow_{R_A} \mid [m]_A \in Mes_0(\Sigma, At)/=_A\}$ and
   - $f^{\mathcal{C}}([m_0]_A \downarrow_{R_A}, \ldots, [m_{n-1}]_A \downarrow_{R_A}) = [f(m_0, \ldots, m_{n-1})]_A \downarrow_{R_A}$,
2. and then we provide appropriate axioms about the objects (reduced A equivalence classes $[m]_A \downarrow_{R_A}$) and the basic operations ($f^{\mathcal{C}}$).

In analogy to the freely generated data type in [75], our axioms

- provide us with a notion of message parts (substructures) and a classification of messages as a basis for
  - *complete case distinctions*,
  - *well founded orders* and
  - *inequalities*,
- and cover the effects of rewrite-based operations $f^{\mathcal{C}}$ defining in particular *constructor-type applications*.

Based on these axioms, we are then able to define (*recursive*) functions and relations that permit us to reason (*inductively*) on the derivability of messages (see Chap. 6) and on the indistinguishability of message sequences (see Chap. 10).

In Chap. 4 we described how to obtain a complete rewriting system R/A from a given set E of equations, where A is a partition of permutative theories on single function symbols. In this chapter, we describe how to obtain the required axioms for the corresponding rewriting-based model $\mathcal{R}_A = (C, \{f^{\mathcal{C}} \mid f \in \Sigma \setminus \Sigma\langle 0\rangle\})$.

In contrast to a freely generated data type, there is a *stronger* correspondence between the given message algebra $((\Sigma, At), E)$ and the resulting axioms:

- First, R/A is gained from E uniformly by a sound algorithm (see Chap. 4).
- Second, the axioms providing the case distinction, the substructure relation and the inequalities are according to uniform schemata, relying on nice properties of permutative theories A on single function symbols (see Sec. 5.1).

- Third, the axioms covering the effects of rewrite-based operations $f^{\mathcal{C}}$ and defining *constructor-type applications* (see Sec. 5.2) are tightly linked to the rules in R and the equations in A by

  1. refining the rules in R (into operation schemata), taking into account the interference during rewriting between applied rules and with the equations in A,
  2. justifying (on the meta-level) that the resulting operation schemata are sound (justified by rewriting situations) and complete (covering all rewriting situations),
  3. and by transforming uniformly the operation schemata to axioms.

The general axiomatization approach is applied to the message algebras of PACE (Sec. 5.3) and TC-AMP (Sec. 5.4).

5.1 Message Structures

Our objects (the messages) are reduced A classes $[m]\downarrow_{R_A} = \{m_1,\ldots,m_{N(m)}\}$. This means, we have

1. $m_i =_A m_j$ for all $m_i, m_j \in \{m_1,\ldots,m_{N(m)}\}$, and
2. m_i is not R-reducible for all $m_i \in \{m_1,\ldots,m_{N(m)}\}$.

Furthermore, we suppose (as in case of the message algebras for PACE and TC-AMP) that A is a partition of independent permutative theories $A_{f_1} \uplus \ldots \uplus A_{f_N}$ on function symbols $f_1,\ldots,f_N$. This allows us to base the axiomatization of the message structures on nice properties, which we introduce in Sec. 5.1.1, together with the axiomatized and defined notions. The corresponding axioms and definitions (on message structures) are described in Sec. 5.1.2.

5.1.1 Reduced A Classes

As A is a partition of independent permutative theories $A_{f_1} \uplus \ldots \uplus A_{f_N}$ on single function symbols $f_1,\ldots,f_N$, we may base the axiomatization of the message structures on the following property.

Theorem 22. *Let $A = A_{f_1} \uplus \ldots \uplus A_{f_N}$ be a partition of independent permutative theories on* different *function symbols $f_1,\ldots,f_N$ and let m and m' be two ground message terms with $m =_A m'$. Then we have*

1. *m and m' have the same top-symbol,*
2. *and for each function symbol f the number of occurrences of f in m and m' coincides.*

Proof: The proof is by induction on the term-depth of m. The base case is trivial, since m belongs to $\Sigma\langle 0\rangle \cup At$ and m' must be syntactically equal with m.

In the step case, m must be an f-term for $f \in \Sigma\langle n\rangle$ and $n > 0$. According to theorem 18, we have a bijection $\hbar[m,m']$ from the subterm positions of m to the subterm positions of m' that satisfies properties (1)–(4) (see theorem 18). We want to apply property (4) (by induction on $N(m)$) of $\hbar[m,m']$ focusing on the positions $p_1^{\overline{f}},\ldots,p_{N(m)}^{\overline{f}}$ of the basic f-subterms of m. By definition,

1. the top-symbol of $m|p_i^{\overline{f}}$ differs from f, and

2. m as well as all non-basic f-subterms of m have f as top-symbol.

It is clear that $p_1^{\overline{f}}, \ldots, p_{N(m)}^{\overline{f}}$ are non-overlapping. Furthermore, we have $\hat{p}_1^{\overline{f}} = p_1^{\overline{f}}, \ldots, \hat{p}_{N(m)}^{\overline{f}} = p_{N(m)}^{\overline{f}}$. Using property (4) of $\hbar[m, m']$ (by induction on $N(m)$), we obtain thus

(a) $m|q =_A m'|\hbar[m, m'](q)$ for all $q \in \{p_1^{\overline{f}}, \ldots, p_{N(m)}^{\overline{f}}\}$, and

(b) $m[.,\ldots,.]_{p_1^{\overline{f}},\ldots,p_{N(m)}^{\overline{f}}} =_A m'[.,\ldots,.]_{\hbar[m,m'](p_1^{\overline{f}}),\ldots,\hbar[m,m'](p_{N(m)}^{\overline{f}})}$

Note that $m[.,\ldots,.]_{p_1^{\overline{f}},\ldots,p_{N(m)}^{\overline{f}}}$ is composed merely from the symbol f and from variables as place-holders for the subterms $m|p_i^{\overline{f}}$. This implies that $m'[.,\ldots,.]_{\hbar[m,m'](p_1^{\overline{f}}),\ldots,\hbar[m,m'](p_{N(m)}^{\overline{f}})}$ and hence m' has the same top-symbol f as m. Furthermore, the derivation of m' from m using exclusively permutative equations on single function symbols allows us to show that f occurs in $m[.,\ldots,.]_{p_1^{\overline{f}},\ldots,p_{N(m)}^{\overline{f}}}$ as many as in $m'[.,\ldots,.]_{\hbar[m,m'](p_1^{\overline{f}}),\ldots,\hbar[m,m'](p_{N(m)}^{\overline{f}})}$. By the induction hypothesis, we have additionally that every function symbol in $m|q$ occurs as many as in $m'|\hbar[m, m'](q)$ for all $q \in \{p_1^{\overline{f}}, \ldots, p_{N(m)}^{\overline{f}}\}$. This allows us to conclude that every function symbol in m occurs as many as in m'. □

Based on property (1) of theorem 22, we characterize the messages as follows:

Definition 23 (Types and Structures of Messages)**.**

1. We say a message $[m] \in C$ to be *atomic*, if $[m] = \{m\}$ for some $m \in \Sigma\langle 0\rangle \cup At$. *Otherwise*, we call $[m]$ a *composed message*.

2. We distinguish the composed messages according to their top-symbols. The messages that have $f \in \Sigma \setminus \Sigma\langle 0\rangle$ as top-symbol are called *f-objects*.

3. For f-objects $[m]$ and $f(m_0, \ldots, m_{n-1}) \in [m]$, (it is obvious that) $[m]$ can be composed by a constructor-type application of f to $[m_0], \ldots, [m_{n-1}]$. We represent this relation between $[m]$ and $([m_0], \ldots, [m_{n-1}])$ using predicates obj^f (one, for each $f \in \Sigma \setminus \Sigma\langle 0\rangle$). Semantically, $[\![obj^f]\!]$ consists of all tuples $([m], [m_0], \ldots, [m_{n-1}]) \in C^{n+1}$ satisfying $f(m_0, \ldots, m_{n-1}) \in [m]$.

4. For an f-object $[m]$ and $f(m_0, \ldots, m_{n-1}) \in [m]$, we call $[m_0], \ldots, [m_{n-1}]$ *f-parts* of $[m]$.

In certain contexts, f-objects are referred to by *f-messages*.

For f-objects, we clearly may speak about their *substructures*, which consist of the corresponding f-parts and of the substructures belonging to the composed f-parts.

Property (2) of theorem 22 allows us to define a measure $|.|$ on messages by counting the occurrences of function symbols whose arities are greater than 0.

Definition 24 ($|.|$ on Messages)**.** We define the *measure* $|.| : C \to \mathbb{N}$ for arbitrary $[m] \in C$ by:

1. $|[m]| = 0$, if $[m] = \{m\}$ for $m \in \Sigma\langle 0\rangle \cup At$.

2. $|[m]| = 1 + |[m_0]| + \ldots + |[m_{n-1}]|$, if $f(m_0, \ldots, m_{n-1}) \in [m]$, $f \in \Sigma\langle n\rangle$ and $n > 0$.

It is clear that $|.|$ is well defined, due to property (2) of theorem 22.
The measure $|.|$ induces a well-founded order $<_{|.|}$ on C by

$$[m] <_{|.|} [m'] \Leftrightarrow |[m]| < |[m']|.$$

This so-called size order $<_{|.|}$ includes inherently the structural order, which permits us to apply *structural induction*. It is furthermore the basis for the definition of other orders if required by other induction schemes.

5.1.2 Message Structures and the $|.|$-Measure

In this section, we describe how to reflect the notions in Def. 23 and 24 on the object level.
We start with the axiomatization of the complete case distinction.

Axiom Schema 25 (Complete Case Distinction):
Let $\Sigma\langle 0\rangle = \{c_1,\ldots,c_{N'}\}$, $\Sigma\langle 1\rangle = \{f_{1,1},\ldots,f_{1,n_1}\}$, ..., $\Sigma\langle N\rangle = \{f_{N,1},\ldots,f_{N,n_N}\}$, and let the sets $\Sigma\langle n\rangle$ be empty for all $n > N$. Then, the axiom about the complete case distinction is obtained according to following schema:

$$\begin{aligned}
&m \in At \dot{\vee} (m = c_1 \dot{\vee} \ldots \dot{\vee} m = c_{N'})\\
&\dot{\vee}(\exists m_0 : obj^{f_{1,1}}(m,m_0) \dot{\vee} \ldots \dot{\vee} obj^{f_{1,n_1}}(m,m_0))\\
&\qquad\qquad\vdots\\
&\dot{\vee}(\exists m_0,\ldots,m_{N-1} : obj^{f_{N,1}}(m,m_0,\ldots,m_{N-1}) \dot{\vee} \ldots \dot{\vee} obj^{f_{N,n_N}}(m,m_0,\ldots,m_{N-1}))
\end{aligned}$$

In this axiom schema, we abbreviate the mutual exclusive case distinction for the types of m using "$\dot{\vee}$" for *exclusive or*. This implies in particular that messages of different types, e.g., $\oplus(a,b)$ and $\ominus(a)$, are not equals.

Next, we provide the definition schema for the $|.|$-measure.

Definition Schema 26 ($|.|$):
Let $\Sigma\langle 0\rangle = \{c_1,\ldots,c_{N'}\}$, $\Sigma\langle 1\rangle = \{f_{1,1},\ldots,f_{1,n_1}\}$, ..., $\Sigma\langle N\rangle = \{f_{N,1},\ldots,f_{N,n_N}\}$, and let the sets $\Sigma\langle n\rangle$ be empty for all $n > N$. Then, the measure $|.|$ is defined according to the following schema:

1. $|m| = 0 \Leftrightarrow (m \in At \vee (m = c_1 \vee \ldots \vee m = c_{N'}))$

2. $$\begin{aligned}
&|m| > 0 \Leftrightarrow\\
&((\exists m_0 : (obj^{f_{1,1}}(m,m_0) \vee \ldots \vee obj^{f_{1,n_1}}(m,m_0)) \wedge |m| = |m_0| + 1) \vee\\
&\qquad\qquad\vdots\\
&\vee (\exists m_0,\ldots,m_{N-1} : (obj^{f_{N,1}}(m,m_0,\ldots,m_{N-1}) \vee \ldots \vee obj^{f_{N,n_N}}(m,m_0,\ldots,m_{N-1}))\\
&\qquad\qquad \wedge |m| = 1 + |m_0| + \ldots + |m_{N-1}|))
\end{aligned}$$

The next axioms are mainly about (the relations represented by) the predicates obj^f, with $f \in \Sigma\langle n\rangle$ and $n > 0$. For given arbitrary messages $m, m_0,\ldots,m_{n-1} \in C$, we have $obj^f(m,m_0,\ldots,m_{n-1})$ iff

1. no rewrite rule $f(t_0,\ldots,t_{n-1}) \to r$ is applicable to $f(m_0,\ldots,m_{n-1})$, and

2. $m =_A f(m_0,\ldots,m_{n-1})$.

We postpone the axiomatization of condition (1) to Sec. 5.2, where we formalize in particular when $obj^f(f(m_0,\ldots,m_{n-1}),m_0,\ldots,m_{n-1})$ holds. The axiomatization of (2), i.e. whether two f-objects are equal modulo A, is according to the following schema:

Axiom Schema 27 (Structure of Composed Messages):
Let $f \in \Sigma \setminus \Sigma\langle 0\rangle$. The axioms about the (dis-)equalities of f-objects are obtained as follows:

1. When f does not occur in A, i.e. f is independent from the permutative equations in A, then we have:
$(obj^f(m,m_0,\ldots,m_{n-1}) \wedge obj^f(m,m'_0,\ldots,m'_{n-1})) \Leftrightarrow (m_0 = m'_0 \wedge \ldots \wedge m_{n-1} = m'_{n-1})$

2. When f is AC, then we have:
$(obj^f(m,m_0,m_1) \wedge obj^f(m,m'_0,m'_1)) \Leftrightarrow$
$((m_0 = m'_0 \wedge m_1 = m'_1) \vee (m_0 = m'_1 \wedge m_1 = m'_0)$
$\vee (\exists m_2 : obj^f(m_0,m'_0,m_2) \wedge obj^f(m'_1,m_2,m_1))$
$\vee (\exists m_2 : obj^f(m_0,m'_1,m_2) \wedge obj^f(m'_0,m_2,m_1))$
$\vee (\exists m_2 : obj^f(m_1,m'_0,m_2) \wedge obj^f(m'_1,m_2,m_0))$
$\vee (\exists m_2 : obj^f(m_1,m'_1,m_2) \wedge obj^f(m'_0,m_2,m_0))$
$\vee (\exists m_2,m_3,m_4,m_5 : obj^f(m_0,m_2,m_3) \wedge obj^f(m_1,m_4,m_5)$
$\wedge obj^f(m'_0,m_2,m_4) \wedge obj^f(m'_1,m_3,m_5)))$

3. When $f \in \Sigma\langle 2\rangle$ and $A_f = \{f(f(x,y),z) = f(f(x,z),y)\}$, then we have:
$(obj^f(m,m_0,m_1) \wedge obj^f(m,m'_0,m'_1)) \Leftrightarrow$
$((m_0 = m'_0 \wedge m_1 = m'_1) \vee (\exists m_2 : obj^f(m_0,m_2,m'_1) \wedge obj^f(m'_0,m_2,m_1)))$

4. When $f \in \Sigma\langle 2\rangle$ and $A_f = \{f(x,f(y,z)) = f(y,f(x,z))\}$, then we have:
$(obj^f(m,m_0,m_1) \wedge obj^f(m,m'_0,m'_1)) \Leftrightarrow$
$((m_0 = m'_0 \wedge m_1 = m'_1) \vee (\exists m_2 : obj^f(m_1,m'_0,m_2) \wedge obj^f(m'_1,m_0,m_2)))$

Axioms 2–4 are adaptations of the decomposition rules (in Sec. 4.4.1) used for unification modulo A as part of the completion algorithm.

Note that these axioms are necessary to prove inequalities between different f-objects, e.g., $\oplus(a,b) \neq \oplus(a,a)$ for atomic messages a and b.

5.2 Basic Operations

We want to axiomatize when $obj^f(f(m_0,\ldots,m_{n-1}),m_0,\ldots,m_{n-1})$ holds. For that purpose, we need to know all complementary cases, i.e. all cases where $f(m_0,\ldots,m_{n-1})$ is reducible by an outermost rewrite step followed in certain cases by further rewriting steps. Recall, a reducible $f(m_0,\ldots,m_{n-1})$ yields after finitely many rewrite steps a message in C, which is not necessary a f-object. The axiomatization of all these cases corresponds to an axiomatization of the basic operations $\{f^{\mathcal{C}} : C^n \to C \mid f \in \Sigma \setminus \Sigma\langle 0\rangle\}$, where

$$f^{\mathcal{C}}([m_0],\ldots,[m_{n-1}]) = [f(m_0,\ldots,m_{n-1})]\downarrow_{R_A}.$$

It is clear that the resulting axioms have to be complete and sound, in the sense that *all* possible reduction cases are covered and that the specified structures are *justified* by (applied) corresponding rewrite-rules. Identifying and characterizing all possible reduction cases necessitates to analyze the interactions between different rewrite rules and permutations by A: Focusing on the question on how $[f(m_0,\ldots,m_{n-1})]$ can be rewritten to $[f(m_0,\ldots,m_{n-1})]\downarrow_{R_A}$ for arbitrary $[m_0],\ldots,[m_{n-1}] \in C$, the *general* purpose is to identify *schemata* relating substructures of $[f(m_0,\ldots,m_{n-1})]\downarrow_{R_A}$ with those from $[m_0],\ldots$, and

$[m_{n-1}]$. These so-called *operation schemata* extend rewrite rules with (additional) structural information, to refine the relation between $[f(m_0,\ldots,m_{n-1})]\downarrow_{R_A}$ and the arguments $[m_0],\ldots,[m_{n-1}]$ into the different cases. Operation schemata are expressed with a relatively simple and intuitive vocabulary, which we introduce in Sec. 5.2.1. They permit us to provide for given R/A a list of operation schemata, which covers all rewrite-specific cases of the basic operations $f^{\mathcal{C}}$. This intermediate step is accompanied with a thorough argumentation that the provided list of operation-schemata is *sound* and *complete* (see above and Sec. 5.3.2.1, 5.4.2 and 5.4.3). After that, the operation schemata are transformed *uniformly* to the axioms on the basic operations, as described in Sec. 5.2.2.

5.2.1 Operation Schemata

We want to introduce the notion of "operation schemata" with the help of a running example from the TC-AMP algebra. It is clear that the rewrite-specific cases of the basic operation $\oplus^{\mathcal{C}}$ heavily depend on the following rewrite-rules:

$$r_3:\ \oplus(x,\ominus(x))\to\infty\ \text{ or }\ r_4:\ \oplus(x,\infty)\to x.$$

Analyzing the interaction of these rules with the remaining rewrite rules and with the A equations in Sec. 4.6 (commutativity and associativity of $\oplus$), we obtain the following definition of $\oplus^{\mathcal{C}}$:

$$\oplus^{\mathcal{C}}([m_0],[m_1])=\begin{cases}
[m_0] & : \ [m_1]=\{\infty\}\\
[m_1] & : \ [m_0]=\{\infty\}\\
\{\infty\} & : \ \ominus(m_0)\in[m_1]\\
\{\infty\} & : \ \ominus(m_1)\in[m_0]\\
[m_3] & : \ \oplus(m_2,m_3)\in[m_1],\ \ominus(m_0)\in[m_2]\\
[m_3] & : \ \ominus(m_2)\in[m_0],\ \oplus(m_2,m_3)\in[m_1]\\
[m_3] & : \ \oplus(m_2,m_3)\in[m_0],\ \ominus(m_1)\in[m_2]\\
[m_3] & : \ \oplus(m_2,m_3)\in[m_0],\ \ominus(m_2)\in[m_1]\\
\oplus^{\mathcal{C}}([m_3],[m_5]) & : \ \oplus(m_2,m_3)\in[m_1],\ \ominus(m_4)\in[m_2],\\
 & \quad \oplus(m_4,m_5)\in[m_0]\\
\oplus^{\mathcal{C}}([m_3],[m_5]) & : \ \oplus(m_2,m_3)\in[m_0],\ \ominus(m_4)\in[m_2],\\
 & \quad \oplus(m_4,m_5)\in[m_1]\\
[\oplus(m_0,m_1)] & : \ \text{otherwise}
\end{cases}$$

We distinguish eleven cases:

1. The first case corresponds to an immediate application of r_4, without use of equations from A.

2. The second case corresponds to an application of r_4 after applying the commutativity of $\oplus$.

3. The third case corresponds to an immediate application of r_3, without use of equations from A.

4. The fourth case corresponds to an application of r_3 after applying the commutativity of $\oplus$.

5. The fifth case corresponds to an application of r_3, prepared by suitable permutations based on A, and ended up by an application of r_4.

6. The sixth case is similar to the fifth case, but the simplified $\ominus$-object is the first argument instead of being a part of the second argument.

7. The seventh case is identical to the fifth case, if we switch the first and the second argument.
8. The eighth case is identical to the sixth case, if we switch the first and the second argument.
9. The ninth case corresponds to an application of r_3, prepared by suitable permutations based on A, and followed by an application of r_4. Contrary to the previous cases, there can be further applications of r_3 and r_4, which yields to the recursive invocation of $\oplus^{\mathcal{C}}$.
10. The tenth case is identical to the ninth case, if we switch the first and the second argument.
11. The eleventh case corresponds to a constructor-type application of $\oplus^{\mathcal{C}}$, where neither r_3 nor r_4 is applicable.

Basically, these eleven cases can be expressed in a more compact manner as an *operation schema*, where the rewrite-specific single cases are rather represented as applications of rewrite rules, incorporating at the same time the main notions used in our axiomatization. This can be seen in the following operation schema, representing the above definition of $\oplus^{\mathcal{C}}$:

$$
\begin{aligned}
r'_{4,1}: &\quad \oplus^{\mathcal{C}}(m_0,\infty) \rightsquigarrow m_0;\\
r'_{4,2}: &\quad \oplus^{\mathcal{C}}(\infty,m_1) \rightsquigarrow m_1;\\
r'_{3,1}: &\quad \oplus^{\mathcal{C}}(m_0,\ominus^{obj}(m_0)) \rightsquigarrow \infty;\\
r'_{3,2}: &\quad \oplus^{\mathcal{C}}(\ominus^{obj}(m_1),m_1) \rightsquigarrow \infty;\\
r'_{3,3}: &\quad \oplus^{\mathcal{C}}(m_0,\oplus^{obj}(\ominus^{obj}(m_0),m_2)) \rightsquigarrow m_2;\\
r'_{3,4}: &\quad \oplus^{\mathcal{C}}(\ominus^{obj}(m_2),\oplus^{obj}(m_2,m_3)) \rightsquigarrow m_3;\\
r'_{3,5}: &\quad \oplus^{\mathcal{C}}(\oplus^{obj}(\ominus^{obj}(m_1),m_2),m_1) \rightsquigarrow m_2;\\
r'_{3,6}: &\quad \oplus^{\mathcal{C}}(\oplus^{obj}(m_2,m_3),\ominus^{obj}(m_2)) \rightsquigarrow m_3;\\
r'_{3,7}: &\quad \oplus^{\mathcal{C}}(\oplus^{obj}(m_2,m_3),\oplus^{obj}(\ominus^{obj}(m_2),m_4)) \rightsquigarrow \oplus^{\mathcal{C}}(m_3,m_4);\\
r'_{3,8}: &\quad \oplus^{\mathcal{C}}(\oplus^{obj}(\ominus^{obj}(m_2),m_3),\oplus^{obj}(m_2,m_4)) \rightsquigarrow \oplus^{\mathcal{C}}(m_3,m_4);
\end{aligned}
$$

Note that the labels on the left-side are used just to link each case with the rewrite-rule applied first in that case. The mentioned cases cover all outermost rewriting cases starting with the use of the given rewrite rules r_3 and r_4. Consequently, the given operation schema is complete. It is also sound, because the specified structures in each case can be justified by the corresponding application of r_3 or r_4.

Basically, the operation schema represents explicitly only the rewrite-specific cases (the first ten cases in the above definition of $\oplus^{\mathcal{C}}$). The case where no rewrite-rule is applicable (the eleventh case in the above definition of $\oplus^{\mathcal{C}}$) is only implicit.

In operation schemata, the structures of arguments and results are expressed using constants from $\Sigma\langle 0\rangle$ and terms with head-symbols h^{obj} for $h \in \Sigma \setminus \Sigma\langle 0\rangle$, which are obviously place-holders for h-objects.

Definition 28 (Operation Schema)**.** Let $f \in \Sigma \setminus \Sigma\langle 0\rangle$ and let R include rewrite rules of the form $f(t_0,\ldots,t_{n-1}) \to r$. An *operation schema* for the rewrite-specific cases of the basic operation $f^{\mathcal{C}}$ is a sequence of the form:

$$
\begin{aligned}
&f^{\mathcal{C}}(\alpha_{1,0},\ldots,\alpha_{1,n-1}) \rightsquigarrow \beta_1;\\
&\vdots\\
&f^{\mathcal{C}}(\alpha_{n_f,0},\ldots,\alpha_{n_f,n-1}) \rightsquigarrow \beta_{n_f};
\end{aligned}
$$

$\alpha_{i,j}$-s on the left-side are called α-terms and β_k-s on the right-side β-terms, where

- the *basic* α-terms and β-terms belong to $\Sigma\langle 0\rangle \cup \mathcal{MV}$, for a finite set $\mathcal{MV}$ of message variables that includes $m_0, \ldots, m_{n-1}$,
- the *composed* α-terms are of the form $h^{obj}(\alpha_0, \ldots, \alpha_{n'-1})$, where $h \in \Sigma\langle n'\rangle$ with $n' > 0$ and where $\alpha_0, \ldots, \alpha_{n'-1}$ are α-terms,
- and where the *composed* β-terms are of two possible forms $h^{obj}(\beta'_0, \ldots, \beta'_{n'-1})$ or $h^{\mathcal{C}}(\beta'_0, \ldots, \beta'_{n'-1})$, where $h \in \Sigma\langle n'\rangle$ with $n' > 0$ and where $\beta'_0, \ldots, \beta'_{n'-1}$ are β-terms.

Every message variable in a β-term must occur in the corresponding left-side.

An operation schema is a sequence of distinguished rewrite-specific cases in a basic operation. In each distinguished case, the arguments of $f^{\mathcal{C}}$ (in the left-side) are clearly in normal-form and can be associated with certain structures as provided by the corresponding rewrite-rule. Contrarily on the right-side, certain results could depend from further (*recursive*) basic operations, which can be expressed through appropriate composed β-terms. For that reason, the axioms on basic operations that we obtain systematically from operation schemata could be seen sometimes as *recursive definitions*. The *base cases*, which we call the *recursion-free* cases, correspond to the constructor-type case and to the cases where the structure of the result is *definitely given* (after *a single rewrite step* in the rewriting-based model) from the structure of the arguments. In the *recursion cases*, *several alternatives exist* for the structure of the result relative to the structure of the arguments (according to the possible *subsequent rewriting steps* in the rewriting-based model).

For the (operation schemata) axioms that include recursion we will justify that the recursive cases use smaller arguments than in the defined basic operation. In the above operation schema for the basic operation $\oplus^{\mathcal{C}}$, it is obvious that the arguments of $\oplus^{\mathcal{C}}$ in the right-side of cases $r'_{3,7}$ and $r'_{3,8}$ are substructures of the respective arguments of $\oplus^{\mathcal{C}}$ in the left-side.

5.2.2 Axioms about Basic Operations

In this section, we exemplify how operation schemata are transformed *uniformly* to the axioms on the basic operations. Given an arbitrary operation schema as in Def. 28, the corresponding axiom is obtained according to the following schema:

$$
\begin{aligned}
&((\exists m_n, \ldots, m_{n'_1} : \Phi_{\alpha,1} \wedge \Psi_{\beta,1}) \vee \\
&\qquad\qquad \vdots \vee \\
&(\exists m_n, \ldots, m_{n'_{n_f}} : \Phi_{\alpha,n_f} \wedge \Psi_{\beta,n_f}) \vee \\
&((\forall m_n, \ldots, m_{n'_1} : \neg\Phi_{\alpha,1}) \wedge \ldots \wedge (\forall m_n, \ldots, m_{n'_{n_f}} : \neg\Phi_{\alpha,n_f}) \\
&\quad \wedge obj^f(f(m_0, \ldots, m_{n-1}), m_0, \ldots, m_{n-1})),
\end{aligned}
$$

where $\Phi_{\alpha,i}$ and respectively $\Psi_{\beta,i}$ (for $1 \leq i \leq n_f$) are structural propositions according to the left-side and the right-side, respectively, of the i-th case in the operation schema. In our running example, the first case ($r'_{4,1}$) provides us with the structural proposition

$$(m_1 = \infty \wedge \oplus(m_0, m_1) = m_0),$$

where $m_1 = \infty$ corresponds to $\Phi_{\alpha,1}$ and the rest to $\Psi_{\beta,1}$; According to the third case ($r'_{3,1}$) we obtain the structural proposition

$$(obj^{\ominus}(m_1, m_0) \wedge \oplus(m_0, m_1) = \infty),$$

where $obj^{\ominus}(m_1, m_0)$ corresponds to $\Phi_{\alpha,3}$ and the rest to $\Psi_{\beta,3}$; The fifth case ($r'_{3,3}$) provides us with the structural proposition

$$(\exists m_2, m_3 : obj^{\oplus}(m_1, m_3, m_2) \wedge obj^{\ominus}(m_3, m_0) \wedge \oplus(m_0, m_1) = m_2),$$

where $obj^{\oplus}(m_1, m_3, m_2) \wedge obj^{\ominus}(m_3, m_0)$ corresponds to $\Phi_{\alpha,5}$ and the rest to $\Psi_{\beta,5}$; According to the sixth case ($r'_{3,4}$) we obtain the structural proposition

$$(\exists m_2, m_3 : obj^{\ominus}(m_0, m_2) \wedge obj^{\oplus}(m_1, m_2, m_3) \wedge \oplus(m_0, m_1) = m_3),$$

where $obj^{\ominus}(m_0, m_2) \wedge obj^{\oplus}(m_1, m_2, m_3)$ corresponds to $\Phi_{\alpha,6}$ and the rest to $\Psi_{\beta,6}$; The ninth case ($r'_{3,7}$) provides us with the structural proposition

$$\begin{aligned}(\exists m_2, m_3, m_4, m_5 : obj^{\oplus}(m_0, m_2, m_3) \wedge obj^{\oplus}(m_1, m_5, m_4) \wedge obj^{\ominus}(m_5, m_2) \wedge \\ \oplus(m_0, m_1) = \oplus(m_3, m_4)),\end{aligned}$$

where $obj^{\oplus}(m_0, m_2, m_3) \wedge obj^{\oplus}(m_1, m_5, m_4) \wedge obj^{\ominus}(m_5, m_2)$ corresponds to $\Phi_{\alpha,9}$ and the rest to $\Psi_{\beta,9}$. Altogether, the operation schema for $\oplus^{\mathcal{C}}$ is transformed to the following axiom:

Axiom$^{\text{VSE}}$ 29 (Axiom about obj$^{\oplus}$):

$$\begin{aligned}&((m_1 = \infty \wedge \oplus(m_0, m_1) = m_0) \vee \\ &(m_0 = \infty \wedge \oplus(m_0, m_1) = m_1) \vee \\ &(obj^{\ominus}(m_1, m_0) \wedge \oplus(m_0, m_1) = \infty) \vee \\ &(obj^{\ominus}(m_0, m_1) \wedge \oplus(m_0, m_1) = \infty) \vee \\ &(\exists m_2, m_3 : obj^{\oplus}(m_1, m_3, m_2) \wedge obj^{\ominus}(m_3, m_0) \wedge \oplus(m_0, m_1) = m_2) \vee \\ &(\exists m_2, m_3 : obj^{\ominus}(m_0, m_2) \wedge obj^{\oplus}(m_1, m_2, m_3) \wedge \oplus(m_0, m_1) = m_3) \vee \\ &(\exists m_2, m_3 : obj^{\oplus}(m_0, m_3, m_2) \wedge obj^{\ominus}(m_3, m_1) \wedge \oplus(m_0, m_1) = m_2) \vee \\ &(\exists m_2, m_3 : obj^{\oplus}(m_0, m_2, m_3) \wedge obj^{\ominus}(m_1, m_2) \wedge \oplus(m_0, m_1) = m_3) \vee \\ &(\exists m_2, m_3, m_4, m_5 : obj^{\oplus}(m_0, m_2, m_3) \wedge obj^{\oplus}(m_1, m_5, m_4) \wedge obj^{\ominus}(m_5, m_2) \wedge \\ &\quad \oplus(m_0, m_1) = \oplus(m_3, m_4)) \vee \\ &(\exists m_2, m_3, m_4, m_5 : obj^{\oplus}(m_0, m_5, m_3) \wedge obj^{\oplus}(m_1, m_2, m_4) \wedge obj^{\ominus}(m_5, m_2) \wedge \\ &\quad \oplus(m_0, m_1) = \oplus(m_3, m_4)) \vee \\ &((m_1 \neq \infty) \wedge (m_0 \neq \infty) \wedge (\neg obj^{\ominus}(m_1, m_0)) \wedge (\neg obj^{\ominus}(m_0, m_1)) \wedge \\ &(\forall m_2, m_3 : \neg obj^{\oplus}(m_1, m_3, m_2) \vee \neg obj^{\ominus}(m_3, m_0)) \wedge \\ &(\forall m_2, m_3 : \neg obj^{\ominus}(m_0, m_2) \vee \neg obj^{\oplus}(m_1, m_2, m_3)) \wedge \\ &(\forall m_2, m_3 : \neg obj^{\oplus}(m_0, m_3, m_2) \vee \neg obj^{\ominus}(m_3, m_1)) \wedge \\ &(\forall m_2, m_3 : \neg obj^{\oplus}(m_0, m_2, m_3) \vee \neg obj^{\ominus}(m_1, m_2)) \wedge \\ &(\forall m_2, m_3, m_4, m_5 : \neg obj^{\oplus}(m_0, m_2, m_3) \vee \neg obj^{\oplus}(m_1, m_5, m_4) \vee \neg obj^{\ominus}(m_5, m_2)) \wedge \\ &(\forall m_2, m_3, m_4, m_5 : \neg obj^{\oplus}(m_0, m_5, m_3) \vee \neg obj^{\oplus}(m_1, m_2, m_4) \vee \neg obj^{\ominus}(m_5, m_2)) \wedge \\ &obj^{\oplus}(\oplus(m_0, m_1), m_0, m_1))).\end{aligned}$$

Note that such axioms could be replaced with sets of equivalent axioms that formalize separately the different rewrite-specific cases using explicit predicates f_r'-s (one for each rewriting case) and that include (abbreviated) definitions of obj^f-s using the newly introduced predicates. In our running example, the predicate for the second case would be $\oplus\_r'_{4,2}$ and the corresponding formalization would be

$$\oplus\_r'_{4,2}(m, m_0, m_1) \Leftrightarrow (m_0 = \infty \wedge m = \oplus(m_0, m_1) = m_1);$$

In the same line, the tenth case would be formalized by

$$\oplus\_r'_{3,8}(m,m_0,m_1) \Leftrightarrow (\exists m_2,m_3,m_4,m_5 : obj^{\oplus}(m_0,m_5,m_3) \wedge obj^{\oplus}(m_1,m_2,m_4) \wedge obj^{\ominus}(m_5,m_2) \wedge m = \oplus(m_0,m_1) = \oplus(m_3,m_4)).$$

Using the introduced predicates, we would obtain the following axiom defining $obj^{\oplus}$:

$$\begin{aligned} obj^{\oplus}(m,m_0,m_1) \Leftrightarrow (&\neg \oplus\_r'_{4,1}(m,m_0,m_1) \wedge \neg \oplus\_r'_{4,2}(m,m_0,m_1) \wedge \neg \oplus\_r'_{3,1}(m,m_0,m_1) \wedge \\ &\neg \oplus\_r'_{3,2}(m,m_0,m_1) \wedge \neg \oplus\_r'_{3,3}(m,m_0,m_1) \wedge \neg \oplus\_r'_{3,4}(m,m_0,m_1) \wedge \\ &\neg \oplus\_r'_{3,5}(m,m_0,m_1) \wedge \neg \oplus\_r'_{3,6}(m,m_0,m_1) \wedge \neg \oplus\_r'_{3,7}(m,m_0,m_1) \wedge \\ &\neg \oplus\_r'_{3,8}(m,m_0,m_1) \wedge m = \oplus(m_0,m_1)). \end{aligned}$$

Altogether, the first axiom about $obj^{\oplus}$ would be replaced with eleven axioms introducing ten new predicates. But, we prefer to do without predicates f_r'-s (as in the first described approach) and thus employ one axiom for each basic operation $f^{\mathcal{C}}$, which serves as definition for obj^f.

5.3 Axiomatization of the PACE Algebra

We start with the axioms about the message structures and the definition of the $|.|$-measure (in Sec. 5.3.1), and then we discuss the axiomatization of the basic operations (in Sec. 5.3.2).

5.3.1 Message Structures and the $|.|$-Measure

The axioms and the definitions in this section are obtained as described in Sec. 5.1.

We start with the axiom generated according to schema 25.

Axiom$^{\mathbf{VSE}}$ 30 (Complete Case Distinction; PACE):

$$\begin{aligned} &m \in At \dot{\vee} (\exists m_0 : obj^{fst}(m,m_0) \dot{\vee} obj^{snd}(m,m_0)) \dot{\vee} \\ &(\exists m_0,m_1 : obj^{pair}(m,m_0,m_1) \dot{\vee} obj^{enc}(m,m_0,m_1) \dot{\vee} obj^{dh}(m,m_0,m_1) \dot{\vee} \\ &\quad obj^{gen}(m,m_0,m_1) \dot{\vee} obj^{dec}(m,m_0,m_1) \dot{\vee} obj^{mac}(m,m_0,m_1)) \end{aligned}$$

Note that the atomic messages in At are as well distinguished into different kinds. The corresponding specification will be discussed later in Chap. 8.

Next, we provide the definition of $|.|$ following schema 26.

Definition$^{\mathbf{VSE}}$ 31 ($|.|$; PACE):

1. $|m| = 0 \Leftrightarrow m \in At$

2. $$\begin{aligned} &|m| > 0 \Leftrightarrow \\ &((\exists m_0 : (obj^{fst}(m,m_0) \vee obj^{snd}(m,m_0)) \wedge |m| = |m_0| + 1) \vee \\ &\ (\exists m_0,m_1 : (obj^{pair}(m,m_0,m_1) \vee obj^{enc}(m,m_0,m_1) \vee obj^{dh}(m,m_0,m_1) \vee \\ &\qquad obj^{gen}(m,m_0,m_1) \vee obj^{dec}(m,m_0,m_1) \vee obj^{mac}(m,m_0,m_1)) \\ &\qquad \wedge |m| = 1 + |m_0| + |m_1|)) \end{aligned}$$

The next axioms are about the structure of *enc*- and *dh*-objects and are obtained according to the axiom schemata 27-(1) and -(3).

Axiom$^{\mathbf{VSE}}$ 32 (Structure of enc- and dh-objects):

1. $(obj^{enc}(m,m_0,m_1) \wedge obj^{enc}(m,m'_0,m'_1)) \Leftrightarrow (m_0 = m'_0 \wedge m_1 = m'_1)$

2. $(obj^{dh}(m,m_0,m_1) \wedge obj^{dh}(m,m'_0,m'_1)) \Leftrightarrow$
   $((m_0 = m'_0 \wedge m_1 = m'_1) \vee (\exists m_2 : obj^{dh}(m_0,m_2,m'_1) \wedge obj^{dh}(m'_0,m_2,m_1)))$

For *dec*-, *pair*-, fst-, *snd*-, *mac*-, and *gen*-objects, the corresponding f-parts are identified with corresponding axioms according to axiom schema 27-(1). This results in six further axioms similar to 32-(1).

5.3.2 Axioms about Basic Operations

First, we want to provide a *complete* list of operation schemata that cover the rewrite-specific cases of the basic operations.

5.3.2.1 Operation Schemata

All rewrite rules in PACE are cancellations that result in a substructure of one argument. Furthermore, they can be applied without taking the permutative property of *dh* into consideration, as *dh* does not occur in any rewrite rule. In such a situation, we know that $f^{\mathcal{C}}([m_0],\ldots,[m_{n-1}])$ yields either $[f(m_0,\ldots,m_{n-1})]$ or a substructure $[m'_i]$ of a $[m_i]$. According to the fact that all $t \in [m_i]$ and their sub-terms are irreducible, m'_i results just by one rewriting step. For that reason, each rewrite rule provides a single case in the corresponding operation schema. In case of the rewrite rules r_1–r_4 in Sec. 4.5, we obtain (through obvious transformations) the corresponding operation schemata:

1. r'_1 : $dec^{\mathcal{C}}(m_0,enc^{obj}(m_0,m_2)) \rightsquigarrow m_2$
2. r'_2 : $enc^{\mathcal{C}}(m_0,dec^{obj}(m_0,m_2)) \rightsquigarrow m_2$
3. r'_3 : $fst^{\mathcal{C}}(pair^{obj}(m_1,m_2)) \rightsquigarrow m_1$
4. r'_4 : $snd^{\mathcal{C}}(pair^{obj}(m_1,m_2)) \rightsquigarrow m_2$

It is clear that the given operation schemata are sound, as each includes a single rewrite-specific case linked directly to a corresponding rewrite rule.

5.3.2.2 Resulting Axioms

The axioms about the basic operations are obtained by a simple transformation of the operation schemata, as described in Sec. 5.2.2. The following axioms result from the transformation of r'_1–r'_4:

Axiom$^{\mathbf{VSE}}$ 33 (Axioms about obj$^{\mathbf{dec}}$, obj$^{\mathbf{enc}}$, obj$^{\mathbf{fst}}$ and obj$^{\mathbf{snd}}$):

1. $((\exists m_2 : obj^{enc}(m_1,m_0,m_2) \wedge dec(m_0,m_1) = m_2) \vee$
   $((\forall m_2 : \neg obj^{enc}(m_1,m_0,m_2)) \wedge obj^{dec}(dec(m_0,m_1),m_0,m_1)))$

2. $((\exists m_2 : obj^{dec}(m_1, m_0, m_2) \wedge enc(m_0, m_1) = m_2) \vee$
$((\forall m_2 : \neg obj^{dec}(m_1, m_0, m_2)) \wedge obj^{enc}(enc(m_0, m_1), m_0, m_1)))$

3. $((\exists m_1, m_2 : obj^{pair}(m_0, m_1, m_2) \wedge fst(m_0) = m_1) \vee$
$((\forall m_1, m_2 : \neg obj^{pair}(m_0, m_1, m_2)) \wedge obj^{fst}(fst(m_0), m_0)))$

4. $((\exists m_1, m_2 : obj^{pair}(m_0, m_1, m_2) \wedge snd(m_0) = m_2) \vee$
$((\forall m_1, m_2 : \neg obj^{pair}(m_0, m_1, m_2)) \wedge obj^{snd}(snd(m_0), m_0)))$

For basic operations without rewrite-specific cases the axioms are straightforward. For instance, we obtain in case of the basic operation $dh^{\mathcal{C}}$ the following axiom:

Axiom$^{\text{VSE}}$ 34 (Axiom about obj$^{\text{dh}}$):

$$obj^{dh}(dh(m_0, m_1), m_0, m_1).$$

Further three similar axioms are about obj^{pair}, obj^{mac} and obj^{gen}.

5.4 Axiomatization of the TC-AMP Algebra

We start with the axioms about the message structures and the definition of the $|.|$-measure (in Sec. 5.4.1), then we identify and discuss the operation schemata (in Sec. 5.4.2) and justify (in Sec. 5.4.3) the conjectured structures of results in certain cases of the identified operation schemata. Finally, we provide (in Sec. 5.4.4) the axioms (about the basic operations) resulting by the systematic transformation of the operation schemata.

5.4.1 Message Structures and the $|.|$-Measure

The axioms and the definitions in this section are obtained as described in Sec. 5.1.

We start with the axiom generated according to schema 25.

Axiom$^{\text{VSE}}$ 35 (Complete Case Distinction; TC-AMP):

$$m \in At \dot{\vee} m = \infty \dot{\vee} (\exists m_0 : obj^{fst}(m, m_0) \dot{\vee} obj^{snd}(m, m_0) \dot{\vee} obj^{inv}(m, m_0) \dot{\vee} obj^{\ominus}(m, m_0))$$
$$\dot{\vee} (\exists m_0, m_1 : obj^{pair}(m, m_0, m_1) \dot{\vee} obj^{\oplus}(m, m_0, m_1) \dot{\vee} obj^{*}(m, m_0, m_1))$$
$$\dot{\vee} (\exists m_0, m_1, m_2 : obj^{h_1}(m, m_0, m_1, m_2) \dot{\vee} obj^{h_2}(m, m_0, m_1, m_2))$$

Note that the atomic messages in At are as well distinguished into different kinds. The corresponding specification will be discussed later in Chap. 9.

Next, we provide the definition of $|.|$ following schema 26.

Definition$^{\text{VSE}}$ 36 ($|.|$; TC-AMP):

1. $|m| = 0 \Leftrightarrow (m \in At \vee m = \infty)$

2. $|m| > 0 \Leftrightarrow$
$((\exists m_0 : (obj^{fst}(m,m_0) \vee obj^{snd}(m,m_0) \vee obj^{inv}(m,m_0) \vee obj^{\ominus}(m,m_0))$
$\wedge |m| = |m_0| + 1) \vee$
$(\exists m_0, m_1 : (obj^{pair}(m,m_0,m_1) \vee obj^{\oplus}(m,m_0,m_1) \vee obj^{*}(m,m_0,m_1))$
$\wedge |m| = 1 + |m_0| + |m_1|) \vee$
$(\exists m_0, m_1, m_2 : (obj^{h_1}(m,m_0,m_1,m_2) \vee obj^{h_2}(m,m_0,m_1,m_2))$
$\wedge |m| = 1 + |m_0| + |m_1| + |m_2|))$

The next axioms are about the structure of *inv*-, $\oplus$- and $*$-objects and are obtained according to the axiom schemata 27-(1),-(2) and -(4).

Axiom[VSE] 37 (Structure of inv-, $\oplus$- and $*$-objects):

1. $(obj^{inv}(m,m_0) \wedge obj^{inv}(m,m_0')) \Leftrightarrow (m_0 = m_0')$

2. $(obj^{\oplus}(m,m_0,m_1) \wedge obj^{\oplus}(m,m_0',m_1')) \Leftrightarrow$
$((m_0 = m_0' \wedge m_1 = m_1') \vee (m_0 = m_1' \wedge m_1 = m_0')$
$\vee (\exists m_2 : obj^{\oplus}(m_0,m_0',m_2) \wedge obj^{\oplus}(m_1',m_2,m_1))$
$\vee (\exists m_2 : obj^{\oplus}(m_0,m_1',m_2) \wedge obj^{\oplus}(m_0',m_2,m_1))$
$\vee (\exists m_2 : obj^{\oplus}(m_1,m_0',m_2) \wedge obj^{\oplus}(m_1',m_2,m_0))$
$\vee (\exists m_2 : obj^{\oplus}(m_1,m_1',m_2) \wedge obj^{\oplus}(m_0',m_2,m_0))$
$\vee (\exists m_2, m_3, m_4, m_5 : obj^{\oplus}(m_0,m_2,m_3) \wedge obj^{\oplus}(m_1,m_4,m_5)$
$\wedge obj^{\oplus}(m_0',m_2,m_4) \wedge obj^{\oplus}(m_1',m_3,m_5)))$

3. $(obj^{*}(m,m_0,m_1) \wedge obj^{*}(m,m_0',m_1')) \Leftrightarrow$
$((m_0 = m_0' \wedge m_1 = m_1') \vee (\exists m_2 : obj^{*}(m_1,m_0',m_2) \wedge obj^{*}(m_1',m_0,m_2)))$

For $\ominus$-, *pair*-, *fst*-, *snd*-, h_1-, and h_2-objects, the corresponding f-parts are identified with corresponding axioms according to axiom schema 27-(1). This results in six further axioms similar to 37-(1).

5.4.2 Operation Schemata

We already gave in Sec. 5.2.1 the operation schema representing the rewrite-specific cases of $\oplus^{\mathcal{C}}$. In the following, we first provide the operation schema for $*^{\mathcal{C}}$ (in Sec. 5.4.2.1) and the operation schemata for the remaining basic operations (in Sec. 5.4.2.2).

5.4.2.1 Operation Schema for $*^{\mathcal{C}}$

We consider the case where the (first) rewrite step is carried out using

$$r_5 : *(x,*(inv(x),y)) \to y, \quad r_7 : *(x,\oplus(y,z)) \to \oplus(*(x,y),*(x,z)),$$
$$r_{11} : *(x,\infty) \to \infty \qquad \text{or} \qquad r_{12} : *(x,\ominus(y)) \to \ominus(*(x,y)).$$

Taking their interactions with other rewrite-rules and the permutative property of $*$ into account, these rewrite-rules can be mapped to the following operation schema:

$$
\begin{aligned}
r'_{5,1}: &\quad *^{\mathcal{C}}(m_0, *^{obj}(inv^{obj}(m_0), m_2)) \rightsquigarrow m_2;\\
r'_{5,2}: &\quad *^{\mathcal{C}}(inv^{obj}(m_2), *^{obj}(m_2, m_3)) \rightsquigarrow m_3;\\
r'_{7}: &\quad *^{\mathcal{C}}(m_0, \oplus^{obj}(m_2, m_3)) \rightsquigarrow \oplus^{obj}(*^{\mathcal{C}}(m_0, m_2), *^{\mathcal{C}}(m_0, m_3));\\
r'_{11}: &\quad *^{\mathcal{C}}(m_0, \infty) \rightsquigarrow \infty;\\
r'_{12}: &\quad *^{\mathcal{C}}(m_0, \ominus^{obj}(m_2)) \rightsquigarrow \ominus^{obj}(*^{\mathcal{C}}(m_0, m_2));
\end{aligned}
$$

The rewrite-rule r_5 is mapped to two cases: Either the simplified *inv*-object is a substructure of the second argument ($r'_{5,1}$) or corresponds to the first argument ($r'_{5,2}$). The remaining rewrite-rules are mapped to single cases. Note that the axiom obtained by the transformation of this operation schema includes two recursive cases, r'_7 and r'_{12}. In all three recursive applications of $*^{\mathcal{C}}$, the second argument (m_2 and m_3 in r'_7; m_2 in r'_{12}) is obviously smaller than its pendant in the defined situation ($\oplus^{obj}(m_2, m_3)$ in r'_7; $\ominus^{obj}(m_2)$ in r'_{12}).

The mentioned cases cover all outermost rewriting cases starting with the use of the given rewrite rules. Consequently, the given operation schema is complete. It is also sound, because the specified structures in each case can be justified by the corresponding application of the mentioned rewrite rule and the conjectured structures of the results in r'_7 and r'_{12} can be justified as described in Sec. 5.4.3.

5.4.2.2 Operation Schemata for fst, snd, inv and $\ominus$

We consider the case where the (first) rewrite step is carried out using

$$
\begin{aligned}
&r_1: fst(pair(x,y)) \to x, \quad && r_2: snd(pair(x,y)) \to y,\\
&r_6: inv(inv(x)) \to x, && r_8: \ominus(\infty) \to \infty,\\
&r_9: \ominus(\ominus(x)) \to x \quad \text{or} && r_{10}: \ominus(\oplus(x,y)) \to \oplus(\ominus(x), \ominus(y)).
\end{aligned}
$$

Except of r_{10}, using one of these rules to rewrite m for an arbitrary $[m] \in C$ yields directly (without further rewrite steps) a message in C. In case of r_{10}, successive rewrite steps could be necessary. Accordingly, these rewrite-rules are mapped to the following operation schemata:

1. $r'_1: \quad fst^{\mathcal{C}}(pair^{obj}(m_1, m_2)) \rightsquigarrow m_1;$

2. $r'_2: \quad snd^{\mathcal{C}}(pair^{obj}(m_1, m_2)) \rightsquigarrow m_2;$

3. $r'_6: \quad inv^{\mathcal{C}}(inv^{obj}(m_1)) \rightsquigarrow m_1;$

4. $$
\begin{aligned}
r'_8: &\quad \ominus^{\mathcal{C}}(\infty) \rightsquigarrow \infty;\\
r'_9: &\quad \ominus^{\mathcal{C}}(\ominus^{obj}(m_1)) \rightsquigarrow m_1;\\
r'_{10}: &\quad \ominus^{\mathcal{C}}(\oplus^{obj}(m_1, m_2)) \rightsquigarrow \oplus^{obj}(\ominus^{\mathcal{C}}(m_1), \ominus^{\mathcal{C}}(m_2));
\end{aligned}
$$

Note that the axiom obtained by the transformation of the operation schema for $\ominus^{\mathcal{C}}$ includes one recursive case, r'_{10}. In both recursive applications of $\ominus^{\mathcal{C}}$, the argument (m_1 and respectively m_2) is obviously smaller than its pendant in the defined situation ($\oplus^{obj}(m_1, m_2)$).

The cases in the given operation schemata cover obviously all outermost rewriting cases starting with the use of the associated rewrite rules. Consequently, the given operation schemata are complete. They are also sound, because the specified structures in each case can be justified by the corresponding application of the associated rewrite rule and the conjectured structure of the result in r'_{10} can be justified as described in Sec. 5.4.3.

5.4.3 Justification of Result Structures in the Operation Schemata

In this section, we justify the conjectured structures of the results in the provided operation schemata.

First, we focus on case r'_{12} and want to justify that the result is indeed a $\ominus$-object with $*(m_0,m_2)$ as $\ominus$-part. This means, the application of $*$ to arbitrary irreducible t_0 and $\ominus(t_2)$ must yield a $\ominus$-term $\ominus(*(t_0,t_2))$ that is irreducible at the top-position. Referring to the operation schema of $\ominus^{\mathcal{C}}$, $\ominus(*(t_0,t_2))$ is reducible at the top-position only after reducing $*(t_0,t_2)$ to ∞ (cp. case r'_8), a $\ominus$-term $\ominus(t_3)$ (cp. case r'_9) or to a $\oplus$-term t_3 (cp. case r'_{10}).

1. According to the operation schema of $*^{\mathcal{C}}$, $*(t_0,t_2)$ reduces to ∞ only if $t_2 = \infty$. This does not fit in with the assumption that $\ominus(t_2)$, i.e. $\ominus(\infty)$, is irreducible.
2. According to the operation schema of $*^{\mathcal{C}}$, $*(t_0,t_2)$ reduces to a $\ominus$-term $\ominus(t_3)$ only if t_2 is a $\ominus$-term $\ominus(t_4)$. This does not fit in with the assumption that $\ominus(t_2)$, i.e. $\ominus(\ominus(t_4))$, is irreducible.
3. According to the operation schema of $*^{\mathcal{C}}$, $*(t_0,t_2)$ reduces to a $\oplus$-term t_3 only if t_2 is a $\oplus$-term $\oplus(t_4,t_5)$. This does not fit in with the assumption that $\ominus(t_2)$, i.e. $\ominus(\oplus(t_4,t_5))$, is irreducible.

Next, we consider case r'_7 to justify that the corresponding result is indeed a $\oplus$-object with $*(m_0,m_2),*(m_0,m_3)$ as $\oplus$-parts. This means, the application of $*$ to arbitrary irreducible t_0 and $\oplus(t_2,t_3)$ must yield a $\oplus$-term $\oplus(*(t_0,t_2),*(t_0,t_3))$ that is irreducible at the top-position. Referring to the operation schema of $\oplus^{\mathcal{C}}$ (in Sec. 5.2.1), $\oplus(*(t_0,t_2),*(t_0,t_3))$ is reducible at the top-position only after reducing $*(t_0,t_2)$ and/or $*(t_0,t_3)$ transforming $\oplus(*(t_0,t_2),*(t_0,t_3))$ to $\oplus(t_{02},t_{03})$ and this $\oplus$-term possesses ∞ as a basic $\oplus$-subterm (cp. cases $r'_{4,1}$ and $r'_{4,2}$) or two basic $\oplus$-subterms $\ominus(t)$ and t (cp. cases $r'_{3,1}$–$r'_{3,8}$). W.l.o.g., we focus on the case where ∞ (resp. $\ominus(t)$) equals or occurs in t_{02}.

1. According to the operation schema of $*^{\mathcal{C}}$, $*(t_0,t_2)$ reduces to t_{02} corresponding to ∞ or to a $\oplus$-term having ∞ as a basic $\oplus$-subterm only if $t_2 = \infty$. This does not fit in with the assumption that $\oplus(t_2,t_3)$, i.e. $\oplus(\infty,t_3)$, is irreducible.
2. According to the operation schema of $*^{\mathcal{C}}$, $*(t_0,t_2)$ reduces to t_{02} corresponding to $\ominus$-term $\ominus(t)$ or to a $\oplus$-term having $\ominus(t)$ as a basic $\oplus$-subterm only in the following cases:
   (a) t_2 is a $\ominus$-term $\ominus(*(t_4,t))$ such that $*(t_0,*(t_4,t))$ reduces to t: Here, $\oplus(t_{02},t_{03})$, i.e. $\oplus(\ominus(t),t_{03})$, possesses t as a basic $\oplus$-subterm only if t_{03} equals t or is a $\oplus$-term having t as a basic $\oplus$-subterm. According to the operation schema of $*^{\mathcal{C}}$, $*(t_0,t_3)$ reduces to such t_{03} only in the following cases:
      i. t_3 is a $*$-term $*(t_5,t)$ such that the term $*(t_0,*(t_5,t))$ reduces to t: Based on $*(t_0,*(t_4,t)) = *(t_0,*(t_5,t))$, we get $t_5 = t_4$.
      ii. t_3 is a $*$-subterm of t, which corresponds to the $*$-term $*(t_0,t_3)$: Based on $*(t_0,*(t_4,t)) = *(t_0,t_3)$, we get $t_3 = *(t_4,t)$.
      iii. t_3 is a $\oplus$-term having a basic $\oplus$-subterm t'_3 that matches $*(t_5,t)$ in case (i) or that is a $*$-subterm of t as in case (ii). In both cases, we deduce that t'_3 equals $*(t_4,t)$.

      In all three cases, $\oplus(t_2,t_3)$ possesses $\ominus(*(t_4,t))$ and $*(t_4,t)$ as basic $\oplus$-subterms, which does not fit in with the assumption that $\oplus(t_2,t_3)$ is irreducible.
   (b) t_2 is a $\ominus$-term $\ominus(t_4)$ such that t corresponds to the $*$-term $*(t_0,t_4)$: Here, $\oplus(t_{02},t_{03})$, i.e. $\oplus(\ominus(t),t_{03})$, possesses t as a basic $\oplus$-subterm only if t_{03} equals t or is a $\oplus$-term having t as a basic $\oplus$-subterm. According to the operation schema of $*^{\mathcal{C}}$, $*(t_0,t_3)$ reduces to such t_{03} only in the following cases:

i. t_3 is a $*$-term $*(t_5,t)$ such that $*(t_0,*(t_5,t))$ reduces to t: Based on $*(t_0,t_4) = *(t_0,*(t_5,t))$, we get $*(t_5,t) = t_4$.

ii. t_3 is a $*$-subterm of t, which corresponds to the $*$-term $*(t_0,t_3)$: Based on $*(t_0,t_4) = *(t_0,t_3)$, we get $t_3 = t_4$.

iii. t_3 is a $\oplus$-term having a basic $\oplus$-subterm t'_3 that matches $*(t_5,t)$ in case (i) or that is a $*$-subterm of t as in case (ii). In both cases, we deduce that t'_3 equals t_4.

In all three cases, $\oplus(t_2,t_3)$ possesses $\ominus(t_4)$ and t_4 as basic $\oplus$-subterms, which does not fit in with the assumption that $\oplus(t_2,t_3)$ is irreducible.

(c) t_2 is a $\oplus$-term with t'_2 as a basic $\oplus$-subterm that matches $\ominus(*(t_4,t))$ in (a) or $\ominus(t_4)$ in (b): Here, $\oplus(t_{02},t_{03})$ possesses t as a basic $\oplus$-subterm only if t occurs in t_{02} besides $\ominus(t)$, t_{03} equals t or t_{03} is a $\oplus$-term having t as a basic $\oplus$-subterm. The first case does not fit in with the assumption that t_{02} is irreducible. The second case is handled similar to case (a) and the third case similar to case (b).

Finally, we consider case r'_{10} to justify that the result is indeed a $\oplus$-object with $\ominus(m_1),\ominus(m_2)$ as $\oplus$-parts. This means, the application of $\ominus$ to an arbitrary irreducible $\oplus(t_1,t_2)$ must yield a $\oplus$-term $\oplus(\ominus(t_1),\ominus(t_2))$ that is irreducible at the top-position. Referring to the operation schema of $\oplus^{\mathcal{C}}$, $\oplus(\ominus(t_1),\ominus(t_2))$ is reducible at the top-position only after reducing $\ominus(t_1)$ and/or $\ominus(t_2)$ transforming $\oplus(\ominus(t_1),\ominus(t_2))$ to $\oplus(t_3,t_4)$ and this $\oplus$-term possesses ∞ as a basic $\oplus$-subterm (cp. cases $r'_{4,1}$ and $r'_{4,2}$) or two basic $\oplus$-subterms $\ominus(t)$ and t (cp. cases $r'_{3,1}$–$r'_{3,8}$). W.l.o.g., we focus on the case where ∞ (resp. $\ominus(t)$) equals or occurs in t_3.

1. According to the operation schema of $\ominus^{\mathcal{C}}$, $\ominus(t_1)$ reduces to t_3 corresponding to ∞ or to a $\oplus$-term having ∞ as a basic $\oplus$-subterm only if $t_1 = \infty$. This does not fit in with the assumption that $\oplus(t_1,t_2)$, i.e. $\oplus(\infty,t_2)$, is irreducible.

2. According to the operation schema of $\ominus^{\mathcal{C}}$, $\ominus(t_1)$ reduces to t_3 corresponding to $\ominus$-term $\ominus(t)$ or to a $\oplus$-term having $\ominus(t)$ as a basic $\oplus$-subterm only in the following cases:

   (a) t_1 is a basic $\ominus$-subterm of $\ominus(t)$, which means t_1 equals t: Here, $\oplus(t_3,t_4)$, i.e. $\oplus(\ominus(t),t_4)$, possesses t as a basic $\oplus$-subterm only if t_4 equals t or is a $\oplus$-term having t as a basic $\oplus$-subterm. According to the operation schema of $\ominus^{\mathcal{C}}$, $\ominus(t_2)$ reduces to such t_4 only in the following cases:

      i. t_2 is a $\ominus$-term $\ominus(t)$.

      ii. t_2 is a $\oplus$-term having $\ominus(t)$ as a basic $\oplus$-subterm.

      In both cases, $\oplus(t_1,t_2)$ possesses $\ominus(t)$ and t as basic $\oplus$-subterms, which does not fit in with the assumption that $\oplus(t_1,t_2)$ is irreducible.

   (b) t_1 is a $\oplus$-term with t as a basic $\oplus$-subterm: Here, $\oplus(t_3,t_4)$ possesses t as a basic $\oplus$-subterm only if t occurs in t_3 besides $\ominus(t)$, t_4 equals t or t_4 is a $\oplus$-term having t as a basic $\oplus$-subterm. The first case does not fit in with the assumption that t_3 is irreducible. The second and third case are handled similar to case (a).

5.4.4 Axioms about Basic Operations

Besides axiom 29 (in Sec. 5.2.2), we obtain the following axioms by the uniform transformation of the operation schemata in Sec. 5.4.2:

Axiom^{VSE} 38 (Axioms about obj*, obj^{fst}, obj^{snd}, obj^{inv} and obj$^{\ominus}$):

1. $((\exists m_2, m_3 : obj^*(m_1, m_3, m_2) \wedge obj^{inv}(m_3, m_0) \wedge *(m_0, m_1) = m_2) \vee$
$(\exists m_2, m_3 : obj^{inv}(m_0, m_2) \wedge obj^*(m_1, m_2, m_3) \wedge *(m_0, m_1) = m_3) \vee$
$(\exists m_2, m_3 : obj^{\oplus}(m_1, m_2, m_3) \wedge obj^{\oplus}(*(m_0, m_1), *(m_0, m_2), *(m_0, m_3))) \vee$
$(m_1 = \infty \wedge *(m_0, m_1) = \infty) \vee$
$(\exists m_2 : obj^{\ominus}(m_1, m_2) \wedge obj^{\ominus}(*(m_0, m_1), *(m_0, m_2))) \vee$
$((\forall m_2, m_3 : \neg obj^*(m_1, m_3, m_2) \vee \neg obj^{inv}(m_3, m_0)) \wedge$
$(\forall m_2, m_3 : \neg obj^{inv}(m_0, m_2) \vee \neg obj^*(m_1, m_2, m_3)) \wedge$
$(\forall m_2, m_3 : \neg obj^{\oplus}(m_1, m_2, m_3)) \wedge m_1 \neq \infty \wedge$
$(\forall m_2 : \neg obj^{\ominus}(m_1, m_2)) \wedge obj^*(*(m_0, m_1), m_0, m_1)))$

2. $((\exists m_1, m_2 : obj^{pair}(m_0, m_1, m_2) \wedge fst(m_0) = m_1) \vee$
$((\forall m_1, m_2 : \neg obj^{pair}(m_0, m_1, m_2)) \wedge obj^{fst}(fst(m_0), m_0)))$

3. $((\exists m_1, m_2 : obj^{pair}(m_0, m_1, m_2) \wedge snd(m_0) = m_2) \vee$
$((\forall m_1, m_2 : \neg obj^{pair}(m_0, m_1, m_2)) \wedge obj^{snd}(snd(m_0), m_0)))$

4. $((\exists m_1 : obj^{inv}(m_0, m_1) \wedge inv(m_0) = m_1) \vee$
$((\forall m_1 : \neg obj^{inv}(m_0, m_1)) \wedge obj^{inv}(inv(m_0), m_0)))$

5. $((m_0 = \infty \wedge \ominus(m_0) = \infty) \vee$
$(\exists m_1 : obj^{\ominus}(m_0, m_1) \wedge \ominus(m_0) = m_1) \vee$
$(\exists m_1, m_2 : obj^{\oplus}(m_0, m_1, m_2) \wedge obj^{\oplus}(\ominus(m_0), \ominus(m_1), \ominus(m_2))) \vee$
$(m_0 \neq \infty \wedge (\forall m_1 : \neg obj^{\ominus}(m_0, m_1)) \wedge (\forall m_1, m_2 : \neg obj^{\oplus}(m_0, m_1, m_2)) \wedge$
$obj^{\ominus}(\ominus(m_0), m_0)))$

For basic operations without rewrite-specific cases the axioms are straightforward. For instance, we obtain in case of the basic operation $h_1^{\mathcal{C}}$ the following axiom:

Axiom^{VSE} 39 (Axiom about obj^{h_1}):

$$obj^{h_1}(h_1(m_0, m_1, m_2), m_0, m_1, m_2).$$

Further two similar axioms are about obj^{pair}, and obj^{h_2}.

5.5 Auxiliary Notions and Short-Cuts

The axiomatization of message structures and basic operations, as described in Sec. 5.1.2 and 5.2, forms the basis for further notions employed in our approach to the inductive verification of protocols.

5.5.1 Message substructures and Induction Schemes

The complete case distinction according to axiom schema 25 and the $|.|$-measure according to definition schema 26 permit us to prove properties on messages by *structural induction*.

Many properties are about recursively defined notions relative to the structure of f-objects with arbitrary many f-parts, where f belongs to the permutative function symbols. For their proofs we prefer to employ more focused induction schemes based on the so-called f-structure of messages.

In PACE, we define the *dh*-structure of messages by the following measure:

Definition[VSE] 40 ($|.|_{dh}$; PACE):

1. $|m|_{dh} = 0 \Leftrightarrow (\forall m_0, m_1 : \neg obj^{dh}(m, m_0, m_1))$

2. $|m|_{dh} > 0 \Leftrightarrow (\exists m_0, m_1 : obj^{dh}(m, m_0, m_1) \wedge |m|_{dh} = 1 + |m_0|_{dh})$

In TC-AMP, we define the $\oplus$-structure (resp. $*$-structure) of messages by the following measures:

Definition[VSE] 41 ($|.|_{\oplus}$ and $|.|_{*}$; TC-AMP):

1. $|m|_{\oplus} = 0 \Leftrightarrow (\forall m_0, m_1 : \neg obj^{\oplus}(m, m_0, m_1))$

2. $|m|_{\oplus} > 0 \Leftrightarrow (\exists m_0, m_1 : obj^{\oplus}(m, m_0, m_1) \wedge |m|_{\oplus} = 1 + |m_0|_{\oplus} + |m_1|_{\oplus})$

1. $|m|_{*} = 0 \Leftrightarrow ((\forall m_0, m_1 : \neg obj^{*}(m, m_0, m_1)) \wedge (\forall m_0 : \neg obj^{\ominus}(m, m_0) \vee |m_0|_{*} = 0))$

2. $|m|_{*} > 0 \Leftrightarrow ((\exists m_0, m_1 : obj^{*}(m, m_0, m_1) \wedge |m|_{*} = 1 + |m_1|_{*})$
$\vee\ (\exists m_0 : obj^{\ominus}(m, m_0) \vee |m|_{*} = |m_0|_{*}))$

The measures $|.|_f$ count top occurrences of "f" in given m. They implicitly identify specific message parts in m, as introduced in the following:

- For *dh*-objects m, the message parts m_0 and respectively m_1 that are identified by $obj^{dh}(m, m_0, m_1)$ are called *a left* and *a right dh*-part of m, respectively. In general, m can have arbitrary many *permutable* right *dh*-parts. We identify these with the help of the auxiliary function map_{dh}, defined for given message m and multiset ms by

$$map_{dh}(m, ms) = \begin{cases} m & : \quad ms = \emptyset \\ dh(map_{dh}(m, ms'), m') & : \quad ms = \{m'\} \uplus ms' \end{cases}$$

  For a *dh*-object m, we call m_0 and a multiset ms *the left dh*-part and respectively *the right dh*-parts of m when these satisfy

$$(\forall m_1, m_2 : \neg obj^{dh}(m_0, m_1, m_2)) \wedge m = map_{dh}(m_0, ms).$$

- For $\oplus$-objects m, we distinguish the $\oplus$-parts that are not $\oplus$-objects and call them *the basic* $\oplus$-parts of m. We also use the auxiliary function $map_{\oplus}$ for the successive application of $\oplus$, which is defined for given multiset ms of messages by

$$map_{\oplus}(ms) = \begin{cases} \infty & : \quad ms = \emptyset \\ \oplus(map_{\oplus}(ms'), m') & : \quad ms = \{m'\} \uplus ms' \end{cases}$$

- For $*$-objects m, the message parts m_0 and respectively m_1 that are identified by $obj^*(m, m_0, m_1)$ are called *a left* and *a right* $*$-part of m, respectively. Since $\ominus$-objects similar to $\ominus(*(a,b))$ can be composed from a and $\ominus(b)$ by an application of "$*$", we generalize the notions "*a left* and *a right* $*$-part" to $\ominus$-objects. In general, m (as $*$-object or $\ominus$-object) can have arbitrary many *permutable* left $*$-parts. We identify these with the help of the auxiliary function map_*, defined for given multiset ms and message m by

$$map_*(ms, m) = \begin{cases} m & : \quad ms = \emptyset \\ *(m', map_*(ms', m)) & : \quad ms = \{m'\} \uplus ms' \end{cases}$$

  For a $*$-object m, we call a multiset ms and m_0 *the left* $*$-parts and respectively *the right* $*$-part of m (and also of $\ominus(m)$) when these satisfy

$$(\forall m_1, m_2 : \neg obj^*(m_0, m_1, m_2)) \wedge (m_3 \in ms \Rightarrow inv(m_3) \notin ms) \wedge m = map_*(m_0, ms).$$

  To express multiple simplifications of left $*$-parts, we use the auxiliary function map_{inv} defined for given multiset ms by

$$map_{inv}(ms) = \begin{cases} \emptyset & : \quad ms = \emptyset \\ \{inv(m')\} \uplus map_{inv}(ms') & : \quad ms = \{m'\} \uplus ms' \end{cases}$$

5.5.2 (In-)equal Messages

The majority of our proof tasks consist in (dis-)proving the equality between two messages, which is carried out by the low-level proof heuristic `eqMsg` (see Sec. 7.2.2.5). This heuristic applies axioms and canonically formalized theorems to equalities occurring in the premises and the conclusions of proof goals. The axioms and theorems that lead to less sub-goals (in the optimal case to no sub-goals) are tried first.

In addition to axioms about the inequalities of atomic messages, we employ five kinds of simple theorems that allow us to close proof goals by contradiction:

1. For $f \in \Sigma\langle n \rangle$ where the axiom about the operations of f includes only the obj^f-case, we use theorems of the form

$$(m \in At \vee m \in \Sigma\langle 0 \rangle) \Rightarrow m \neq f(m_0, \ldots, m_{n-1}).$$

   Compare theorem $\infty \neq h_1(m_0, m_1, m_2)$ in TC-AMP.

2. For different $f, g \in \Sigma$ where the axiom about the operations of f and respectively that about the operations of g includes only the obj^f-case and the obj^g-case, respectively, we use theorems of the form

$$f(m_0, \ldots, m_{n-1}) \neq g(m'_0, \ldots, m'_{n'-1}).$$

   Compare theorem $dh(m_0, m_1) \neq mac(m'_0, m'_1)$ in PACE.

3. For $f \in \Sigma\langle n \rangle$ where the axiom about the operations of f includes non-constructor-type cases, we use theorems of the form

$$\begin{aligned} &((m \in At \vee m \in \Sigma\langle 0 \rangle) \wedge obj^f(f(m_0, \ldots, m_{n-1}), m_0, \ldots, m_{n-1})) \\ &\Rightarrow m \neq f(m_0, \ldots, m_{n-1}). \end{aligned}$$

   Compare theorem $obj^*(*(m_0, m_1), m_0, m_1) \Rightarrow \infty \neq *(m_0, m_1)$ in TC-AMP.

4. For $f, g \in \Sigma$ where the axiom about the operations of f includes non-constructor-type cases and where the axiom about the operations of g includes only the obj^g-case, we use theorems of the form

$$obj^f(f(m_0,\ldots,m_{n-1}),m_0,\ldots,m_{n-1}) \Rightarrow f(m_0,\ldots,m_{n-1}) \neq g(m'_0,\ldots,m'_{n'-1}).$$

   Compare theorem $obj^{enc}(enc(m_0,m_1),m_0,m_1) \Rightarrow enc(m_0,m_1) \neq mac(m'_0,m'_1)$ in PACE.

5. For different $f, g \in \Sigma$ where the axiom about the operations of f and that about the operations of g include non-constructor-type cases, we use theorems of the form

$$\begin{aligned}&(obj^f(f(m_0,\ldots,m_{n-1}),m_0,\ldots,m_{n-1}) \wedge obj^g(g(m'_0,\ldots,m'_{n'-1}),m'_0,\ldots,m'_{n'-1}))\\ &\Rightarrow f(m_0,\ldots,m_{n-1}) \neq g(m'_0,\ldots,m'_{n'-1}).\end{aligned}$$

   In TC-AMP, we obtain according to this principle for instance theorem

$$(obj^*(*(m_0,m_1),m_0,m_1) \wedge obj^{\oplus}(\oplus(m'_0,m'_1),m'_0,m'_1)) \Rightarrow *(m_0,m_1) \neq \oplus(m'_0,m'_1).$$

Note that `eqMsg` applies theorems of kind (3)–(5) only to proof goals that include corresponding obj^f-premises.

If theorems of kind (1)–(5) are not applicable, `eqMsg` tries to apply the axioms generated according to schema 27-(1–4). These axioms are applied in a controlled way and only to proof goals that include corresponding obj^f-premises.

Required obj^f-premises are included through the application of the axioms about the operations of f. When such axiom includes non-constructor-type cases, the proof goal is reduced by the corresponding case distinction. The resulting sub-goals are then simplified applying equations (the rewrite rules and the permutative equations in one direction).

5.5.3 Short-cuts and Abbreviations

In this section we introduce certain short-cuts and abbreviations that we often use in the rest of the thesis.

We sometimes use predicates $isObj^f$ instead of obj^f in contexts where the f-parts can be abstracted away. $isObj^f$ are defined by

$$isObj^f(m) \Leftrightarrow (\exists m_0,\ldots,m_{n-1} : obj^f(m,m_0,\ldots,m_{n-1})).$$

We sometimes use the following short-cuts:

- We use the notations $f[m_0,\ldots,m_{n-1}]$ and respectively $m = f[m_0,\ldots,m_{n-1}]$ to abbreviate $obj^f(f(m_0,\ldots,m_{n-1}),m_0,\ldots,m_{n-1})$ and $obj^f(m,m_0,\ldots,m_{n-1})$, respectively.
- We use the notations $\overline{dh}(m,ms)$, $\overline{\oplus}(ms)$, $\overline{*}(ms,m)$ and respectively $\overline{inv}(ms)$ to abbreviate $map_{dh}(m,ms)$, $map_{\oplus}(ms)$, $map_*(ms,m)$ and $map_{inv}(ms)$, respectively.

  Similar to the previous abbreviation, we use $\overline{\oplus}[ms]$, $\overline{*}[ms,m]$ and respectively $\overline{inv}[ms]$ to express that these function applications include only constructor-type applications of "$\oplus$", "$*$" and "inv", respectively.
- In our formalization tool (VSE), we use lists as multisets, extended clearly with the required notions of multisets. So, we sometimes denote multisets of the form $\{m_0\} \uplus \ldots \uplus \{m_{n-1}\} \uplus ms$ as a list of the form $m_0 \bullet \ldots \bullet m_{n-1} \bullet ms$. We also abbreviate multisets composed from particular elements $m_0,\ldots m_{n-1}$ and from rest multisets ms by $m_0 \uplus \ldots \uplus m_{n-1} \uplus ms$.

Further short-cuts and auxiliary notions will be introduced in the rest of the thesis.

5.5.4 Definition of uses

In section 7.1.5 we define freshness with the help of a binary relation *uses* on messages. $uses(m_0, m_1)$ holds iff m_1 occurs in m_0. This relation is defined recursively based on the substructure notion induced by the obj^f-predicates.

In PACE, we obtain the following definition.

Definition[VSE] 42 (uses ; PACE):

$$
\begin{aligned}
&uses(m_0, m_1) \Leftrightarrow \\
&(m_0 = m_1 \vee \\
&\ (\exists m_2 : (obj^{fst}(m_0, m_2) \vee obj^{snd}(m_0, m_2)) \wedge uses(m_2, m_1)) \vee \\
&\ (\exists m_2, m_3 : (obj^{pair}(m_0, m_2, m_3) \vee obj^{gen}(m_0, m_2, m_3) \vee obj^{dh}(m_0, m_2, m_3) \vee \\
&\qquad obj^{mac}(m_0, m_2, m_3) \vee obj^{enc}(m_0, m_2, m_3) \vee obj^{dec}(m_0, m_2, m_3)) \\
&\qquad \wedge (uses(m_2, m_1) \vee uses(m_3, m_1))))
\end{aligned}
$$

This definition is adapted straightforwardly to obtain the following definition of *uses* in TC-AMP.

Definition[VSE] 43 (uses ; TC-AMP):

$$
\begin{aligned}
&uses(m_0, m_1) \Leftrightarrow \\
&(m_0 = m_1 \vee \\
&\ (\exists m_2 : (obj^{fst}(m_0, m_2) \vee obj^{snd}(m_0, m_2) \vee \\
&\qquad obj^{inv}(m_0, m_2) \vee obj^{\ominus}(m_0, m_2)) \\
&\qquad \wedge uses(m_2, m_1)) \vee \\
&\ (\exists m_2, m_3 : (obj^{pair}(m_0, m_2, m_3) \vee obj^{*}(m_0, m_2, m_3) \vee obj^{\oplus}(m_0, m_2, m_3)) \\
&\qquad \wedge (uses(m_2, m_1) \vee uses(m_3, m_1))) \vee \\
&\ (\exists m_2, m_3, m_4 : (obj^{h_1}(m_0, m_2, m_3, m_4) \vee obj^{h_2}(m_0, m_2, m_3, m_4)) \\
&\qquad \wedge (uses(m_2, m_1) \vee uses(m_3, m_1) \vee uses(m_4, m_1))))
\end{aligned}
$$

According to this definition, $uses(\oplus(\ominus(a), \ominus(b)), \oplus(a, b))$ does not hold, although $\oplus(\ominus(a), \ominus(b))$ is derivable from $\oplus(a, b)$ by the application of $\ominus$. In Chap. 6, we will refer to $\oplus(a, b)$ as an implicit substructure of $\oplus(\ominus(a), \ominus(b))$.

For the verification of PACE and TC-AMP, the above definition of *uses* based on explicit substructures is sufficient to specify new items, which do not occur before in prior protocol traces. Theoretically, there can be other protocol settings where the definition of *uses* needs to incorporate implicit substructures, too.

Chapter 6

Inductive Proof Technique

The proofs of protocol properties include (eventually) arguments that the attacker is *not able to derive* certain messages or message parts s from the (*immediately*) *observable message sets* ik. In this chapter we describe our approach to show these proof arguments *by induction* on all (protocol-given) message sets ik. It is based on the observation that the protocol (designer) aims at the protection of these so-called *targeted secrets* s through the protection of long-term secrets from the initial knowledge and of further message parts generated during protocol runs. Typical secrets from the initial knowledge are private keys, shared keys and passwords. The protected message parts generated during protocol runs are generally nonces.

Basically, the protocol security relies on the intention that the attacker is not able to violate the protection of these basic secrets, despite their occurrence in protocol messages. We present *basic check-functions* used to verify this intention. They are used with *finite* sets S of selected secrets to check that the protocol messages are not "critical" for the protection of S. Basic check-functions must be correct with respect to the DY closure: If the tests of all protocol messages $m \in ik$ are negative, i.e. no $m \in ik$ is "critical" for the protection of S, then $DY(ik) \cap S$ is empty.

We start (in Sec. 6.1) with an introduction of the basic ideas and the general principles, discussing the tackled problems, before we describe the proposed solutions (in Sec. 6.2 – 6.5).

6.1 Basic Ideas and General Principles

Before we discuss the general principles of our approach to verifying confidentiality properties, we introduce the basic ideas in comparison with Paulson's approach.

6.1.1 Basic Ideas

During a protocol run the intruder sees the messages that are exchanged between the other protocol participants. In each protocol step the prior (immediately) observable messages in ik_p are extended to $ik = ik_p \cup \{m\}$. Starting with this basic knowledge ik the intruder derives additional messages according to the inductive definition of $DY(ik)$. To show that some (targeted) message s is kept confidential we have to prove that $s \notin DY(ik)$ holds.

Showing that something is not derivable is a notoriously difficult task in most cases. Without referring to message algebras and DY derivations Paulson has defined a function $analz$ recursively on ik, [82]. For an *atomic* secret s, in the induction step, one then has to show that $s \notin analz(ik_p)$ implies $s \notin analz(ik_p \cup \{m\})$. In case s is a composed message, one need to show that "s cannot be $synth$esized from $analz(ik_p)$" implies "s cannot

be *synth*esized from $analz(ik_p \cup \{m\})$". This task is achieved by *reducing* the confidentiality of s to the confidentiality of its substructures, employing a second (predicate) function *synth*. It determines inductively all messages that can be synthesized from any given finite message set.

The definitions of *analz* and *synth* in the original inductive approach are relatively simple, since only decomposition of message pairs, simple decryption and standard constructor application had to be taken into account. It was for that reason possible to come up with a general theorem employed in the (fake) case where the arbitrary message m in the induction step belongs to the intruder knowledge given by $synth(analz(ik_p))$. This theorem permits, in case $m \in synth(analz(ik_p))$, to reduce $s \notin synth(analz(ik_p \cup \{m\}))$ back to $s \notin synth(analz(ik_p))$.

Proofs of confidentiality of targeted (single) secrets using *analz* and *synth* necessitate to reason on how the protocol protects them. This is generally traced back to specific (confidentiality) properties in form of (proof-structuring) lemmas. For instance, long-term private keys of non-compromised participants are confidential. A second example are the so-called session key compromise theorems, [82], which guarantee that secrets are not encrypted using compromised session keys.

Instead of adapting *analz* and *synth* to other kinds of basic operations, we take a different route. Our approach allows us not only to work directly with the more intuitive definition of DY-knowledge, but also to do with much less proof-structuring lemmas. For a targeted secret s, we analyze the regular protocol messages to select message parts that need to be confidential for the protection of s. The result is a finite set S, including s, which we call the set of selected secrets. While $analz(ik)$ includes the message parts that are derivable by extraction from regular protocol messages, the set S contains message parts that must not be derivable.

For a given set S of selected secrets, we define an extension, denoted by $ccl(S)$, such that $ik \cap ccl(S) = \emptyset$ guarantees the secrecy of (the elements in) S, i.e. $S \cap DY(ik) = \emptyset$: No element of $ccl(S)$ may belong to the observable messages, otherwise the confidentiality of S could be violated.

For $S = \{a\}$ (containing a as a targeted secret), we immediately see that $a \in ccl(S)$ and $pair(a,b) \in ccl(S)$. In the latter case a derivation using fst would yield a in one step. For $enc(b,a)$ we see that $b \notin ccl(S)$, since a cannot be derived from b. This means that b is allowed in ik and consequently $enc(b,a)$ has to be critical by setting $enc(b,a) \in ccl(S)$. If we want to protect a (in $enc(b,a)$) against decryption we have to add the key b to S as well. Note that the set of messages for which the protection actually works is $ccl(S_{ik})$, where S_{ik} is an appropriate extension of $\{a\}$ according to the protocol messages in ik.

One may consider $ik \cap ccl(S) = \emptyset$ as a generic invariant where the parameter S depends on the protocol under consideration while the definition of *ccl* depends only on the underlying algebra.

While S is finite, $ccl(S)$ is an infinite extension of S that is closed under a certain "critical" property relative to S. We define $ccl(S)$ with the help of what we call a "*critical*" *closure* function (*ccl*-function), where *ccl* uses technically a predicate that checks candidates m for being in $ccl(S)$. The predicate $check_{ccl}(m,S)$ is defined recursively on the structure of m. We have $ccl(S) = \{m \mid check_{ccl}(m,s)\}$. For sake of readability we continue to use $m \in ccl(S)$ for $check_{ccl}(m,S)$ and refer to the predicate $check_{ccl}$ with *ccl*-function, too.

For our purpose, i.e. proving the confidentiality of s as described above by showing that ik is disjoint with $ccl(S)$, it is necessary to use a correct *ccl*-function when checking the elements in ik relative to S. This means, it must follow, whenever the checks $check_{ccl}(m,S)$ of all messages m in ik are negative, that S is disjoint with $DY(ik)$.

Clearly, confidentiality proofs necessitate to consider all possible effects of operations wrt. the derivation of messages. That is, the definition of *ccl*-functions appropriate for the above introduced proof technique heavily depends on the basic operations of the under-

lying message algebra. Nevertheless, we came up with general principles that we applied in our definition of the *ccl*-functions used in our verification of PACE and TC-AMP. These general principles are described in the following section, where we also discuss related problems together with proposed solutions.

6.1.2 Definition and Use of *ccl*-Functions

The definition of an appropriate *ccl*-function that can be used as a basic check-function to prove confidentiality as described in Sec. 6.1.1 heavily depends on the basic operations provided by the message algebra. For a given set S of selected secrets, $ccl(S)$ is intuitively an extension of S with arbitrary messages that *might* be sources for derivations of messages in S. Accordingly, an arbitrary given m belongs to $ccl(S)$ *iff* there is some $s_i \in S$ that can be derived from m and all additional items $m_0, \ldots, m_n$ required for this derivation are not critical for the protection of S. That is, for given m and S we check in the definition of *ccl*

1. whether m occurs in S, or
2. whether there is a substructure s_i of m that occurs in S and whether
   (a) there is a derivation of s_i from m without needing additional items, e.g., s_i is derivable from $pair(s_i, m')$ by applying fst and without using any additional item,
   (b) or there is a derivation of s_i from m using additional items that are not critical for the protection of S.

It is clear that obtaining $ik \cap ccl(S) = \emptyset$ with a *ccl*-function defined according to the above principles excludes all derivations of selected secrets by extraction. However, excluding derivations by extraction is not enough as illustrated in the following example: For $S = \{mac(a,b)\}$ and $ik = \{a,b\}$, we obviously obtain $ik \cap ccl(S) = \emptyset$. Though, we clearly have $mac(a,b) \in DY(ik)$ and thus $DY(ik) \cap S = \emptyset$ does not hold. This example shows that derivations by composition need to be excluded through additional conditions on S. To protect $mac(a,b)$ against derivation by composition, we need to include a or b as additional selected secrets. Using for instance $S' = \{mac(a,b), a\}$, we obtain $a \in ccl(S')$ and thus $ik \cap ccl(S') = \emptyset$ does not hold.

Consequently, basic check-functions shall be used only with sets S of selected secrets that fulfill tailored additional conditions, so that all possible derivations are taken into consideration.

To make sure that a *ccl*-function can be used correctly as a basic check-function, we have to prove that

$$ik \cap ccl(S) = \emptyset \Rightarrow DY(ik) \cap ccl(S) = \emptyset$$

holds (under appropriate conditions on S).

Note that the generalized conclusion $DY(ik) \cap ccl(S) = \emptyset$ is stronger than our ultimate proof goal $DY(ik) \cap S = \emptyset$. This is crucial in our proof technique for two reasons:

- The proof of this correctness theorem is by induction on the DY-levels; Through the generalized form, we obtain an induction hypothesis that is strong enough, so that we can handle the step case by a case distinction on the possible basic operations provided by the message algebra (see Sec. 6.2.4).
- This correctness theorem permits us to apply the *ccl*-function in confidentiality proofs *only* to the *regular* protocol messages and to handle the faked messages (by the attacker) for free, based on the induction hypothesis. In the fake case, the proof goal $(ik_p \cup \{m\}) \cap ccl(S) = \emptyset$ follows immediately (by the correctness theorem) from the induction hypothesis $ik_p \cap ccl(S) = \emptyset$ and from $m \in DY(ik_p)$.

As introduced above, the sets of the selected secrets are determined relative to the observable regular protocol messages. That is, we need to provide for every ik a set S_{ik} satisfying $ik \cap ccl(S_{ik}) = \emptyset$. This existence conjecture is shown by induction on ik. In the step case, the proof goal consists in providing S_{ik} (by extending S_p from the induction hypothesis) such that $(ik_p \cup \{m\}) \cap ccl(S_{ik}) = \emptyset$ can be shown with the help of the induction hypothesis $ik_p \cap ccl(S_p) = \emptyset$. Fortunately, S_{ik} equals S_p or is a partition $S_p \uplus S_e$ of S_p and some set S_e.

1. The set S_e may not be empty *only* when the last message m could be used to derive selected secrets in S_p. New selected secrets are required to prevent that m violates the protection of S_p.

2. Furthermore, including additional selected secrets in S_e is *necessary only when* these secrets cannot be fixed before the generation of m. This is usually the case when the last protocol step giving rise to m generates a *fresh* message part (generally a *new nonce*) $\widetilde{m}$.

Making sure that

$$(\forall s \in S_e : uses(s, \widetilde{m})) \wedge (\forall s \in S_p : \neg uses(s, \widetilde{m})) \tag{6.1}$$

holds, we require that the used *ccl*-functions fulfill

$$(\neg uses(m, \widetilde{m}) \wedge m \in ccl(S_p \uplus S_e)) \Rightarrow m \in ccl(S_p). \tag{6.2}$$

Since $\neg uses(m', \widetilde{m})$ holds for all $m' \in ik_p$, conditions (6.1) and (6.2) permit us to use the induction hypothesis $ik_p \cap ccl(S_p) = \emptyset$ to reduce $(ik_p \cup \{m\}) \cap ccl(S_p \uplus S_e) = \emptyset$ to the check that the the last message m is not critical, i.e. to $m \notin ccl(S_p \uplus S_e)$.

Following the above described principles, we came up with a *canonical ccl*-function ccl_1, which is defined according to the same *simple* schema in all message algebras, as described in Sec. 6.2 and 6.3 for ccl_1 in PACE and respectively TC-AMP. Our verification of PACE and TC-AMP makes use of ccl_1 as the sole basic check-function in all confidentiality proofs about the confidential message parts occurring in protocol steps. Besides this kind of what we call *basic confidentiality properties*, protocol verification (also of PACE and TC-AMP) includes proof situations where *only* partial knowledge about the structure of a targeted secret s is given. In such proof situations, it is (often) not possible to prove the confidentiality of s by providing finite sets of selected secrets (without the need of auxiliary notions). Instead of that, we use generic theorems and a second canonical *ccl*-function ccl_2 to reduce the protection of s to the protection of substructures of s (see Sec. 6.4). The check-function ccl_2 is defined relative to s only and that by a *straightforward* adaptation of ccl_1, where additional items in derivations are neglected. By $ik \cap ccl_2(s) = \emptyset$ we check that s is not derivable by *extraction* (*en-bloc*) from all messages in ik, so that s can be derived only by *composition*. The generic theorems are used to reduce s to the message parts required for a composition of s, according to its top structure.

The reduction theorems and ccl_2, which permit for a kind of backward reasoning, do not apply when s can be composed using items different from its substructures. That is, the structure of s does not limit the infinite number of such candidate items. This can be only limited by following a kind of forward reasoning in protocol-specific invariants about all derivable messages $\hat{m}$ having the same type like s. In particular, the invariant links $\hat{m}$ to basic confidential message parts, so that their occurrence in $\hat{m}$ shall be shown to be inconsistent with that in s, which basically forms the main confidentiality argument of s. The occurrence of an arbitrary but fixed $\hat{m}$ is generalized in this kind of invariants to an infinite message set given by a (specific) *ccl*-function (applied to $\hat{m}$), in order to support the invariant verification by nested induction, first on protocol traces and then on DY-levels (see Sec. 6.5.1).

After the description of the *ccl*-functions used in the verification of PACE and TC-AMP, we demonstrate (in Sec. 6.5.2) that our proof technique of basic confidentiality properties can be extended to proof situations where the canonical basic check-function ccl_1 does not apply: Using an example (key establishment) protocol, we argue that the basic confidentiality of the session key necessitates sets S of selected secrets that do not satisfy the required conditions for being correctly used with ccl_1 as basic check-function. For the discussed kind of basic confidentiality proofs, we propose another basic check-function ccl_3 and sketch its correctness proof, including the corresponding additional conditions on these new sets S of selected secrets.

6.2 Basic Check-Function ccl_1 in PACE

The use of basic check-functions heavily depends on the provided operations by the message algebra. In case of PACE, we identify the following kinds of basic operations.

6.2.1 Characterization of Basic Operations

Confidentiality proofs necessitate to consider all possible effects of operations wrt. the derivation of messages. Analyzing the (axioms about) operations in PACE, we come up with the characterization of their effects, as described below.

To characterize operations $f(m_0,\ldots,m_{n-1})$ resulting in m', we focus on structural relations about $m_0,\ldots,m_{n-1}$ and m' that are crucial in dealing with corresponding derivation schemata in our proof technique.

Aside from the constructor-type operation, where $m_0,\ldots,m_{n-1}$ are f-parts, i.e. substructures, of m, we identify the following kinds of operations.

selector-type operation: The application of "f" to a g-object m_0 is called a *selector-type* operation (abbreviated by sel_g^f), when the result $m = f(m_0)$ is a substructure of m_0. We refer to the result m as a *select-part* of the g-object m_0.

Altogether, the PACE algebra includes two selector-type operations: sel_{pair}^{fst} and sel_{pair}^{snd} (e.g., $snd(pair(a,b)) = b$). Corresponding predicates can be defined as auxiliary notions, according to the basic operations. For instance, $sel_{pair}^{fst}(m,m_0)$ can be defined by

$$sel_{pair}^{fst}(m,m_0) \Leftrightarrow (\exists m_1 : obj^{pair}(m_0,m,m_1)).$$

decrypt-type operation: The application of "f" to $m_0,\ldots,m_{n-1}$ where the result $m = f(m_0,\ldots,m_{n-1})$ is a substructure of a g-object m_i and where $m_0,\ldots,m_{i-1},m_{i+1},\ldots,m_{n-1}$ can be fixed relative to substructures of m_i is called a *decrypt-type* operation (abbreviated by dec_g^f). We refer to the result m as a *crypt-part* of the g-object m_i and to $m_0,\ldots,m_{i-1},m_{i+1},\ldots,m_{n-1}$ as the corresponding *crypt-keys*.

Altogether, the PACE algebra includes two decrypt-type operations: dec_{dec}^{enc} and dec_{enc}^{dec} (e.g., $dec(a,enc(a,b)) = b$). Corresponding predicates can be defined as auxiliary notions, according to the basic operations. For instance, $dec_{dec}^{enc}(m,m_0,m_1)$ can be defined by

$$dec_{dec}^{enc}(m,m_0,m_1) \Leftrightarrow obj^{enc}(m_1,m_0,m).$$

When applying the *ccl*-function ccl_1, we want to treat derivations by extraction of *explicit* substructures, as permitted by the selector-type and the decrypt-type operations, together with derivations by composition using as well *explicit* substructures, as permitted by the constructor-type operations. So, we refer to these operations by the *canonical operations*.

6.2.2 Definition of ccl_1

Referring to the general required checks by basic check-functions in Sec. 6.1.2, we define $m \in ccl_1(S)$ *iff* one of the following cases holds:

- $m \in S$ (base case corresponding to check (1)),
- there is a select-part m_i of m such that $m_i \in ccl_1(S)$ (recursion case(s) corresponding to check (2-a)), or
- there is a crypt-part m_i of m such that $m_i \in ccl_1(S)$ and all required crypt-keys $[m, m_i]_j^{key}$ satisfy $[m, m_i]_j^{key} \notin ccl_1(S)$ (recursion case(s) corresponding to check (2-b)).

For each f-object and each identified *explicit* select-part or decrypt-part, we have one recursion case. According to the canonical operations of PACE, we obtain thus the following definition of ccl_1:

Definition^{VSE} 44 (ccl_1 ; PACE):
For a finite message set S and an arbitrary message m we have

$$\begin{aligned} m \in ccl_1(S) \Leftrightarrow (m \in S \vee \\ \exists m_0, m_1 : (obj^{pair}(m, m_0, m_1) \wedge (m_0 \in ccl_1(S) \vee m_1 \in ccl_1(S))) \vee \\ \exists m_0, m_1 : (obj^{enc}(m, m_0, m_1) \wedge m_1 \in ccl_1(S) \wedge m_0 \notin ccl_1(S)) \vee \\ \exists m_0, m_1 : (obj^{dec}(m, m_0, m_1) \wedge m_1 \in ccl_1(S) \wedge m_0 \notin ccl_1(S))). \end{aligned}$$

Before we discuss the correct use of ccl_1 as a basic check-function, we make sure that ccl_1 satisfies condition (6.2) required for proof situations where the set of selected secrets (from the induction hypothesis) must be extended with other items (see Sec. 6.1.2).

Theorem^{VSE} 45 (Extension Condition of $\mathbf{ccl_1(S)}$):
Let $m, \widetilde{m}$ be two arbitrary messages and let S, E be two finite sets of messages satisfying

$$(\forall s \in S : \neg uses(s, \widetilde{m})) \wedge (\forall s \in E : uses(s, \widetilde{m})).$$

Then we have

$$\neg uses(m, \widetilde{m}) \Rightarrow (m \in ccl_1(S \uplus E) \Leftrightarrow m \in ccl_1(S)).$$

This theorem is identical in all message algebras and it is shown by structural induction, i.e. by induction on $|m|$, according to the same proof schema:

Proof (Schema):

Base-Case: Here, m is atomic. In the left-right case, $m \in ccl_1(S \uplus E)$ implies $m \in S \uplus E$ and the *uses*-conditions permit us to obtain $m \in S$ and thus $m \in ccl_1(S)$, as required.

In the right-left case, we assume $m \notin ccl_1(S \uplus E)$ and want to show $m \notin ccl_1(S)$. For an atomic m, the assumption implies $m \notin S \uplus E$ and thus $m \notin S$. The latter is sufficient to get $m \notin ccl_1(S)$, as required.

Step-Case: Here, m is an f-object. In the left-right case, we expand $m \in ccl_1(S \uplus E)$ by the definition of ccl_1 and the resulting cases can be distinguished as follows:

- In the base case, we get $m \in S \uplus E$: It is shown as in the Base Case.

- In a recursion case corresponding to a selector-type operation, the definition of ccl_1 yields $obj^g(m, m_0, \ldots, m_{n-1})$ and $m_i \in ccl_1(S \uplus E)$: Since $\neg uses(m, \widetilde{m})$ implies $\neg uses(m_i, \widetilde{m})$, we may use the induction hypothesis to obtain $m_i \in ccl_1(S)$ and then employ the definition of ccl_1 to show $m \in ccl_1(S)$, as required.
- In a recursion case corresponding to a decrypt-type operation, the definition of ccl_1 yields $obj^g(m, m_0, \ldots, m_{n-1})$ and $m_i \in ccl_1(S \uplus E)$, where the corresponding crypt-keys $[m, m_i]_1^{key}$, ..., $[m, m_i]_{n'}^{key}$ are not in $ccl_1(S \uplus E)$: The crypt-keys are expected to be smaller than m and not to include $\widetilde{m}$ as a message part. This permits to apply the induction hypothesis and obtain that they do not belong to $ccl_1(S)$. Additionally, $m_i \in ccl_1(S)$ is ensued like in the previous case. All these permit us to employ the definition of ccl_1 and obtain $m \in ccl_1(S)$, as required.

The right-left case is handled as in the base case, by assuming $m \notin ccl_1(S \uplus E)$ and showing $m \notin ccl_1(S)$. The assumption $m \notin ccl_1(S \uplus E)$ yields

1. $m \notin S \uplus E$, and
2. if $obj^f(m, m_0, \ldots, m_{n-1})$ holds, then for all $i \in \{0, \ldots, n-1\}$
   (a) $m_i \notin ccl_1(S \uplus E)$, or
   (b) some of the crypt-keys (if any) $[m, m_i]_1^{key}$, ..., $[m, m_i]_{n'}^{key}$ belongs to $ccl_1(S \uplus E)$.

(1) implies $m \notin S$, as in the base case, and (2) transfers to $ccl_1(S)$, by the induction hypothesis and based on the same argument as in the left-right case. This permits us to employ the definition of ccl_1 and obtain $m \notin ccl_1(S)$, as required. □

In the corresponding proof (for the PACE algebra), we need to consider concrete message structures. For instance, the Step-Case includes a case where $obj^{enc}(m, m_0, m_1)$ holds. Here, the definition of ccl_1 provides in the left-right case practically two proof situations: $m \in S \uplus E$, which is handled as in the Base-Case, and $m_1 \in ccl_1(S \uplus E) \wedge m_0 \notin ccl_1(S \uplus E)$, which is handled by the induction hypothesis to get $m_1 \in ccl_1(S) \wedge m_0 \notin ccl_1(S)$ and then by the definition of ccl_1 to obtain $m \in ccl_1(S)$.

6.2.3 Appropriate Sets of Selected Secrets

Basically, we want to use ccl_1 as basic check-function to reliably exclude derivations of selected secrets by canonical operations. According to the general principles in Sec. 6.1.2, appropriate sets S of selected secrets used correctly with ccl_1 must satisfy an additional condition that excludes derivations by constructor-type operations. In the light of this requirement, appropriate sets S for the confidentiality of a targeted secret s are obtained by extending $\{s\}$ according to the following principles:

1. ("decrypt-prevention" principle): If a regular protocol message $m \in ik$ includes a "*public*" message part m' such that some selected secret is crypt-part of m', then *at least one* of the corresponding crypt-keys must belong to the selected secrets.
2. ("f-part inclusion" principle): If a selected secret is an f-object, then for each tuple of its f-parts, *at least one* f-part must belong to the selected secrets.

Contrarily to rule (1), where the added selected secrets generally include *new* message parts from regular protocol messages, selected secrets included due to rules (2) correspond *usually* to message parts of *prior* selected secrets. That is, rule (2) can be seen as a necessary *closure property* for sets of selected secrets.

Actually, the extension principles (1) and (2) are rather guidance instructions on how to define the content of S relative to the regular protocol steps: Referring to the general principles in Sec. 6.1.2, the sets S of selected secrets are provided during the induction proof of a

protocol invariant. In simple basic confidentiality proofs, one appropriate set S can be fixed for all protocol traces. Other basic confidentiality proofs necessitate successive extensions of $\{s\}$ according to the protocol steps. In such proofs, the sets S are defined as *smallest sets* closed under protocol-specific inclusion rules, translating the extension principles (1) and (2). In the step case of these proofs, S is given by an extension E of a set S_p provided by the induction hypothesis, i.e. $S = S_p \uplus E$. For the last protocol steps where E is not empty, we need

- to satisfy the protocol-specific inclusion rules defining S,
- paying attention to requirement (6.1) in Sec. 6.1.2.

The cases where E is not empty shall coincide with those protocol steps where *new* message parts $\widetilde{m}$ are generated. This permits to fulfill requirement (6.1), i.e. to make sure that a message part $\widetilde{m}$ occurs in each selected secret in E and does not occur in any element of S_p.

In Sec. 8.2, we discuss the basic confidentiality properties of PACE, which are shown using ccl_1 as basic check-function. For instance, property 78-(1) is about the confidentiality of the fresh generator $gen(dh(g,nc_1),m)$. It is shown employing

$$S = \{gen(dh(g,nc_1),m), dh(g,nc_1), nc_1, pwd(ag_A)\}$$

as the set of selected secrets, fixed for all protocol traces containing a first PACE message $enc(pwd(ag_A),nc_1)$.

It is clear that S is closed under the "f-part inclusion" principle.

In case of this (simple) confidentiality property, there is no need for extending the fixed set S: The selected secrets in S do not occur (outside of $enc(pwd(ag_A),nc_1)$ neither as select-part nor) as crypt-part of other protocol messages.

Using $ccl_1(S)$, the checks of the regular protocol messages of PACE show that they are not critical. For instance, $enc(pwd(A),nc_1) \notin ccl_1(S)$ holds, because $enc(pwd(A),nc_1) \notin S$ and $pwd(A) \in S$. The latter excludes that nc_1 (in S) can be derived from $enc(pwd(A),nc_1)$.

6.2.4 Correctness of ccl_1 as Basic Check-Function

In this section, we prove that ccl_1 can be used as basic check-function using sets S of selected secrets that respect the "f-part inclusion" principle in Sec. 6.2.3. In case of PACE, this inclusion principle is sufficient to exclude the critical derivations by composition, as required among the general principles in Sec. 6.1.2. It is simply expressed by a conjunction of conditions

$$(m \in S \wedge obj^f(m, m_0, \ldots, m_{n-1})) \Rightarrow (m_0 \in S \vee \ldots \vee m_{n-1} \in S), \tag{6.3}$$

one for each $f \in (\Sigma \setminus \Sigma\langle 0 \rangle)$.

We use $\Re_1(S)$ in the correctness theorem of ccl_1 to denote the condition about S, which corresponds in the PACE algebra to the conjunction of the above given conditions.

Theorem$^{\mathbf{VSE}}$ 46 (Correctness of $\mathbf{ccl_1(S)}$):
Let ik be a finite message set, and let S be a finite set of messages such that $\Re_1(S)$ holds. Then we have

$$ik \cap ccl_1(S) = \emptyset \Rightarrow DY(ik) \cap ccl_1(S) = \emptyset.$$

Except of $\Re_1(S)$, which consists of the conditions for "f-part inclusion" and can include in rich message algebras further conditions to exclude critical derivations by composition, this theorem is identical in all message algebras. The proof is according to a common schema, which applies for theorems about ccl-functions shown by induction on DY-levels.

Proof

The proof of theorem 46 is by induction on the extension of the DY-knowledge. This means for ccl_1, we show

$$ik \cap ccl_1(S) = \emptyset \Rightarrow DYl(ik,n) \cap ccl_1(S) = \emptyset$$

by induction on n.

Base Case: It is obvious, since $DYl(ik,0) = ik$.

Step Case: We assume that some m from $DYl(ik,n+1)$ belongs to $ccl_1(S)$ and try to find some m_i from $DYl(ik,n) \cap ccl_1(S)$, which permits us to apply the induction hypothesis and obtain $ik \cap ccl_1(S) \neq \emptyset$:

First, the definition of DYl provides us with a case distinction on how such a message m is obtained. Each case fixes some function symbol $f \in Op$ and (candidate) messages $m_0,\ldots,m_{N_f-1}$ (for m_i) from $DYl(ik,n)$ with $m = f(m_0,\ldots,m_{N_f-1})$. The next step consists in employing $f(m_0,\ldots,m_{N_f-1}) \in ccl_1(S)$ and the axiom about the operations of $f(m_0,\ldots,m_{N_f-1})$ to show that some message from $m_0,\ldots,m_{N_f-1}$ belongs to $ccl_1(S)$.

Case "f = dh": The axiom about the operations of $dh(m_0,m_1)$ provides us only with the obj^{dh}-case, where $obj^{dh}(dh(m_0,m_1),m_0,m_1)$ holds. Here, the definition of ccl_1 applied to $dh(m_0,m_1) \in ccl_1(S)$ yields practically only the base case, i.e. $dh(m_0,m_1) \in S$. Then, $\Re_1(S)$ permits to obtain $m_i \in S$ and thus $m_i \in ccl_1(S)$, for $m_i \in \{m_0,m_1\}$.

Cases "$f = mac$" and "$f = gen$" are similar.

Case "f = pair": The axiom about the operations of $pair(m_0,m_1)$ provides us only with the obj^{pair}-case, where $obj^{pair}(pair(m_0,m_1),m_0,m_1)$ holds. Here, the definition of ccl_1 applied to $pair(m_0,m_1) \in ccl_1(S)$ yields practically three cases:

1. $pair(m_0,m_1) \in S$: This case is handled based on $\Re_1(S)$ as in the "$f = dh$"-case.
2. $m_0 \in ccl_1(S)$: This case is trivial.
3. $m_1 \in ccl_1(S)$: This case is trivial.

Case "f = fst": The axiom about the operations of $fst(m_0)$ provides us with two cases:

1. $obj^{fst}(fst(m_0),m_0)$: This case is handled by the definition of ccl_1 similar to the "$f = dh$"-case.
2. $obj^{pair}(m_0,m_0',m_1')$ and $fst(m_0) = m_0'$: Here, we use $fst(m_0) \in ccl_1(S)$ to deduce $m_0' \in ccl_1(S)$. Then, we apply the definition of ccl_1 using $obj^{pair}(m_0,m_0',m_1')$ and $m_0' \in ccl_1(S)$ to obtain $m_0 \in ccl_1(S)$.

Case "$f = snd$" is similar.

Case "f = enc": The axiom about the operations of $enc(m_0,m_1)$ provides us with two cases:

1. $obj^{enc}(enc(m_0,m_1),m_0,m_1)$: Here, the definition of ccl_1 applied to $enc(m_0,m_1) \in ccl_1(S)$ yields practically two cases:
   (a) $enc(m_0,m_1) \in S$: This case is handled based on $\Re_1(S)$ as in the "$f = dh$"-case.

(b) $m_1 \in ccl_1(S)$ and $m_0 \notin ccl_1(S)$: This case is trivial.

2. $obj^{dec}(m_1,m_0,m_2)$ and $enc(m_0,m_1) = m_2$: To use the definition of ccl_1, we proceed by a case distinction:

   (a) $m_0 \in ccl_1(S)$: This case is trivial.

   (b) $m_0 \notin ccl_1(S)$: Here, we use $enc(m_0,m_1) \in ccl_1(S)$ to deduce $m_2 \in ccl_1(S)$. Then, we apply the definition of ccl_1 using $obj^{dec}(m_1,m_0,m_2)$, $m_2 \in ccl_1(S)$ and $m_0 \notin ccl_1(S)$ to obtain $m_1 \in ccl_1(S)$.

Case "$f = dec$" is similar. □

6.3 Basic Check-Function ccl_1 in TC-AMP

Before we discuss the use of ccl_1 in the TC-AMP algebra, we describe the identified kinds of basic operations.

6.3.1 Characterization of Basic Operations

As in Sec. 6.2.1, we identify the different kinds of operations $f(m_0,\ldots,m_{n-1})$ based on structural relations about $m_0,\ldots,m_{n-1}$ and the result $m = f(m_0,\ldots,m_{n-1})$. Besides the constructor-type operations, the TC-AMP algebra includes sel^{fst}_{pair}, sel^{snd}_{pair}, $sel^{\ominus}_{\ominus}$ (e.g., $\ominus(\ominus(a)) = a$) and sel^{inv}_{inv} (e.g., $inv(inv(a)) = a$) as selector-type operations, together with dec^{*}_{*} (e.g., $*(a,*(inv(a),b)) = b$) and $dec^{\oplus}_{\oplus}$ (e.g., $\oplus(a,\oplus(\ominus(a),b)) = b$) as decrypt-type operations, in the sense of Sec. 6.2.1. The corresponding predicates for dec^{*}_{*} and $dec^{\oplus}_{\oplus}$ are defined by:

$$dec^{*}_{*}(m,m_0,m_1) \Leftrightarrow (\exists m_2 : obj^{*}(m_1,m_2,m) \wedge m_0 = inv(m_2))$$

$$dec^{\oplus}_{\oplus}(m,m_0,m_1) \Leftrightarrow ((\exists m_2 : obj^{\oplus}(m_1,m_2,m) \wedge m_0 = \ominus(m_2)) \vee (\exists m_2 : obj^{\oplus}(m_0,m_2,m) \wedge m_0 = \ominus(m_1)))$$

The definition of dec^{*}_{*} incorporates two cases: m_0 results from m_2 by composition or by sel^{inv}_{inv}. Similarly, the definition of $dec^{\oplus}_{\oplus}$ covers two cases: m is a crypt-part of m_1 or of m_0.

In addition to the mentioned canonical operations, we identify the following non-canonical operations.

synthesis operation: A composed operation $f(m_0,\ldots,m_{n-1})$ that results in a g-object m where *all* nested recursion-free operations are constructor-type and where $m_0,\ldots,m_{n-1}$ can be fixed relative to substructures of m is called a *synthesis* operation (abbreviated by syn^{f}_{g}). We refer to $m_0,\ldots,m_{n-1}$ as *synth-parts* of the g-object m, to emphasize two features:

1. $m_0,\ldots,m_{n-1}$ are *uniformly smaller* than m, and

2. m is *derivable by composition* from $m_0,\ldots,m_{n-1}$, *besides* composition(s) from its g-parts.

Note that composed messages that can be derived by synthesis operations possess the synth-parts as *implicit substructures*.

Altogether, the TC-AMP algebra includes three different synthesis operations: $syn^{*}_{\ominus}$ (e.g., $*(a,\ominus(b)) = \ominus(*(a,b))$), $syn^{\ominus}_{\oplus}$ (e.g., $\ominus(\oplus(a,b)) = \oplus(\ominus(a),\ominus(b))$) and $syn^{*}_{\oplus}$ (e.g.,

$*(a,\oplus(b,c)) = \oplus(*(a,b),*(a,c))$). Their corresponding predicates are defined by:

$$syn^*_{\ominus}(m,m_0,m_1) \Leftrightarrow (\exists m_2 : obj^{\ominus}(m_1,m_2) \wedge obj^{\ominus}(m,*(m_0,m_2)))$$

$$syn^{\ominus}_{\oplus}(m,m_0) \Leftrightarrow (\exists m_1,m_2 : obj^{\oplus}(m_0,m_1,m_2) \wedge obj^{\oplus}(m,\ominus(m_1),\ominus(m_2)) \wedge obj^{\ominus}(\ominus(m_1),m_1) \wedge (obj^{\ominus}(\ominus(m_2),m_2) \vee syn^{\ominus}_{\oplus}(\ominus(m_2),m_2)))$$

$$syn^*_{\oplus}(m,m_0,m_1) \Leftrightarrow (\exists m_2,m_3 : obj^{\oplus}(m_1,m_2,m_3) \wedge obj^{\oplus}(m,*(m_0,m_2),*(m_0,m_3)) \wedge (obj^*(*(m_0,m_2),m_0,m_2) \vee syn^*_{\ominus}(*(m_0,m_2),m_0,m_2)) \wedge (obj^*(*(m_0,m_3),m_0,m_3) \vee syn^*_{\ominus}(*(m_0,m_3),m_0,m_3) \vee syn^*_{\oplus}(*(m_0,m_3),m_0,m_3)))$$

Note that $syn^{\ominus}_{\oplus}$ and $syn^*_{\oplus}$ are defined by recursion, as they are about $\oplus$-objects with arbitrary many basic $\oplus$-parts.

(composed) selector-type operation: A composed operation $f(m_0)$ that results in a synth-part of a g-object m_0 is also called a *selector-type* operation (abbreviated by syn^f_g). The resulting synth-part $m = f(m_0)$ is also called a select-part of m_0.

Altogether, the TC-AMP algebra includes one composed selector-type operation: $sel^{\ominus}_{\oplus}$ (e.g., $\ominus(\oplus(\ominus(a),\ominus(b))) = \oplus(a,b)$). Its corresponding predicate is defined by

$$sel^{\ominus}_{\oplus}(m,m_0) \Leftrightarrow syn^{\ominus}_{\oplus}(m_0,m).$$

(composed) decrypt-type operation: A composed operation $f(m_0,\ldots,m_{n-1})$ that results in a synth-part of a g-object m_i and where $m_0,\ldots,m_{i-1},m_{i+1},\ldots,m_{n-1}$ can be fixed relative to m_i is also called a *decrypt-type* operation (abbreviated by dec^f_g). The resulting synth-part $m = f(m_0,\ldots,m_{n-1})$ is also called a crypt-part of m_i and $m_0,\ldots,m_{i-1},m_{i+1},\ldots,m_{n-1}$ are the corresponding crypt-keys.

Altogether, the TC-AMP algebra includes two composed decrypt-type operations: $dec^*_{\ominus}$ (e.g., $*(inv(a),\ominus(*(a,b))) = \ominus(b)$) and $dec^*_{\oplus}$ (e.g., $*(inv(a),\oplus(*(a,b),*(a,c))) = \oplus(b,c)$). Their corresponding predicates are defined by

$$dec^*_{\ominus}(m,m_0,m_1) \Leftrightarrow (\exists m_2 : syn^*_{\ominus}(m_1,m_2,m) \wedge m_0 = inv(m_2))$$

$$dec^*_{\oplus}(m,m_0,m_1) \Leftrightarrow (\exists m_2 : syn^*_{\oplus}(m_1,m_2,m) \wedge m_0 = inv(m_2))$$

merge operation: The application of "f" to m_0,m_1 resulting in g-object $m = f(m_0,m_1)$ where m_0 and m have a common substructure and two distinct substructures m_2,m_3 and where m_1 can be fixed from m_2 and m_3 is called a *merge* operation (abbreviated by mrg^f_g). We refer to m_0 and m_1 as two *merge-sides* of the g-object m, to emphasize that m is *derivable* from m_0 *necessarily using* m_1 as additional item.

Here, the substructures of m_1 occur *in part* inside m_0 and *in the other part* inside m. For that reason, *neither* m_0 (resp. m_1) is *uniformly smaller* than m *nor vice versa*. Compared to a decrypt-type operation $f(m_0,m_1)$, where the result (crypt-part) and the additional item (crypt-key) are fixed *alone* by m_0 or m_1, neither m_0 nor m_1 in a merge operation is alone sufficient to fix the derivable message.

Altogether, the TC-AMP algebra includes only one merge operation: $mrg^{\oplus}_{\oplus}$ (e.g., $\oplus(\oplus(a,b),\oplus(c,\ominus(b))) = \oplus(a,c)$). Its corresponding predicate is defined by

$$mrg^{\oplus}_{\oplus}(m,m_0,m_1) \Leftrightarrow (\exists m_2,m_3,m_4 : obj^{\oplus}(m_0,m_2,m_3) \wedge obj^{\oplus}(m,m_2,m_4) \wedge obj^{\oplus}(m_1,m_4,\ominus(m_3))).$$

transform operation: The application of the function "$\ominus$" to $\oplus(\ominus(a),b)$ yields a new $\oplus$-object $\oplus(a,\ominus(b))$, where no general $|.|$-size relation between the used message and the result exists, despite the preservation of message parts (a and b). For that reason, it is called a transform operation.

More general, a composed operation $f(m_0)$ resulting in a g-object m where the nested recursion-free operations are in part selector-type and in the other part constructor-type is called a *transform* operation (abbreviated by trs_g^f). We refer to the result m as a *trans-message* of m_0.

For $n > 1$, we call a composed operation $f(m_0,\ldots,m_{n-1})$ similarly a transform operation, when *all* nested recursion-free operations are applied to substructures of the same m_i where $m_0,\ldots,m_{i-1},m_{i+1},\ldots,m_{n-1}$ are used in constructor-type operations and *in parallel* as crypt-keys in decrypt-type operations. We refer to the result $m = f(m_0,\ldots,m_{n-1})$ by a *trans-message* of m_i and to $m_0,\ldots,m_{i-1},m_{i+1},\ldots,m_{n-1}$ by the corresponding *trans-keys*, which form the necessary additional items to derive m from m_i.

Here, *neither* the transformed m_0 (resp. m_i) is *universally smaller* than m, nor vice versa.

Altogether, the TC-AMP algebra includes two transform operations: $trs_{\oplus}^{\ominus}$ and $trs_{\oplus}^{*}$ (e.g., $*(a,\oplus(*(inv(a),b),c)) = \oplus(b,*(a,c))$). Their corresponding predicates are defined by

$$\begin{aligned} trs_{\oplus}^{\ominus}(m,m_0) \Leftrightarrow (\exists m_1,m_2 : obj^{\oplus}(m_0,m_1,m_2) \wedge obj^{\oplus}(m,\ominus(m_1),\ominus(m_2)) \wedge \\ (obj^{\ominus}(\ominus(m_1),m_1) \vee syn_{\oplus}^{\ominus}(\ominus(m_1),m_1)) \wedge \\ (sel_{\ominus}^{\ominus}(\ominus(m_2),m_2) \vee sel_{\oplus}^{\ominus}(\ominus(m_2),m_2))) \end{aligned}$$

$$\begin{aligned} & trs_{\oplus}^{*}(m,m_0,m_1) \Leftrightarrow \\ & (\exists m_2,m_3 : obj^{\oplus}(m_1,m_2,m_3) \wedge obj^{\oplus}(m,*(m_0,m_2),*(m_0,m_3)) \wedge \\ & \quad (obj^{*}(*(m_0,m_2),m_0,m_2) \vee syn_{\ominus}^{*}(*(m_0,m_2),m_0,m_2) \vee syn_{\oplus}^{*}(*(m_0,m_2),m_0,m_2)) \wedge \\ & \quad (dec_{*}^{*}(*(m_0,m_2),m_0,m_2) \vee dec_{\ominus}^{*}(*(m_0,m_2),m_0,m_2) \vee dec_{\oplus}^{*}(*(m_0,m_2),m_0,m_2))) \end{aligned}$$

redundant operation: An application of "f" to $m_0,\ldots,m_{n-1}$ that results in m_i is called a *redundant* operation (abbreviated by red^f). Such an operation is not relevant in confidentiality proofs, as it does not permit to derive any *new* message.

Similarly, we call an operation $f(m_0,m_1)$ also *redundant*, when it results in c from $\Sigma\langle 0\rangle$ and when m_1 is derivable from m_0 *without additional items*. The latter condition means that c is derivable from any message, independent of its content. Such an atomic message c is always *public* and may not, for that reason, be considered neither as a selected secret nor as a critical message. Consequently, its derivation by any redundant operation is not relevant in confidentiality proofs.

Altogether, the TC-AMP algebra includes three redundant operations: $red^{\ominus}$, red^{*} (e.g., $*(a,\infty) = \infty$) and $red^{\oplus}$. Their corresponding predicates are defined by

$$red^{\ominus}(m,m_0) \Leftrightarrow (m_0 = \infty \wedge m = \infty)$$

$$red^{*}(m,m_0,m_1) \Leftrightarrow (m_1 = \infty \wedge m = \infty)$$

$$\begin{aligned} & red^{\oplus}(m,m_0,m_1) \Leftrightarrow \\ & ((m_0 = \infty \wedge m = m_1) \vee (m_1 = \infty \wedge m = m_0) \vee (m_0 = \ominus(m_1) \wedge m = \infty)) \end{aligned}$$

6.3.2 Definition of ccl_1

The *ccl*-function ccl_1 is defined according to the general principle in Sec. 6.2.2. Referring to the above identified canonical operations, we obtain the following definition:

Definition[VSE] 47 (ccl_1 ; TC-AMP):
For a finite message set S and an arbitrary message m we have

$$\begin{aligned} m \in ccl_1(S) \Leftrightarrow (m \in S \vee \\ \exists m_0 : (obj^{\ominus}(m, m_0) \wedge m_0 \in ccl_1(S)) \vee \\ \exists m_0 : (obj^{inv}(m, m_0) \wedge m_0 \in ccl_1(S)) \vee \\ \exists m_0, m_1 : (obj^{pair}(m, m_0, m_1) \wedge (m_0 \in ccl_1(S) \vee m_1 \in ccl_1(S))) \vee \\ \exists m_0, m_1 : (obj^{*}(m, m_0, m_1) \wedge m_1 \in ccl_1(S) \wedge inv(m_0) \notin ccl_1(S)) \vee \\ \exists m_0, m_1 : (obj^{\oplus}(m, m_0, m_1) \wedge m_0 \in ccl_1(S) \wedge \ominus(m_1) \notin ccl_1(S))). \end{aligned}$$

Note that the definition of ccl_1 does not include explicit cases for the non-canonical operations. The integration of derivations by these operations is implicit, which is verified during the proof of the correctness theorem.

Theorem 45 can be easily shown for ccl_1 in TC-AMP, according to the proof schema in Sec. 6.2.2. Consequently, ccl_1 satisfies condition (6.2) required for proof situations where the set of selected secrets (from the induction hypothesis) must be extended with other items (see Sec. 6.1.2).

6.3.3 Appropriate Sets of Selected Secrets

The sets S of selected secrets that shall be used correctly with ccl_1 as basic check-function are obtained according to the guidance instructions in Sec. 6.2.3. Besides the closure property of S implied by the "f-part inclusion" principle, we have to pay attention to additional conditions as a consequence of possible derivations by non-canonical operations (see Sec. 6.3.1):

- S may not include ∞, as ∞ is derivable from every non-empty ik.
- S may not include $\oplus$-objects, because they can be composed by merge operations, where the used items differ from their (implicit) substructures.

In Sec. 9.2, we discuss the basic confidentiality properties of TC-AMP. Properties 90-(1) and respectively (2) are about the confidentiality of message parts sent in the first and the second TC-AMP step, respectively, using a non-compromised password π_A. They are shown with the help of ccl_1 as basic check-function using identical sets S of selected secrets. In our proof by induction, S is initialized in the base case with $\emptyset$ and then composed in the step case by $S_p \uplus E$ for S_p provided by the induction hypothesis and for an appropriate (possibly empty) extension E. As explained in Sec. 6.2.3, the partition of S in S_p and a non-empty E has to satisfy the required extension condition (6.1), i.e.

$$(\forall s \in E : uses(s, \widetilde{m})) \wedge (\forall s \in S_p : \neg uses(s, \widetilde{m})),$$

for some message part $\widetilde{m}$ that is not used in ik, i.e. the set of the observable messages in the induction hypothesis, (but in the last message m). Guided by this requirement, we define the set E according to (the occurrence of π_A in) the protocol steps as follows:

1. The first occurrence of π_A (together with the associated static generators g_1 and g_2) is in some protocol event where a honest participant accesses these initial data, to use them in subsequent protocol steps. Here, S_p is empty and its extension E is set to $\{\pi_A, *(\pi_A, g_1), *(\pi_A, g_2)\}$, where π_A corresponds to the message part $\widetilde{m}$ that does not occur in prior observable messages.
2. Next occurrences of the password π_A in first TC-AMP steps with messages of the form $\oplus(*(nc_1, g_1), \ominus(*(\pi_A, g_2)))$ are accompanied with new nonces nc_1, not used before. This permits us to set $E = \{nc_1, *(nc_1, g_1)\}$ and use nc_1 as $\widetilde{m}$.

3. Further occurrences of π_A are in second TC-AMP steps with "public" message parts of the form $*(nc_2,*(\pi_A,g_1))$, accompanied with new nonces nc_2, not used before. This permits us to include nc_2 and $*(nc_2,g_1)$ in E and to use nc_2 as $\widetilde{m}$.

   Since $*(nc_2,g_1)$ could occur as crypt-parts of some messages that represent accidentally lost session keys of the form $\oplus(*(nc_2,*(nc_1,g_1)),*(m_{(\pi_A,nc_1)},*(nc_2,g_1)))$ for $m_{(\pi_A,nc_1)}$ abbreviating a first TC-AMP message, we add $*(nc_2,*(nc_1,g_1))$ to E (for every $*(nc_1,g_1) \in S_p$), to prevent the derivation of $*(nc_2,g_1)$ from lost session keys. Note that we add $*(nc_2,*(nc_1,g_1))$ already at this stage (before dealing with the oops step modeling the accidental loss of session keys). This is necessary, in order not to violate the above mentioned extension condition (if otherwise we would need to add $*(nc_2,*(nc_1,g_1))$ at the corresponding oops step). The new nonce nc_2, which does not occur neither in ik nor in S_p, is also part of $*(nc_2,*(nc_1,g_1))$.

4. In all other cases, E is set to $\emptyset$.

Note that we extended the set of selected secrets respecting the "decrypt-prevention" principle, but not exactly as described in Sec. 6.2.3. Instead of adding crypt-keys like $\ominus(*(nc_2,*(nc_1,g_1)))$ and respectively $inv(nc_2)$, we simply added $*(nc_2,*(nc_1,g_1))$ and nc_2, respectively. In doing so, we do not change the set $ccl_1(S)$ of the critical messages. This is explained by the fact that $inv(m)$ (resp. $\ominus(m)$) is critical *iff* m is critical (cp. lemma 48).

For $S = S_p \uplus E$, where E is defined as described above, it is clear that S satisfies the above mentioned conditions: It does not include neither ∞ nor any $\oplus$-object and it is closed under the "f-part inclusion" principle.

Using $ccl_1(S_p \uplus E)$, the checks of the regular protocol messages of TC-AMP show that they are not critical. For instance, $\oplus(*(nc_1,g_1),\ominus(*(\pi_A,g_2))) \notin ccl_1(S)$ holds, because $\oplus(*(nc_1,g_1),\ominus(*(\pi_A,g_2))) \notin S$ and $*(nc_1,g_1),*(\pi_A,g_2) \in S$. The latter excludes that $*(nc_1,g_1)$ and $*(\pi_A,g_2)$ (in S) can be derived from $\oplus(*(nc_1,g_1),\ominus(*(\pi_A,g_2)))$.

6.3.4 Correctness of ccl_1 as Basic Check-Function

In this section, we prove that ccl_1 can be used as basic check-function using sets S of selected secrets that satisfy the above fixed conditions. Accordingly, condition $\Re_1(S)$ used in the correctness theorem 46 includes the implications (6.3) (for the "f-part inclusion" principle) together with the following condition, which forbids ∞ and $\oplus$-objects as selected secrets:

$$m \in S \Rightarrow (m \neq \infty \wedge \neg isObj^{\oplus}(m)) \tag{6.4}$$

Using (the extended) $\Re_1(S)$, we describe how the proof of theorem 46 in Sec. 6.2.4 is adapted to the TC-AMP algebra. This consists mainly in adapting the step case according to the possible f-s applied to derive $f(m_0,\ldots,m_{n-1})$ in $ccl_1(S)$ from $m_0,\ldots,m_{n-1}$ in $DYl(ik,n)$. Recall that the proof goal consists in showing $m_i \in ccl_1(S)$ for $0 \leq i < n$, in order to conclude with the induction hypothesis.

- Cases "$f = pair$", "$f = fst$" and "$f = snd$" are mainly identical as in Sec. 6.2.4.
- Cases "$f = h_1$" and "$f = h_2$" are handled based on $\Re_1(S)$ as in the "$f = dh$"-case in Sec. 6.2.4.
- In case "$f = inv$", the axiom about the operations of $inv(m_0)$ provides us with two cases:
  1. $obj^{inv}(inv(m_0),m_0)$: The definition of ccl_1 yields two cases:
     (a) $inv(m_0) \in S$: This case is handled similar to the "$f = dh$"-case.

(b) $m_0 \in ccl_1(S)$: This case is trivial.

2. $obj^{inv}(m_0, m_0')$ and $inv(m_0) = m_0'$: Here, $inv(m_0) \in ccl_1(S)$ implies $m_0' \in ccl_1(S)$. Then, we apply the definition of ccl_1 using $obj^{inv}(m_0, m_0')$ and $m_0' \in ccl_1(S)$ to obtain $m_0 \in ccl_1(S)$.

- Cases "$f = \ominus$", "$f = \oplus$" and "$f = *$" are closed by the lemmata in Sec. 6.3.4.1–6.3.4.3. These lemmata are shown by induction appropriate to the recursion-based cases in the axioms about the operations of $\ominus(m_0)$, $\oplus(m_0, m_1)$ and $*(m_0, m_1)$.

6.3.4.1 Lemma for the $\ominus$-case:

Lemma$^{\text{VSE}}$ 48 (Correctness of $\mathbf{ccl_1}$; $\mathbf{f = \ominus}$-case):
Let m be an arbitrary message, and let S be a finite set of messages such that $\Re_1(S)$ holds. Then we have

$$\ominus(m) \in ccl_1(S) \Leftrightarrow m \in ccl_1(S).$$

Proof: This lemma is proven by induction on the $\oplus$-structure of m:

Base Case: m may not be a $\oplus$-object. That is, case (3) resulting from the axiom about the operations for $\ominus(m)$ where m is a $\oplus$-object is closed by refutation. The complementary cases are handled as follows:

- Case (1), i.e. $m = \infty$ and $\ominus(m) = \infty$: In the left-right case and in the right-left case, we obtain $\infty \in ccl_1(S)$ as assumption, which permits us to conclude by refutation.
- Case (2), i.e. $obj^{\ominus}(m, m_0)$ and $\ominus(m) = m_0$: In the left-right case, we have $m_0 \in ccl_1(S)$. This permits together with $obj^{\ominus}(m, m_0)$ to apply the definition of ccl_1 and to obtain $m \in ccl_1(S)$, as required.

  In the right-left case, we assume $m \in ccl_1(S)$ and want to prove $\ominus(m) \in ccl_1(S)$, i.e. $m_0 \in ccl_1(S)$. Here, the definition of ccl_1 applied to $m \in ccl_1(S)$ yields practically two cases:

  – $m \in S$, i.e. $\ominus(m_0) \in S$: Using $\Re_1(S)$, we obtain $m_0 \in S$ and thus $m_0 \in ccl_1(S)$.
  – One matching case, where $obj^{\ominus}(m, m_0')$ and $m_0' \in ccl_1(S)$ hold, is handled by applying the (dis-)equality axiom for $\ominus$-objects. This yields $m_0 = m_0'$, which permits us to obtain $m_0 \in ccl_1(S)$.
- Case (4), i.e. $obj^{\ominus}(\ominus(m), m)$: In the left-right case, we assume $\ominus(m) \in ccl_1(S)$ and want to prove $m \in ccl_1(S)$. It is shown similar to the right-left case in case (2).

  In the right-left case, we assume $m \in ccl_1(S)$ and want to prove $\ominus(m) \in ccl_1(S)$. It is shown similar to the left-right case in case (2).

Step Case: m must be a $\oplus$-object. That is, cases (1), (2) and (4) resulting from the axiom about the operations for $\ominus(m)$ are closed by refutation. In case (3), we have $obj^{\oplus}(m, m_0, m_1)$ and $obj^{\oplus}(\ominus(m), \ominus(m_0), \ominus(m_1))$.

In the left-right case, we assume $\ominus(m) \in ccl_1(S)$ and want to prove $m \in ccl_1(S)$. For a $\oplus$-object $\ominus(m)$, the definition of ccl_1 applied to $\ominus(m) \in ccl_1(S)$ yields $obj^{\oplus}(\ominus(m), m_0', m_1')$, $m_0' \in ccl_1(S)$ and $\ominus(m_1') \notin ccl_1(S)$. By the induction hypothesis, we get $\ominus(m_0') \in ccl_1(S)$ from $m_0' \in ccl_1(S)$ and $m_1' \notin ccl_1(S)$, i.e. $\ominus(\ominus(m_1')) \notin ccl_1(S)$, from $\ominus(m_1') \notin ccl_1(S)$. In

addition, $obj^{\oplus}(\ominus(m), m_0', m_1')$ implies $obj^{\oplus}(m, \ominus(m_0'), \ominus(m_1'))$. Thus, we have two $\oplus$-parts $\ominus(m_0')$ and $\ominus(m_1')$ of m satisfying the conditions in the matching decrypt-type case of the definition of ccl_1, which permits us to conclude with $m \in ccl_1(S)$.

In the right-left case, we assume $m \in ccl_1(S)$ and want to prove $\ominus(m) \in ccl_1(S)$. For a $\oplus$-object m, Def. 47 applied to $m \in ccl_1(S)$ yields $obj^{\oplus}(m, m_0', m_1')$, $m_0' \in ccl_1(S)$ and $\ominus(m_1') \notin ccl_1(S)$. Similar to the left-right case, we apply the induction hypothesis to obtain $\ominus(m_0') \in ccl_1(S)$ and $\ominus(\ominus(m_1')) \notin ccl_1(S)$. In addition, $obj^{\oplus}(m, m_0', m_1')$ implies $obj^{\oplus}(\ominus(m), \ominus(m_0'), \ominus(m_1'))$. Thus, we have two $\oplus$-parts $\ominus(m_0')$ and $\ominus(m_1')$ of $\ominus(m)$ satisfying the conditions in the matching decrypt-type case of Def. 47, which permits us to conclude with $\ominus(m) \in ccl_1(S)$. □

6.3.4.2 Lemma for the $\oplus$-case:

Lemma$^{\text{VSE}}$ 49 (Correctness of $\mathbf{ccl_1}$; $\mathbf{f = \oplus}$-case):
Let m_0 and m_1 be arbitrary messages, and let S be a finite set of messages such that $\Re_1(S)$ holds. Then we have

$$\oplus(m_0, m_1) \in ccl_1(S) \Rightarrow$$
$$((m_0 \in ccl_1(S) \wedge \ominus(m_1) \notin ccl_1(S)) \vee (m_1 \in ccl_1(S) \wedge \ominus(m_0) \notin ccl_1(S))).$$

Proof: Taking into account lemma 48, we may focus in the proof of lemma 49 on showing $m_i \in ccl_1(S)$ and $m_{1-i} \notin ccl_1(S)$ from $\oplus(m_0, m_1) \in ccl_1(S)$, where $i \in \{0,1\}$. This is proven by induction on the $\oplus$-structures of m_0 and m_1, i.e. $|m_0|_{\oplus} + |m_1|_{\oplus}$, where the induction hypothesis permits us also to obtain $\oplus(m_0, m_1) \notin ccl_1(S)$ from $m_0, m_1 \in ccl_1(S)$ and from $m_0, m_1 \notin ccl_1(S)$.

Base Case: Neither m_0 nor m_1 may be a $\oplus$-object. That is, all cases resulting from the axiom about the operations for $\oplus(m_0, m_1)$ where m_0 or m_1 is a $\oplus$-object are closed by refutation. The complementary cases are handled as follows:

- Case (1), where $m_1 = \infty$ and $\oplus(m_0, m_1) = m_0$ hold, is obvious: The equality permits us to obtain $m_0 \in ccl_1(S)$ from $\oplus(m_0, m_1) \in ccl_1(S)$; According to $\Re_1(S)$, we have $\infty \notin S$. This permits us to employ Def. 47 and obtain $\infty \notin ccl_1(S)$ and thus $m_1 \notin ccl_1(S)$, as required.
- Case (2), where $m_0 = \infty$ and $\oplus(m_0, m_1) = m_1$ hold, is similar to case (1).
- Case (3), where $obj^{\ominus}(m_1, m_0)$ and $\oplus(m_0, m_1) = \infty$ hold, is handled by refutation employing $\infty \notin ccl_1(S)$.
- Case (4), where $obj^{\ominus}(m_0, m_1)$ and $\oplus(m_0, m_1) = \infty$ hold, is similar to case (3).
- In case (11), where $obj^{\oplus}(\oplus(m_0, m_1), m_0, m_1)$ holds, the application of Def. 47 to $\oplus(m_0, m_1) \in ccl_1(S)$ yields mainly two cases:
  - $\oplus(m_0, m_1) \in S$: It is shown by refutation, since S may not include $\oplus$-objects according to $\Re_1(S)$.
  - One matching case, where $obj^{\oplus}(\oplus(m_0, m_1), m_0', m_1')$, $m_0' \in ccl_1(S)$ and the condition $\ominus(m_1') \notin ccl_1(S)$ hold: It is handled applying the (dis-)equality axiom for $\oplus$-objects, which yields five cases closed by refutation to $|m_0|_{\oplus} = 0$ and $|m_1|_{\oplus} = 0$, and the following two cases:

  * $m_0 = m'_0$ and $m_1 = m'_1$: This implies $m_0 \in ccl_1(S)$ and $\ominus(m_1) \notin ccl_1(S)$, as required.
  * $m_0 = m'_1$ and $m_1 = m'_0$: This implies $m_1 \in ccl_1(S)$ and $\ominus(m_0) \notin ccl_1(S)$, as required.

Step Case: m_0 or m_1 is a $\oplus$-object. That is, all cases resulting from the axiom about the operations for $\oplus(m_0, m_1)$ where m_0 and m_1 do not match $\oplus$-objects are closed by refutation. The complementary cases are handled as follows:

- Case (5), where $obj^{\oplus}(m_1, m_3, m_2)$, $obj^{\ominus}(m_3, m_0)$ and $\oplus(m_0, m_1) = m_2$ hold, corresponds to a $dec^{\oplus}_{\oplus}$-operation: We proceed by a case distinction:
  - $m_0 \in ccl_1(S)$: That is, we have $m_3 \in ccl_1(S)$ and $m_2 \in ccl_1(S)$. This permits us to apply the induction hypothesis and obtain $\oplus(m_3, m_2) \notin ccl_1(S)$, i.e. $m_1 \notin ccl_1(S)$, as required.
  - $m_0 \notin ccl_1(S)$: That is, we have $\ominus(m_3) \notin ccl_1(S)$ and $m_2 \in ccl_1(S)$. Thus, we have the required conditions to employ the matching decrypt-type case in Def. 47 and obtain $\oplus(m_3, m_2) \in ccl_1(S)$, i.e. $m_1 \in ccl_1(S)$, as required.
- Case (6), where $obj^{\ominus}(m_0, m_2)$, $obj^{\oplus}(m_1, m_2, m_3)$, and $\oplus(m_0, m_1) = m_3$ hold, is handled similar to case (5).
- Case (7), where $obj^{\oplus}(m_0, m_3, m_2)$, $obj^{\ominus}(m_3, m_1)$ and $\oplus(m_0, m_1) = m_2$ hold, is handled similar to case (5).
- Case (8), where $obj^{\ominus}(m_1, m_2)$, $obj^{\oplus}(m_0, m_2, m_3)$, and $\oplus(m_0, m_1) = m_3$ hold, is handled similar to case (5).
- Case (9), where $obj^{\oplus}(m_0, m_2, m_3)$, $obj^{\oplus}(m_1, m_5, m_4)$, $obj^{\ominus}(m_5, m_2)$ and $\oplus(m_0, m_1) = \oplus(m_3, m_4)$ hold: By the induction hypothesis, we obtain from $\oplus(m_3, m_4) \in ccl_1(S)$ w.l.o.g. $m_3 \in ccl_1(S)$ and $m_4 \notin ccl_1(S)$. We proceed by a case distinction:
  - $m_2 \in ccl_1(S)$: This permits us to apply the induction hypothesis and obtain $\oplus(m_2, m_3) \notin ccl_1(S)$, i.e. $m_0 \notin ccl_1(S)$. In parallel, we apply lemma 48 to obtain $\ominus(m_2) \in ccl_1(S)$, i.e. $m_5 \in ccl_1(S)$, from $m_2 \in ccl_1(S)$ and obtain $\ominus(m_4) \notin ccl_1(S)$ from $m_4 \notin ccl_1(S)$. Thus, we have the required conditions to employ the matching decrypt-type case in Def. 47 and obtain $\oplus(m_5, m_4) \in ccl_1(S)$, i.e. $m_1 \in ccl_1(S)$, as required.
  - $m_2 \notin ccl_1(S)$: This yields $m_5 \notin ccl_1(S)$, which permits us to apply the induction hypothesis and obtain $\oplus(m_5, m_4) \notin ccl_1(S)$, i.e. $m_1 \notin ccl_1(S)$. In parallel, we apply lemma 48 to obtain $\ominus(m_2) \in ccl_1(S)$ from $m_2 \notin ccl_1(S)$. This and consequence $m_3 \in ccl_1(S)$ provide us with the required conditions to employ the matching decrypt-type case in Def. 47 and obtain $\oplus(m_2, m_3) \in ccl_1(S)$, i.e. $m_0 \in ccl_1(S)$, as required.
- Case (10), where $obj^{\oplus}(m_0, m_5, m_3)$, $obj^{\oplus}(m_1, m_2, m_4)$, $obj^{\ominus}(m_5, m_2)$ and $\oplus(m_0, m_1) = \oplus(m_3, m_4)$ hold, is handled similar to case (9).
- Case (11), where $obj^{\oplus}(\oplus(m_0, m_1), m_0, m_1)$ holds, is handled similar to its pendant in the Base Case (see above). Besides two trivial cases (as in the Base Case), resulting by the application of the (dis-)equality axiom for $\oplus$-objects, we obtain the following cases:
  - (iii) $obj^{\oplus}(m_0, m'_0, m)$ and $obj^{\oplus}(m'_1, m_1, m)$: Applying the induction hypothesis to $\ominus(m'_1) \notin ccl_1(S)$, i.e. $m'_1 \notin ccl_1(S)$, we obtain $m_1, m \in ccl_1(S)$ or $m_1, m \notin ccl_1(S)$. We proceed by a case distinction:

  - $m_1, m \in ccl_1(S)$: We want to show $m_0 \notin ccl_1(S)$. This is done applying the induction hypothesis to $m, m'_0 \in ccl_1(S)$, which yields $\oplus(m'_0, m) \notin ccl_1(S)$, i.e. $m_0 \notin ccl_1(S)$.
  - $m_1, m \notin ccl_1(S)$: We want to show $m_0 \in ccl_1(S)$. This is done deducing $\ominus(m) \notin ccl_1(S)$ from $m \notin ccl_1(S)$ and then using $m'_0 \in ccl_1(S)$ to employ the matching decrypt-type case in Def. 47 and obtain $\oplus(m'_0, m) \in ccl_1(S)$, i.e. $m_0 \in ccl_1(S)$.
- (iv) $obj^{\oplus}(m_0, m'_1, m)$ and $obj^{\oplus}(m'_0, m_1, m)$: Applying the induction hypothesis to $m'_0 \in ccl_1(S)$, i.e. $\oplus(m_1, m) \in ccl_1(S)$ yields two cases:
  - $m_1 \in ccl_1(S)$ and $m \notin ccl_1(S)$: We want to show $m_0 \notin ccl_1(S)$. This is done applying the induction hypothesis to $m'_1 \notin ccl_1(S)$ (obtained from the condition $\ominus(m'_1) \notin ccl_1(S)$) and $m \notin ccl_1(S)$, which yields $\oplus(m'_1, m) \notin ccl_1(S)$, i.e. $m_0 \notin ccl_1(S)$.
  - $m_1 \notin ccl_1(S)$ and $m \in ccl_1(S)$: We want to show $m_0 \in ccl_1(S)$. This is done using $m \in ccl_1(S)$ and $\ominus(m'_1) \notin ccl_1(S)$ to employ the matching decrypt-type case in Def. 47 and obtain $\oplus(m'_1, m) \in ccl_1(S)$, i.e. $m_0 \in ccl_1(S)$.
- (v) $obj^{\oplus}(m_1, m'_0, m)$ and $obj^{\oplus}(m'_1, m_0, m)$: It is handled similar to case (iii).
- (vi) $obj^{\oplus}(m_1, m'_1, m)$ and $obj^{\oplus}(m'_0, m_0, m)$: It is handled similar to case (iv).
- (vii) $obj^{\oplus}(m_0, m_2, m_3)$, $obj^{\oplus}(m_1, m_4, m_5)$, $obj^{\oplus}(m'_0, m_2, m_4)$ and $obj^{\oplus}(m'_1, m_3, m_5)$: Applying the induction hypothesis to $m'_1 \notin ccl_1(S)$, i.e. $\oplus(m_3, m_5) \in ccl_1(S)$ yields $m_3, m_5 \in ccl_1(S)$ or $m_3, m_5 \notin ccl_1(S)$. Combined with both cases resulting by the application of the induction hypothesis to the condition $m'_0 \in ccl_1(S)$, i.e. $\oplus(m_2, m_4) \in ccl_1(S)$, we obtain the following cases:
  1. $m_3, m_5, m_2 \in ccl_1(S)$ and $m_4 \notin ccl_1(S)$: By the induction hypothesis, we obtain $\oplus(m_2, m_3) \notin ccl_1(S)$, i.e. $m_0 \notin ccl_1(S)$, and by Def. 47, we obtain $\oplus(m_4, m_5) \in ccl_1(S)$, i.e. $m_1 \in ccl_1(S)$.
  2. $m_3, m_5, m_4 \in ccl_1(S)$ and $m_2 \notin ccl_1(S)$: We replay the steps in (1) to obtain $m_1 \notin ccl_1(S)$ and $m_0 \in ccl_1(S)$.
  3. $m_3, m_5, m_4 \notin ccl_1(S)$ and $m_2 \notin ccl_1(S)$: We replay the steps in (1) to obtain $m_1 \notin ccl_1(S)$ and $m_0 \in ccl_1(S)$.
  4. $m_3, m_5, m_2 \notin ccl_1(S)$ and $m_4 \notin ccl_1(S)$: We replay the steps in (1) to obtain $m_0 \notin ccl_1(S)$ and $m_1 \in ccl_1(S)$. □

6.3.4.3 Lemma for the $*$-case:

Lemma$^{\text{VSE}}$ 50 (Correctness of $\mathbf{ccl_1}$; $\mathbf{f = *}$-case):
Let m_0 and m_1 be arbitrary messages, and let S be a finite set of messages such that $\Re_1(S)$ holds. Then we have

$$m_0 \notin ccl_1(S) \Rightarrow (*(m_0, m_1) \in ccl_1(S) \Leftrightarrow m_1 \in ccl_1(S)).$$

Proof:
The proof is by induction on the $\oplus$-structure of m_1.

Base Case: m_1 may not be a $\oplus$-object. Since we need to reason on the $*$-parts of m_1, we proceed here by induction on the $*$-structure of m_1, i.e. $|m_1|_*$.

(Nested) Base Case: $|m_1|_* = 0$, i.e. m_1 is neither a $*$-object nor a $\ominus$-object having a $*$-object as a synth-part. The application of the axiom about the operations for $*(m_0, m_1)$ yields the following cases:

- Case (1), where $obj^*(m_1, m_3, m_2)$, $obj^{inv}(m_3, m_0)$ and $*(m_0, m_1) = m_2$ hold, is closed by refutation, as m_1 may not be a $*$-object.
- Case (2), where $obj^{inv}(m_0, m_2)$, $obj^*(m_1, m_2, m_3)$, and $*(m_0, m_1) = m_3$ hold, is closed by refutation, as m_1 may not be a $*$-object.
- Case (3), where $obj^{\oplus}(m_1, m_2, m_3)$ and $obj^{\oplus}(*(m_0, m_1), *(m_0, m_2), *(m_0, m_3))$ hold, is closed by refutation, as m_1 may not be a $\oplus$-object.
- Case (4), where $m_1 = \infty$, and $*(m_0, m_1) = \infty$ hold, is closed by refutation based on $\infty \notin ccl_1(S)$.
- Case (6), i.e. $obj^*(*(m_0, m_1), m_0, m_1)$: In the left-right case, the application of Def. 47 to $*(m_0, m_1) \in ccl_1(S)$ yields mainly two cases:
  - $*(m_0, m_1) \in S$: This implies $m_0 \in S$ or $m_1 \in S$, according to $\Re_1(S)$. Due to the condition $m_0 \notin ccl_1(S)$, we obtain $m_1 \in S$ and thus $m_1 \in ccl_1(S)$, as required.
  - The matching decrypt-type case, where $obj^*(*(m_0, m_1), m_0', m_1')$, $m_1' \in ccl_1(S)$ and $inv(m_0') \notin ccl_1(S)$ hold, is handled by applying the (dis-)equality axiom for $*$-objects. This yields the injection case, i.e. $m_0 = m_0'$ and $m_1 = m_1'$, and a second case where m_1 is a $*$-object. The former case is closed immediately using the condition $m_1' \in ccl_1(S)$ and $m_1 = m_1'$, and the latter case is closed by refutation using $|m_1|_* = 0$.

  The right-left case is proven using $m_1 \in ccl_1(S)$ and $inv(m_0) \notin ccl_1(S)$ (a straightforwardly provable consequence of $m_0 \notin ccl_1(S)$) to employ the matching decrypt-type case in Def. 47 and obtain $*(m_0, m_1) \in ccl_1(S)$.
- Case (5), where $obj^{\ominus}(m_1, m_2)$ and $obj^{\ominus}(*(m_0, m_1), *(m_0, m_2))$ hold: Applying the axiom about the operations for $*(m_0, m_2)$ yields in this situation a sole case, i.e. $obj^*(*(m_0, m_2), m_0, m_2)$. This case is closed the same way as case (6), after using lemma 48 to handle $\ominus(m_2)$, i.e. m_1, and respectively $\ominus(*(m_0, m_2))$, i.e. $*(m_0, m_1)$, like m_2 and $*(m_0, m_2)$, respectively.

(Nested) Step Case: $|m_1|_* > 0$, i.e. m_1 is a $*$-object or a $\ominus$-object having a $*$-object as its $\ominus$-part. Cases (3) and (4), resulting from the axiom about the operations for $*(m_0, m_1)$ are closed by refutation, and the complementary cases are handled as follows:

- Case (1), where $obj^*(m_1, m_3, m_2)$, $obj^{inv}(m_3, m_0)$ and $*(m_0, m_1) = m_2$ hold, corresponds to a dec_*^*-operation: The left-right case is proven using $m_2 \in ccl_1(S)$ and $inv(m_3) \notin ccl_1(S)$ (a consequence of $m_0 \notin ccl_1(S)$, $m_3 = inv(m_0)$ and $inv(m_3) = inv(inv(m_0)) = m_0$) to employ the matching decrypt-type case in Def. 47 and obtain $*(m_3, m_2) \in ccl_1(S)$, i.e. $m_1 \in ccl_1(S)$.

  In the right-left case, the application of Def. 47 to the condition $m_1 \in ccl_1(S)$, i.e. $*(m_3, m_2) \in ccl_1(S)$, yields practically two cases:
  - $m_1 \in S$: This implies $m_3 \in S$ or $m_2 \in S$, according to $\Re_1(S)$. Due to $m_0 \notin ccl_1(S)$, i.e. $m_3 \notin ccl_1(S)$, we obtain $m_3 \notin S$ and thus $m_2 \in S$. This implies $m_2 \in ccl_1(S)$, i.e. $*(m_0, m_1) \in ccl_1(S)$, as required.
  - The matching decrypt-type case, where $obj^*(m_1, m_0', m_1')$, $m_1' \in ccl_1(S)$ and $inv(m_0') \notin ccl_1(S)$ hold, is handled by applying the (dis-)equality axiom for $*$-objects, yielding two cases:
    - $m_3 = m_0'$ and $m_2 = m_1'$: Using $m_1' \in ccl_1(S)$, we obtain immediately the consequence $m_2 \in ccl_1(S)$ and thus $*(m_0, m_1) \in ccl_1(S)$, as required.

  - $obj^*(m_2, m_0', m)$ and $obj^*(m_1', m_3, m)$: Using $m_3 \notin ccl_1(S)$ (as a consequence of $m_0 \notin ccl_1(S)$), we apply the induction hypothesis to $m_1' \in ccl_1(S)$, i.e. $*(m_3, m) \in ccl_1(S)$, to obtain $m \in ccl_1(S)$. This permits us together with $inv(m_0') \notin ccl_1(S)$ to employ the matching decrypt-type case in Def. 47 and obtain $*(m_0', m) \in ccl_1(S)$, i.e. $m_2 \in ccl_1(S)$, and thus $*(m_0, m_1) \in ccl_1(S)$, as required.

- Case (2), where $obj^{inv}(m_0, m_2)$, $obj^*(m_1, m_2, m_3)$, and $*(m_0, m_1) = m_3$ hold, corresponds to a dec_*^*-operation: It is handled replaying the proof steps of case (1) and using $m_2 \notin ccl_1(S)$ as a straightforwardly provable consequence of $m_0 \notin ccl_1(S)$.

- Case (6), where $obj^*(*(m_0, m_1), m_0, m_1)$ holds: The left-right case is proven similar to the same case in the "(Nested) Base Case", except of the matching decrypt-type case resulting by the application of Def. 47 to $*(m_0, m_1) \in ccl_1(S)$. In the proof situation, where $obj^*(*(m_0, m_1), m_0', m_1')$, $m_1' \in ccl_1(S)$ and $inv(m_0') \notin ccl_1(S)$ hold, the application of the (dis-)equality axiom for $*$-objects yields the equally handled injection case and a complementary case requiring the application of the induction hypothesis. In this complementary case, where $obj^*(m_1, m_0', m)$ and $obj^*(m_1', m_0, m)$ hold, we obtain first $m \in ccl_1(S)$ from $m_0 \notin ccl_1(S)$ and the consequence $m_1' \in ccl_1(S)$, by the induction hypothesis. Then, we use $obj^*(m_1, m_0', m)$, $m \in ccl_1(S)$ and $inv(m_0') \notin ccl_1(S)$ to employ the matching decrypt-type case in Def. 47 and obtain $m_1 \in ccl_1(S)$, as required.

  The right-left case is proven similar to the same case in the "(Nested) Base Case".

- Case (5), where $obj^{\ominus}(m_1, m_2)$ and $obj^{\ominus}(*(m_0, m_1), *(m_0, m_2))$ hold: Using lemma 48, we handle $m_2 \in ccl_1(S)$ and respectively $*(m_0, m_2) \in ccl_1(S)$ like $m_1 \in ccl_1(S)$, i.e. $\ominus(m_2) \in ccl_1(S)$, and $*(m_0, m_1) \in ccl_1(S)$, i.e. $\ominus(*(m_0, m_2)) \in ccl_1(S)$, respectively. Then, we apply the axiom about the operations for $*(m_0, m_2)$ and obtain (besides trivial cases handled by refutation) the following three cases:

  - $obj^*(m_2, m_3, m_4)$, $obj^{inv}(m_3, m_0)$ and $*(m_0, m_2) = m_4$: It is handled similar to case (1) in the "(Nested) Step Case".
  - $obj^{inv}(m_0, m_3)$, $obj^*(m_2, m_3, m_4)$, and $*(m_0, m_2) = m_4$: It is handled similar to case (2) in the "(Nested) Step Case".
  - $obj^*(*(m_0, m_2), m_0, m_2)$: It is handled similar to case (6) in the "(Nested) Step Case".

Step Case: m_1 must be a $\oplus$-object. Except of case (3), all cases resulting from the axiom about the operations for $*(m_0, m_1)$ yield to m_1 differing from $\oplus$-objects and are thus closed by refutation.

In case (3), we have $obj^{\oplus}(m_1, m_2, m_3)$ and $obj^{\oplus}(*(m_0, m_1), *(m_0, m_2), *(m_0, m_3))$. That is, the left-right case turns into showing $m_1 \in ccl_1(S)$ from $\oplus(*(m_0, m_2), *(m_0, m_3)) \in ccl_1(S)$ and $m_0 \notin ccl_1(S)$. Using lemma 49, we obtain two cases:

- $*(m_0, m_2) \in ccl_1(S)$ and $\ominus(*(m_0, m_3)) \notin ccl_1(S)$: Applying lemma 48 and the induction hypothesis, we obtain $m_2 \in ccl_1(S)$ and $\ominus(m_3) \notin ccl_1(S)$. This permits us to employ the matching decrypt-type case in Def. 47 and obtain $\oplus(m_2, m_3) \in ccl_1(S)$, i.e. $m_1 \in ccl_1(S)$, as required.

- $*(m_0, m_3) \in ccl_1(S)$ and $\ominus(*(m_0, m_2)) \notin ccl_1(S)$: It is handled similar to the previous case.

The right-left case turns into showing $*(m_0, m_1) \in ccl_1(S)$ from $\oplus(m_2, m_3) \in ccl_1(S)$ and $m_0 \notin ccl_1(S)$. Using lemma 49, we obtain two cases:

- $m_2 \in ccl_1(S)$ and $\ominus(m_3) \notin ccl_1(S)$: Applying lemma 48 and the induction hypothesis, we obtain $*(m_0,m_2) \in ccl_1(S)$ and $\ominus(*(m_0,m_3)) \notin ccl_1(S)$. This permits us to employ the matching decrypt-type case in Def. 47 and obtain the consequence $\oplus(*(m_0,m_2),*(m_0,m_3)) \in ccl_1(S)$, i.e. $*(m_0,m_1) \in ccl_1(S)$, as required.
- $m_3 \in ccl_1(S)$ and $\ominus(m_2) \notin ccl_1(S)$: It is handled similar to the previous case. □

6.4 Check-Function ccl_2 for Reduction to Substructures

Targeted secrets s occurring in protocol verification tasks are not restricted to protected message parts inside protocol messages. It is often needed to prove that the attacker is not able to derive *further* targeted secrets s, which are clearly non-atomic. This kind of proof tasks can be handled with the help of basic check-functions, if appropriate sets S of selected secrets can be provided without the need of auxiliary notions. Otherwise, e.g., when only a partially known structure of the targeted secret s is given, we use a second canonical *ccl*-function ccl_2 to reduce the protection of s to the protection of message parts required for its derivation by composition.

6.4.1 Use and Definition of the *ccl*-Function ccl_2

We use the check-function ccl_2 to test whether a targeted secret s is derivable by *extraction* (*en-bloc*) from immediately observable messages in ik. So, this check-function is defined relative to a given s only and that by a *straightforward* adaptation of ccl_1, where additional items in derivations are neglected. By $ik \cap ccl_2(s) = \emptyset$ we check that s is not derivable by *extraction* from all messages in ik, so that s can be derived only by *composition*. In such proof situations, generic theorems are used to reduce s to the message parts required for a composition of s, according to its top structure.

The slight modification of ccl_1 yields to the following definition of ccl_2 in PACE:

Definition$^{\text{VSE}}$ 51 (ccl_2 ; PACE):
For a given message s and an arbitrary message m we have

$$\begin{aligned} m \in ccl_2(s) \Leftrightarrow (m = s \vee \\ \exists m_0,m_1 : (obj^{pair}(m,m_0,m_1) \wedge (m_0 \in ccl_2(s) \vee m_1 \in ccl_2(s))) \vee \\ \exists m_0,m_1 : (obj^{enc}(m,m_0,m_1) \wedge m_1 \in ccl_2(s)) \vee \\ \exists m_0,m_1 : (obj^{dec}(m,m_0,m_1) \wedge m_1 \in ccl_2(s))). \end{aligned}$$

Analogously, we obtain the following definition of ccl_2 in TC-AMP:

Definition$^{\text{VSE}}$ 52 (ccl_2 ; TC-AMP):
For a given message s and an arbitrary message m we have

$$\begin{aligned} m \in ccl_2(s) \Leftrightarrow (m = s \vee \\ \exists m_0 : (obj^{\ominus}(m,m_0) \wedge m_0 \in ccl_2(s)) \vee \\ \exists m_0 : (obj^{inv}(m,m_0) \wedge m_0 \in ccl_2(s)) \vee \\ \exists m_0,m_1 : (obj^{pair}(m,m_0,m_1) \wedge (m_0 \in ccl_2(s) \vee m_1 \in ccl_2(s))) \vee \\ \exists m_0,m_1 : (obj^{*}(m,m_0,m_1) \wedge m_1 \in ccl_2(s)) \vee \\ \exists m_0,m_1 : (obj^{\oplus}(m,m_0,m_1) \wedge m_0 \in ccl_2(s))). \end{aligned}$$

The reduction theorems and ccl_2, which permit for a kind of backward reasoning, do not apply when s can be composed using items different from its substructures. That is, this reduction technique applies in PACE for *all* composed messages, *but* in TC-AMP *only* for the composed messages that differ from $\oplus$-objects.

Principally, reducing the protection of a targeted secret s to the protection of its substructures $s_1,\ldots,s_n$ is based on a generic theorem of the form:

If $ik \cap ccl_2(s) = \emptyset$ and not all $s_1,\ldots,s_n$ occur in $DY(ik)$, then $DY(ik) \cap ccl_2(s) = \emptyset$.

In proof situations where s is assumed to occur (or occurs) in $DY(ik)$, the generic theorem of ccl_2 implies that $ik \cap ccl_2(s) \neq \emptyset$ holds or that $DY(ik)$ includes corresponding occurrences of $s_1,\ldots,s_n$ permitting the composition of s. The reduction to the occurrence of $s_1,\ldots,s_n$ in $DY(ik)$ necessitates to refute $ik \cap ccl_2(s) \neq \emptyset$, which is done in many proof situations with the help of a corresponding proof structuring lemma (a regularity property) of the verified protocol. Such a regularity property implies $ik \cap ccl_2(s) = \emptyset$ for all observable message sets ik given by the protocol that fulfill certain conditions of the corresponding proof situations. It is proven by induction on protocol traces, where $(ik_p \cup \{m\}) \cap ccl_2(s) = \emptyset$ is shown in the step case for each m added by an arbitrary protocol step, using the induction hypothesis $ik_p \cap ccl_2(s) = \emptyset$: In case m originates from a regular protocol step, we simply check $m \notin ccl_2(s)$. But for the fake case, i.e. $m \in DY(ik_p)$, we show $m \notin ccl_2(s)$ with the help of the generic theorem of ccl_2, which requires the additional condition that not all $s_1,\ldots,s_n$ occur in $DY(ik_p)$. For that reason, the employed regularity properties are generally of the form:

> *If* certain conditions on the occurrence of s in ik, e.g., $s \notin ik$ in the majority of authenticity proofs, hold *and* not all $s_1,\ldots,s_n$ occur in $DY(ik)$, *then* we have $ik \cap ccl_2(s) = \emptyset$.

The function ccl_2 and corresponding generic theorem of $ccl_2(s)$ are also employed to reason on the binding of arbitrary message parts of s due to other confidential message parts in s, as given by the protocol context. This could necessitate induction on (the arbitrary message parts given by) the top-structure of s, where the application of the corresponding induction hypothesis is prepared by a suitable application of the generic theorem of $ccl_2(s)$. Concrete examples are in Chap. 8 and 9.

In the following, we describe the typical types of the generic theorems of $ccl_2(s)$, according to the structure of s, and their use in some proofs of PACE and TC-AMP.

6.4.2 Generic Theorem of Type 1

In reality, the proof technique using ccl_2 is applicable not only for composed targeted secrets but also for *atomic* targeted secrets, provided they do not occur in regular protocol messages neither in clear-text, nor as a select-part, nor as a crypt-part. Such confidentiality proofs are straightforward, because we do need to bother about the confidentiality of other message parts, as can be seen in the following correctness theorem of ccl_2.

Theorem^{VSE} 53 (Correctness of $ccl_2(s)$ for $s \in At$):
Let ik be a finite message set, and let s be an *arbitrary atomic* message from At. Then we have

$$ik \cap ccl_2(s) = \emptyset \Rightarrow DY(ik) \cap ccl_2(s) = \emptyset.$$

This theorem is identical in all message algebras. It is shown in the PACE and the TC-AMP algebra by replaying the proof in Sec. 6.2.4 and respectively 6.3.4, where the unnecessary cases are just abstracted away.

Theorem 53 is employed for instance to prove the confidentiality of local secrets in PACE and TC-AMP. In case of TC-AMP, the local confidentiality of the nonce nc_1 and respectively nc_2 generated in the first and the second TC-AMP step, respectively, is required for the forward secrecy of the session key. Here, we prove $nc_1, nc_2 \notin DY(\{\pi_A\} \cup ik)$ just by showing $(\{\pi_A\} \cup ik) \cap ccl_2(nc_1) = \emptyset$ and $(\{\pi_A\} \cup ik) \cap ccl_2(nc_2) = \emptyset$. This holds, since nc_1 and nc_2 occur only as left $*$-parts, which cannot be extracted neither by selector-type nor by decrypt-type operations.

Next, we describe the use of ccl_2 with *composed* targeted secrets.

6.4.3 Generic Theorem of Type 2

In the authenticity lemmas of PACE, which we use to prove the authentication properties 79 and 80, the targeted message $\widehat{s}$ is a *mac*-object ($\widehat{s} = mac(dh(M_4, nc_5), dh(\widehat{g}_A, nc_5))$ in the authentication by the card A and $\widehat{s} = mac(dh(M_5, nc_4), dh(\widehat{g}_B, nc_4))$ in the authentication by the terminal B). In Sec. 8.5 we see that the fake case is handled similarly in both proofs. Referring to the first proof, the assumption that the MAC message could be forged by the attacker, i.e. the assumption that $mac(dh(M_4, nc_5), dh(\widehat{g}_A, nc_5)) \in DY(ik)$ holds, is reduced to the occurrence of the MAC key $dh(M_4, nc_5)$ and the MAC-ed message $dh(\widehat{g}_A, nc_5)$ in the intruder knowledge $DY(ik)$. This necessitates to employ the following generic theorem of ccl_2:

Theorem^{VSE} 54 (Correctness of $\mathbf{ccl_2(mac(m_0, m_1))}$; PACE):
Let ik be a finite message set, and let m_0, m_1 be two *arbitrary* messages. Then we have

$$ik \cap ccl_2(mac(m_0, m_1)) = \emptyset \Rightarrow$$
$$(DY(ik) \cap ccl_2(mac(m_0, m_1)) = \emptyset \vee m_0, m_1 \in DY(ik)).$$

Intuitively, this theorem means that a *mac*-object non-occurring (neither as a crypt-part nor as a select-part) in ik cannot be derived without deriving its *mac*-parts.

The theorem is shown by replaying the proof in Sec. 6.2.4, with slight adaptations. In particular, the inclusion condition as part of $\Re_1(S)$ in theorem 46 is substituted in theorem 54 by the proposition $m_0, m_1 \in DY(ik)$ in the conclusion. The inclusion condition in $\Re_1(S)$ implies that the protection of a *mac*-object necessitates to protect at least one of its *mac*-parts, which is obviously equivalent to the negation of $m_0, m_1 \in DY(ik)$.

The generic theorems about $ccl_2(f(m_0, \ldots, m_{n-1}))$ for all f-objects where f is not permutative are similar to theorem 54. An assumption $obj^f(f(m_0, \ldots, m_{n-1}), m_0, \ldots, m_{n-1})$ is added, when some applications of f are not constructor-type.

In the above mentioned proof situation, theorem 54 is employed together with the regularity property 81 of the PACE protocol, which matches the schema described in Sec. 6.4.1 (see Sec. 8.5).

6.4.4 Generic Theorem of Type 3

The obtained proof situation in Sec. 6.4.3 with an occurrence of the MAC-key $dh(M_4, nc_5)$ and the MAC-ed message $dh(\widehat{g}_A, nc_5)$ in the intruder knowledge $DY(ik)$ is reduced to a

proof situation where M_4 is of the form $dh(\ldots,dh(\widehat{g}_A,x_1),\ldots,x_n)$ for $x_1,\ldots,x_n$ in $DY(ik)$. This is achieved with the help of a binding property of PACE, based on the confidentiality of nc_5 and on the binding of $\widehat{g}_A$ to nc_5 in $dh(\widehat{g}_A,nc_5)$. The binding property on M_4 is shown by nested induction as described in Sec. 8.5, i.e. first on the dh-structure of M_4, i.e. $|M_4|_{dh}$, and then on protocol traces. This includes the reduction of the occurrence of $dh(M_4,nc_5)$ to the occurrence of $dh(M_4',nc_5)$ with $|M_4'|_{dh} < |M_4|_{dh}$, to prepare the application of the corresponding induction hypothesis. The reduction necessitates to employ the following generic theorem of ccl_2:

Theorem$^{\mathrm{VSE}}$ 55 (Correctness of $\mathbf{ccl_2(dh(m_0,m_1))}$; PACE):
Let ik be a finite message set, and let m_0,m_1 be two *arbitrary* messages. Then we have

$$\begin{aligned}&ik \cap ccl_2(dh(m_0,m_1)) = \emptyset \Rightarrow\\ &(DY(ik) \cap ccl_2(dh(m_0,m_1)) = \emptyset \vee\\ &\ (\exists m_0',m_1' : dh(m_0',m_1') = dh(m_0,m_1) \wedge m_0',m_1' \in DY(ik))).\end{aligned}$$

This theorem differs from theorem 54 *only* in the following issue: While a composition of a *mac*-object uses necessarily a sole pair of *mac*-parts, there is basically arbitrary but finitely many pairs of *dh*-parts that permit the composition of a same *dh*-object. All the relevant pairs of the *dh*-parts are covered in theorem 55 by the existentially quantified variables m_0',m_1' in $dh(m_0',m_1') = dh(m_0,m_1)$, due to the (dis-)equality axiom of h-objects.

Such an adaptation for a correctness theorem of $ccl_2(f(m_0,\ldots,m_{n-1}))$ is necessary for all function symbols f that have permutative equations in A.

The conjectured form of M_4 in the above mentioned proof situation permits to deduce based on the occurrence of M_4 in $DY(ik)$ and with the help of a corresponding regularity property of PACE that $\widehat{g}_A$ belongs to $DY(ik)$. The employed regularity property (on the occurrence of $dh(\ldots,dh(\widehat{g}_A,x_1),\ldots,x_n)$ together with $x_1,\ldots,x_n$ in $DY(ik)$) is shown with the help of the correctness theorem 55, too (see Sec. 8.5).

6.4.5 Generic Theorem of Type 4

In the authenticity lemma of TC-AMP, which allows us to prove immediately the authentication property 9.3, the targeted message $\widehat{s}$ as described in Sec. 9.5 is a *pair*-object ($\widehat{s} = pair(M_2,h_1(\widehat{m}_B,M_2,\widehat{k}_B))$ where $\widehat{m}_B = \oplus(*(nc_1,g_1),*(\ominus(\pi_A),g_2))$, M_2 is arbitrary, and $\widehat{k}_B = \oplus(*(inv(\pi_A),*(nc_1,M_2)),*(inv(\pi_A),*(\widehat{m}_B,M_2))))$. In the fake case, the assumption that the targeted *pair*-object could be forged by the attacker, i.e. $pair(M_2,h_1(\widehat{m}_B,M_2,\widehat{k}_B))$ belongs to $DY(ik)$, is reduced in prior steps to the occurrence of M_2 and of the third h_1-part $\widehat{k}_B$ in the intruder knowledge $DY(ik)$. Subsequent proof steps ensure that $\widehat{k}_B$ and respectively M_2 are of the form $\oplus(*(nc_1,\widehat{M}_2),*(\widehat{m}_B,\widehat{M}_2))$ and $*(\pi_A,\widehat{M}_2)$, respectively. $\widehat{M}_2$ is an arbitrary message satisfying $obj^*(*(nc_1,\widehat{M}_2),nc_1,\widehat{M}_2)$ and $obj^*(*(\pi_A,\widehat{M}_2),\pi_A,\widehat{M}_2)$ or is an arbitrary $\oplus$-object $\oplus(x_1,\ldots,\oplus(x_{n-1},x_n))$ where all basic $\oplus$-parts x_i-s satisfy $obj^*(*(nc_1,x_i),nc_1,x_i)$ and $obj^*(*(\pi_A,x_i),\pi_A,x_i)$. These proof steps necessitate to employ the following generic theorem of ccl_2:

Theorem$^{\mathrm{VSE}}$ 56 (Correctness of $\mathbf{ccl_2(*(m_0,m_1))}$; TC-AMP):
Let ik be a finite message set, and let m_0,m_1 be two *arbitrary* messages. Then we have

$$\begin{aligned}&(obj^*(*(m_0,m_1),m_0,m_1) \wedge ik \cap ccl_2(*(m_0,m_1)) = \emptyset) \Rightarrow\\ &(DY(ik) \cap ccl_2(*(m_0,m_1)) = \emptyset \vee\\ &\ (\exists m_0',m_1' : obj^*(*(m_0,m_1),m_0',m_1') \wedge m_0' \in DY(ik) \wedge DY(ik) \cap ccl_2(m_1') \neq \emptyset)).\end{aligned}$$

This theorem differs from theorems 54 and 55 *mainly* in the following issue: The used function symbol "$*$" is not only permutative, but it can be also used in decrypt-type operations dec^*_*, where applications of "$*$" result in crypt-parts of $*$-objects. Here, $DY(ik) \cap ccl_2(*(m_0,m_1)) \neq \emptyset$ for ik satisfying $ik \cap ccl_2(*(m_0,m_1)) = \emptyset$ could be traced back, for instance, to some $*$-object $*(x,*(m_0,m_1))$ that is composed from m_0 and $*(x,m_1)$. This example shows that $m_0,m_1 \in DY(ik)$ does not cover all cases for the composition of some message where $*(m_0,m_1)$ occurs as crypt-part (or select-part). For that reason, $DY(ik) \cap ccl_2(m_1) \neq \emptyset$ is used in theorem 56 instead of $m_1 \in DY(ik)$, which permits to cover all relevant composition cases, including the most simple case, i.e. $m_0,m_1 \in DY(ik)$.

The conjectured form of $\widehat{k}_B$ in the above mentioned proof situation is obtained by refuting the occurrence of $*$-objects $*(inv(\pi_A),m)$ in $\widehat{k}_B$ and respectively $*(inv(nc_1),m)$ in M_2 with a left $*$-part $inv(\pi_A)$ and $inv(nc_1)$, respectively. For that purpose, correctness theorem 56 is employed with corresponding regularity properties of TC-AMP, which matches the schema described in Sec. 6.4.1 (see Sec. 9.5.2).

Having $\widehat{k}_B = \oplus(\oplus(*(nc_1,x_1),\oplus(\ldots,*(nc_1,x_n))),\oplus(*(\widehat{m}_B,x_1),\oplus(\ldots,*(\widehat{m}_B,x_n))))$ in the above proof situation necessitates further reduction steps to close the original fake case (by contradiction). Since $\widehat{k}_B$ is a $\oplus$-object, the reduction technique with the help of ccl_2 and a corresponding generic theorem is not applicable. Instead of that, we need to employ a proof technique permitting to deal with derivations by merging, as described in the next section.

6.5 Dealing with Derivations by Merging

Our proof technique using the canonical *ccl*-functions is relatively simple and intuitive. But, it does not deal with targeted or necessary selected secrets s that are derivable by merging operations, i.e. by composition using items different from their substructures. In our verification of TC-AMP, it was necessary to deal with proof situations where such targeted secrets s, like $\widehat{k}_B = \oplus(\oplus(*(nc_1,x_1),\oplus(\ldots,*(nc_1,x_n))),\oplus(*(\widehat{m}_B,x_1),\oplus(\ldots,*(\widehat{m}_B,x_n))))$ in the above discussed example, are not derivable by extraction from regular protocol messages. This makes it possible to prove their confidentiality with the help of protocol-specific invariants, which permit to delimit the items used in derivations by merging according to the structure of the regular protocol messages (see Sec. 6.5.1).

In another envisaged kind of confidentiality proofs, there can be regular protocol messages where message parts are protected with selected secrets derivable by merging. Here, the canonical *ccl*-function ccl_1 cannot be used as a basic check-function, because the set S of selected secrets does not fit in with the required additional conditions (see Sec. 6.3.4). So, our proposed proof technique would necessitate to come up with a basic check-function that copes with derivations by merging. This issue is discussed in Sec. 6.5.2 with the help of an example protocol (based on the TC-AMP algebra). We present another basic check-function that can be used with $\oplus$-objects as selected secrets and provide a proof sketch for its correctness.

6.5.1 Invariants on Derivations by Merging from Protocol Messages

It was not possible to come up with a generic theorem applicable with ccl_2 to reduce a targeted secret s derivable by merging to message parts required for its composition. Referring to the structure of s only, there are an *unlimited number* of message parts that can be used to derive s by merging. Fortunately, the number of such candidate message parts can be *limited* by following a kind of forward reasoning in protocol-specific invariants about all derivable messages $\hat{m}$ having the same type like s. Such an invariant must link $\hat{m}$ to basic

confidential message parts, so that their occurrence in $\hat{m}$ shall be shown to be inconsistent with that in s, which basically forms the main confidentiality argument of s.

In case of TC-AMP, many protocol properties require to reason on the confidentiality (derivability) of $\oplus$-objects with arbitrary large $\oplus$-structure. For instance, the authenticity of the second message necessitates that the attacker may not derive a $\oplus$-object of the form $\oplus(\oplus(*(nc_1,x_1),\oplus(\ldots,*(nc_1,x_n))),\oplus(*(\widehat{m}_B,x_1),\oplus(\ldots,*(\widehat{m}_B,x_n))))$, which could be used as $\widehat{k}_B$ to fake a valid second message $pair(M_2,h_1(\widehat{m}_B,M_2,\widehat{k}_B))$. The corresponding assumption in the fake case of this authenticity property is reduced to the given form of $\widehat{k}_B$, as described in Sec. 6.4.5. The remaining proof steps (in Sec. 9.5.2) consist in tracing back the occurrence of $\widehat{k}_B$ as a $\oplus$-object to proof situations where structural conditions and basic confidentiality properties can be used to close the fake case by corresponding refutations. This shall be done, as introduced above, with the help of an *invariant* that links every derivable $\hat{m}$ of the same type like the targeted secret (here $\widehat{k}_B$) with *regular* protocol messages where specific message parts of $\hat{m}$ have to originate from. We want to clarify this idea explaining how we obtain the invariants used in the TC-AMP proof.

Basically, every $\oplus$-object $\hat{m}$ with *confidential* basic $\oplus$-parts is *linked* to regular protocol messages: A basic $\oplus$-part of $\hat{m}$ is confidential *only if* it matches or includes the $\oplus$-part of a $\oplus$-object that corresponds to or is extracted from a regular protocol message. Consequently, every derivable $\oplus$-object $\hat{m}$ can be composed from two finite subsets ms_b and ms_p of the intruder knowledge:

- The set ms_b consists of the non-confidential basic $\oplus$-parts of $\hat{m}$.
- The set ms_p consists of the $\oplus$-parts of $\hat{m}$ that are composed from confidential basic $\oplus$-parts and thus *linked* to regular protocol messages.

The elements of ms_p inherit structural conditions and confidential message parts from the linked regular protocol messages. These form the structural and confidentiality conditions of the invariant the confidentiality proof of s is based on (see Sec. 9.5.4 and 9.6.3).

Before we describe such invariants and the corresponding proof schema (in Sec. 6.5.1.2), we introduce the employed short-cuts and *ccl*-function.

6.5.1.1 Used Short-Cuts and *ccl*-Function

The invariants about derivable $\oplus$-objects $\hat{m}$ are formalized with the help of the following short-cuts:

$$syn^*(m,m_0,m_1) \Leftrightarrow (obj^*(m,m_0,m_1) \vee syn^*_{\ominus}(m,m_0,m_1) \vee syn^*_{\oplus}(m,m_0,m_1))$$

$$\begin{aligned} syn^{\overline{*}}(m,ms,x) \Leftrightarrow ((&ms = \emptyset \wedge x = m \wedge m \neq \infty) \vee \\ &(\exists m_0,ms_0 : ms = m_0 \uplus ms_0 \wedge syn^*(*(m_0,x),m_0,x) \wedge \\ &\quad syn^{\overline{*}}(m,ms_0,*(m_0,x)))) \end{aligned}$$

$$\begin{aligned} syn^{\overline{\oplus}}(m,ms) \Leftrightarrow ((&ms = \{m\} \wedge m \neq \infty) \vee \\ &(\exists m_0,m_1,ms_1 : ms = m_0 \uplus ms_1 \wedge obj^{\oplus}(m,m_0,m_1) \wedge syn^{\overline{\oplus}}(m_1,ms_1))) \end{aligned}$$

$syn^{\overline{*}}(m,ms,x)$ permits to identify a successive decomposition of m into the so-called left $*$*-sub-messages* in ms and a corresponding right $*$*-sub-message* x. Similarly, $syn^{\overline{\oplus}}(m,ms)$ permits to identify a successive decomposition of m into $\oplus$-parts in ms.

The invariants about derivable $\oplus$-objects $\hat{m}$ are formalized with the help of the *ccl*-function $ccl_2^{\oplus}$. Instead fixing one arbitrary $\oplus$-object $\hat{m}$, we consider an infinite set $ccl_2^{\oplus}(\hat{m})$ of messages with specific occurrence of a fixed arbitrary $\oplus$-object $\hat{m}$. This is helpful in the step

case (from $DYl(n,.)$ to $DYl(n+1,.)$), as it permits to focus on derivations by composition of $\hat{m}$. Derivations (at level $DYl(n+1,.)$) by extraction of an element in $ccl_2^{\oplus}(\hat{m})$ use m from $ccl_2^{\oplus}(\hat{m})$ (and from level $DYl(n,.)$) and are thus handled for free by the induction hypothesis.

Definition[VSE] 57 ($\mathbf{ccl_2^{\oplus}}$; TC-AMP):
For a given message $\hat{m}$ and an arbitrary message m we have

$$
\begin{aligned}
& m \in ccl_2^{\oplus}(\hat{m}) \Leftrightarrow \\
& ((isObj^{\oplus}(m) \wedge (m = \hat{m} \vee \ominus(m) = \hat{m}) \vee \\
& \quad (\exists m_0, m_1 : obj^{\oplus}(m, m_0, m_1) \wedge \neg isObj^{\oplus}(m_0) \wedge m_0 \in ccl_2^{\oplus}(\hat{m})))) \vee \\
& (\exists m_0, m_1 : obj^{*}(m, m_0, m_1) \wedge m_1 \in ccl_2^{\oplus}(\hat{m})) \vee \\
& (\exists m_0 : obj^{\ominus}(m, m_0) \wedge m_0 \in ccl_2^{\oplus}(\hat{m})) \vee \\
& (\exists m_0 : obj^{inv}(m, m_0) \wedge m_0 \in ccl_2^{\oplus}(\hat{m})) \vee \\
& (\exists m_0, m_1 : obj^{pair}(m, m_0, m_1) \wedge (m_0 \in ccl_2^{\oplus}(\hat{m}) \vee m_1 \in ccl_2^{\oplus}(\hat{m}))))
\end{aligned}
$$

Compared to ccl_2, the recursion case of $ccl_2^{\oplus}$ for $\oplus$-objects m is restricted to their *basic* $\oplus$-parts: If a $\oplus$-object m differs from $\hat{m}$ and $\ominus(\hat{m})$, either $\hat{m}$ is derivable by extraction from a basic $\oplus$-part of m or $\hat{m}$ does not belong to $ccl_2^{\oplus}(\hat{m})$. For instance, we have $\oplus(a, \oplus(b,c))$ is not in $ccl_2^{\oplus}(\oplus(b,c))$, although $\oplus(b,c)$ is derivable by extraction from $\oplus(a, \oplus(b,c))$.

6.5.1.2 Invariant and Proof Schema

In the verification of TC-AMP, we will employ two *similar* invariants about derivable $\oplus$-objects $\hat{m}$: In the invariant used (in Chap. 9) for authenticity proofs, the derivations start with immediately observable message sequences. This must be adapted (in Chap. 12) to prove the resistance against offline guessing, where the derivations start with immediately observable message sequences extended by a candidate password. Both invariants are formalized and proven according to a general schema, which we want to describe in this section. Focusing on the first invariant, we emphasize in the following the general principles and postpone the details to Chap. 9. The second invariant is obtained straightforwardly by simple adaptations, as described in Chap. 12.

First of all, the proof of the invariants by nested induction incorporates a kind of message derivations by forward reasoning. This explains, why the available message parts of $\hat{m}$ (if any) are defined based on successive applications of DYl (DY-level operator).

In particular, the cases where $\hat{m}$ is composed exclusively from confidential basic $\oplus$-parts are linked to regular protocol messages using protocol-specific predicates: In general, $\oplus$-objects $\hat{m}$ can be decomposed in non-confidential basic $\oplus$-parts (in ms_b) and in other $\oplus$-parts (in ms_p) that are themselves composed from confidential basic $\oplus$-parts. The latter are linked (after simplifying the available common $*$-sub-messages) to regular protocol messages with the help of appropriate predicates. In case of TC-AMP, we use (in the first invariant)

- TCAMP_msg_1 for $\oplus$-objects originating from first messages, which is defined by

$$
\begin{aligned}
& \text{TCAMP\_}msg_1(m, ik) \Leftrightarrow \\
& (\exists nc_1, g_1, g_2, pw(j) : \oplus(*(nc_1, g_1), \ominus(*(pw(j), g_2))) \in ik \wedge \\
& \quad g_1 \neq g_2 \wedge nc_1, pw(j), *(nc_1, g_1), *(pw(j), g_2) \notin DY(ik) \wedge \\
& \quad (m = \oplus(*(nc_1, g_1), \ominus(*(pw(j), g_2))) \vee m = \oplus(\ominus(*(nc_1, g_1)), *(pw(j), g_2)))),
\end{aligned}
$$

- TCAMP_$oops$ for $\oplus$-objects originating from oops events,

- and TCAMP_mrg for $\oplus$-objects resulting by merging.

In general, these predicates specify how arbitrary $\oplus$-objects m without non-confidential $\oplus$-parts and $*$-sub-messages originate from regular protocol messages. They are expected to identify not only corresponding structural conditions for m but also their confidential message parts. For a canonical proof of the invariant as described below, these predicates have to satisfy two main requirements: Having $\mathcal{P}case_1,\ldots$ and $\mathcal{P}case_{N_\mathcal{P}}$ as such predicates defined for m relative to ik, the corresponding requirements are:

$\aleph_1$: When $\mathcal{P}case_1(m,ik) \vee \ldots \vee \mathcal{P}case_{N_\mathcal{P}}(m,ik)$ holds and if there is a basic $\oplus$-part m_0 of m satisfying $syn^*(m_0,inv(m_1),m_2)$ for a non-confidential message part m_1, then $\mathcal{P}case_1(*(m_1,m),ik) \vee \ldots \vee \mathcal{P}case_{N_\mathcal{P}}(*(m_1,m),ik)$ must hold, too.

$\aleph_2$: When we have $mrg^{\oplus}_{\oplus}(\oplus(m_0,m_1),m_0,m_1)$, $\mathcal{P}case_1(m_0,ik) \vee \ldots \vee \mathcal{P}case_{N_\mathcal{P}}(m_0,ik)$ and $\mathcal{P}case_1(m_1,ik) \vee \ldots \vee \mathcal{P}case_{N_\mathcal{P}}(m_1,ik)$, then there must be $ms_{01} \subset DY(ik)$ and m_{01} satisfying $syn^{\overline{*}}(\oplus(m_0,m_1),ms_{01},m_{01})$ and $\mathcal{P}case_1(m_{01},ik) \vee \ldots \vee \mathcal{P}case_{N_\mathcal{P}}(m_{01},ik)$.

For example, having $m = \oplus(*(a,b_1),b_2)$ derivable en-bloc from a regular protocol message m' where a (and thus $inv(a)$) is not confidential, then requirement $\aleph_1$ necessitates to link $*(inv(a),m) = \oplus(b_1,*(inv(a),b_2))$ to m', too. Having $m_0 = \oplus(b_0,b)$ and $m_1 = \oplus(b_1,\ominus(b))$ as derivable $\oplus$-objects linked to regular messages, then requirement $\aleph_2$ needs that the used protocol-specific predicates cover $\oplus(m_0,m_1) = \oplus(b_0,b_1)$, too.

In Chap. 9, we explain how the above introduced predicates for TC-AMP satisfy requirements $\aleph_1$ and $\aleph_2$, after providing the corresponding definitions. Using these predicates, we obtain the following invariant about derivable $\oplus$-objects from immediately observable messages of TC-AMP.

Property[VSE] 58 (Derivable $\oplus$-Objects ; TC-AMP):
Let TCAMP be the inductively defined set of TC-AMP traces and let $spies(tr)$ be the message sequence immediately observable by the attacker from a trace tr. Then, we have

$$
\begin{aligned}
&(tr \in \text{TCAMP} \wedge isObj^{\oplus}(\hat{m}) \wedge DY(spies(tr)) \cap ccl_2^{\oplus}(\hat{m}) \neq \emptyset) \Rightarrow \\
&(\exists ms_b, ms_p, n : syn^{\overline{\oplus}}(\hat{m}, ms_b \uplus ms_p) \wedge \\
&\quad (\forall m \in ms_b : \neg isObj^{\oplus}(m) \wedge m \in DYl(spies(tr),n)) \wedge \\
&\quad (\forall m \in ms_p : (\exists m', ms : m' \in DYl(spies(tr),n) \wedge ms \subseteq DYl(spies(tr),n) \wedge \\
&\qquad syn^{\overline{*}}(m,ms,m') \wedge (\text{TCAMP\_}msg_1(m',spies(tr)) \vee \\
&\qquad\quad \text{TCAMP\_}mrg(m',spies(tr)) \vee \text{TCAMP\_}oops(m',spies(tr)))))).
\end{aligned}
$$

In the following, we describe the proof for all similar invariants about $\oplus$-objects $\hat{m}$ derivable from ik. For that purpose, we use $\mathcal{P}case_1,\ldots,\mathcal{P}case_{N_\mathcal{P}}$ for the protocol-specific predicates and we assume that they satisfy requirements $\aleph_1$ and $\aleph_2$. Furthermore, we employ $\Im_1(ms_b,ms_p,n,ik)$ to abbreviate the conditions about ms_b and ms_p, i.e.

1. ms_b consists of the non-confidential basic $\oplus$-parts m of $\hat{m}$ (in $DYl(ik,n)$), and
2. ms_p consists of the $\oplus$-parts m of $\hat{m}$ (occurring in $DYl(ik,n)$) that satisfy $syn^{\overline{*}}(m,ms,m')$ for $ms \subset DYl(ik,n)$ and for m' fulfilling $(\mathcal{P}case_1(m',ik) \vee \ldots \vee \mathcal{P}case_{N_\mathcal{P}}(m',ik))$.

Proof (Schema):
The proof is by nested induction, first on traces tr from TCAMP and then on the extension of DY-knowledge. Like in the first part of this chapter, we use ik_p for $spies(tr)$ and $ik_p \cup \{m\}$ for $spies(ev.tr)$, where m represents the message of the last event ev.

Base Case: The conjecture about the empty trace is trivial, since the corresponding intruder knowledge is empty.

Step Case: In the step case, we need to prove the conjecture for every possible extended trace $ev.tr \in \text{TCAMP}$. The induction hypothesis IH_{tr} states that the conjecture holds for tr, i.e. for ik_p. The proof goal about the extended trace $ev.tr$, i.e. $ik_p \cup \{m\}$, is handled by induction on the extension of DY-knowledge:

(Nested) Base Case: The third assumption is replaced with $(ik_p \cup \{m\}) \cap ccl_2^{\oplus}(\hat{m}) \neq \emptyset$, as $DYl(ik_p \cup \{m\}, 0) = ik_p \cup \{m\}$. This provides us with two cases:

1. $ik_p \cap ccl_2^{\oplus}(\hat{m}) \neq \emptyset$: Here, we apply the induction hypothesis IH_{tr}.
2. $ik_p \cap ccl_2^{\oplus}(\hat{m}) = \emptyset$: In this case, we obtain $m \in ccl_2^{\oplus}(\hat{m})$ for each possible message m that belongs to the last event ev. Here, the proof work consists in establishing the corresponding $\mathcal{P}case_i$-case for the m-s that permit for an en-bloc derivation of $\hat{m}$ composed from confidential basic $\oplus$-parts.

   In case of TC-AMP, only the first step and the oops case provide m that yields such a $\oplus$-object, where m matches $\hat{m}$ (cp. step 1 and the oops-case in the definition of TC-AMP steps, described in Sec. 9.1.2). Here, we confirm that m satisfies TCAMP_msg_1 and respectively TCAMP_oops using protocol conditions and basic confidentiality properties of corresponding message parts.

   In the other regular cases, the structure of m permits to show that it does not belong to $ccl_2^{\oplus}(\hat{m})$.

   In the fake case, m is not structured. But, its membership to $DY(ik_p)$ allows us to handle this case with the help of the induction hypothesis IH_{tr}.

(Nested) Step Case: The obtained induction hypothesis IH_{DYl} provides for all $\oplus$-objects $\hat{m}'$ with $DYl(ik_p \cup \{m\}, n) \cap ccl_2^{\oplus}(\hat{m}') \neq \emptyset$ some $ms'_b, ms'_p, \hat{n}'$ such that $syn^{\overline{\oplus}}(\hat{m}', ms'_b \uplus ms'_p)$ and $\Im_1(ms'_b, ms'_p, \hat{n}', ik)$ hold.

For the (fixed arbitrary) $\oplus$-object $\hat{m}$ with $DYl(ik_p \cup \{m\}, n+1) \cap ccl_2^{\oplus}(\hat{m}) \neq \emptyset$, the proof task consists in providing ms_b, ms_p and $\hat{n}$ and establishing $syn^{\overline{\oplus}}(\hat{m}, ms_b \uplus ms_p)$ and $\Im_1(ms_b, ms_p, \hat{n}, ik)$. Here, we proceed similar to the step case in the proof about the correct use of ccl_1, described in Sec. 6.2.4. Except of the cases

1. $*(m_0, m_1) \in ccl_2^{\oplus}(\hat{m})$ for $m_0, m_1 \in DYl(ik_p \cup \{m\}, n)$
2. and $\oplus(m_0, m_1) \in ccl_2^{\oplus}(\hat{m})$ for $m_0, m_1 \in DYl(ik_p \cup \{m\}, n)$,

all other cases are reduced with the help of definition 57 to proof states that can be immediately closed by the application of IH_{DYl}. For instance, the definition of $ccl_2^{\oplus}$ satisfies $\ominus(m_0) \in ccl_2^{\oplus}(m') \Rightarrow m_0 \in ccl_2^{\oplus}(m')$ for all m_0 and all $\oplus$-objects m'. Thus, we get in case $\ominus(m_0) \in ccl_2^{\oplus}(\hat{m})$ for $m_0 \in DYl(ik_p \cup \{m\}, n)$ immediately $m_0 \in ccl_2^{\oplus}(\hat{m})$. This implies $DYl(ik_p \cup \{m\}, n) \cap ccl_2^{\oplus}(\hat{m}) \neq \emptyset$, which permits to apply IH_{DYl} that provides $syn^{\overline{\oplus}}(\hat{m}, ms'_b \uplus ms'_p)$ and $\Im_1(ms'_b, ms'_p, \hat{n}', ik)$, as required. □

Handling of Case (1): According to definition 57, $*(m_0, m_1) \in ccl_2^{\oplus}(\hat{m})$ holds when

(a) $*(m_0, m_1) = \hat{m}$,

(b) $*(m_0, m_1) = \ominus(\hat{m})$

(c) or when there is a proper substructure m' of $*(m_0,m_1)$ with $m' \in ccl_2^{\oplus}(\hat{m})$.

Case (c) corresponds to a recursive case of $ccl_2^{\oplus}$, where $*(m_0,m_1) \in ccl_2^{\oplus}(\hat{m})$ reduces to $m_1 \in ccl_2^{\oplus}(\hat{m})$. Here, we conclude by the application of IH_{DYl}.

Cases (a) and (b) can be straightforwardly shown equivalent based on the definition of $ccl_2^{\oplus}$. For that reason, we focus on case (a), which we handle by the following lemma.

Lemma[VSE] 59 (Generation of $\oplus$-Objects ; TC-AMP ; $*$-case):
Let $\hat{m}$ be an arbitrary $\oplus$-object and let m_0 and m_1 belong to $DYl(ik,n)$. Then, we have

$$\begin{aligned}&(*(m_0,m_1) = \hat{m} \wedge syn^{\overline{\oplus}}(m_1, ms_b^1 \uplus ms_p^1) \wedge \Im_1(ms_b^1, ms_p^1, \hat{n}_1, ik)) \Rightarrow\\ &\exists ms_b, ms_p, \hat{n} : (syn^{\overline{\oplus}}(\hat{m}, ms_b \uplus ms_p) \wedge \Im_1(ms_b, ms_p, \hat{n}, ik) \wedge\\ &\quad (\forall m : m \in ms_b^1 \uplus ms_p^1 \Leftrightarrow *(m_0,m) \in ms_b \uplus ms_p)).\end{aligned}$$

Proof: The proof is by induction on the length of $ms_b^1 \uplus ms_p^1$.

In the base case, we have $ms_b^1 = \emptyset$, $ms_p^1 = \{m_1\}$ and $isObj^{\oplus}(m_1)$, i.e. m_1 in ms_p^1: Here, $\Im_1(ms_b^1, ms_p^1, \hat{n}_1, ik)$ provides $syn^{\overline{*}}(m_1, ms_1, m_2)$ for $ms_1 \subset DYl(ik,\hat{n}_1)$ and $m_2 \in DYl(ik,\hat{n}_1)$, where $\mathcal{P}case_1(m_2,ik) \vee \ldots \vee \mathcal{P}case_{N_{\mathcal{P}}}(m_2,ik)$ holds. We set $ms_b = \emptyset$ and $ms_p = \{*(m_0,m_1)\}$. We obviously have $syn^{\overline{\oplus}}(\hat{m}, ms_b \uplus ms_p)$ and $m \in ms_p^1 \Leftrightarrow *(m_0,m) \in ms_p$. It remains to show $\Im_1(ms_b, ms_p, \hat{n}, ik)$ in the following case distinction:

1. In case $inv(m_0) \in ms_1$, we have $ms_1 = inv(m_0) \uplus ms_2$ and $syn^{\overline{*}}(*(m_0,m_1), ms_2, m_2)$. This implies $\Im_1(ms_b, ms_p, \hat{n}, ik)$ for $\hat{n} = \hat{n}_1$.
2. Otherwise, if there is a basic $\oplus$-part m_3 of m_2 satisfying $syn^*(m_3, inv(m_0), m_4)$, then we obtain $\mathcal{P}case_1(*(m_0,m_2),ik) \vee \ldots \vee \mathcal{P}case_{N_{\mathcal{P}}}(*(m_0,m_2),ik)$, according to the requirement $\aleph_1$ on the expected definitions of $\mathcal{P}case_i$-s (see above). This permits to get $syn^{\overline{*}}(*(m_0,m_1), ms_1, *(m_0,m_2))$. Hence, there is some $\hat{n}$ with $\Im_1(ms_b, ms_p, \hat{n}, ik)$.
3. If $inv(m_0) \notin ms_1$, i.e. $inv(m_0)$ is not in ms_1, and there is no basic $\oplus$-part m_3 of m_2 satisfying $syn^*(m_3, inv(m_0), m_4)$, then we have $syn^{\overline{*}}(*(m_0,m_1), m_0 \uplus ms_1, m_2)$ and thus $\Im_1(ms_b, ms_p, \hat{n}, ik)$ for some $\hat{n}$.

In the step case, $ms_b^1 \uplus ms_p^1$ includes an arbitrary complete decomposition of m_1 into $\oplus$-parts and we distinguish the following cases:

1. $obj^{\oplus}(m_1, m_2, m_3)$ for arbitrary basic $\oplus$-parts m_2 and m_3, $ms_b^1 = \{m_2, m_3\}$ and $ms_p^1 = \emptyset$: Here, we set $ms_b = \{*(m_0,m_2), *(m_0,m_3)\}$ and $ms_p = \emptyset$.

   Since we have $m_2, m_3 \in DYl(ik,\hat{n}_1)$ and $m_0 \in DYl(ik,n)$, there must be some $\hat{n}$ with $*(m_0,m_2), *(m_0,m_3) \in DYl(ik,\hat{n})$. This implies $\Im_1(ms_b, ms_p, \hat{n}, ik)$. Furthermore, we have $obj^{\oplus}(*(m_0,m_1), *(m_0,m_2), *(m_0,m_3))$, otherwise $obj^{\oplus}(m_1,m_2,m_3)$ would be refuted. Finally, we obviously have $m \in ms_b^1 \Leftrightarrow *(m_0,m) \in ms_b$.
2. $ms_b^1 = m_2 \uplus ms_b^2$ and $m_3 = \overline{\oplus}(ms_b^2 \uplus ms_p^1)$ is a $\oplus$-object: Here, we first deduce $syn^{\overline{\oplus}}(m_3, ms_b^2 \uplus ms_p^1)$ and $\Im_1(ms_b^2, ms_p^1, \hat{n}_1, ik)$ to apply the induction hypothesis. For $\hat{m}' = *(m_0,m_3)$, we obtain $syn^{\overline{\oplus}}(\hat{m}', ms_b' \uplus ms_p')$, $\Im_1(ms_b', ms_p', \hat{n}', ik)$ and the equivalence $m \in ms_b^2 \uplus ms_p^1 \Leftrightarrow *(m_0,m) \in ms_b' \uplus ms_p'$.

   Here, we set $ms_b = *(m_0,m_2) \uplus ms_b'$ and $ms_p = ms_p'$. We obtain immediately $\Im_1(ms_b, ms_p, \hat{n}, ik)$ for some $\hat{n}$, and $m \in ms_b^1 \uplus ms_p^1 \Leftrightarrow *(m_0,m) \in ms_b \uplus ms_p$ follows from $m \in ms_b^2 \uplus ms_p^1 \Leftrightarrow *(m_0,m) \in ms_b' \uplus ms_p'$ and from the equalities $ms_b^1 = m_2 \uplus ms_b^2$

and $ms_b = *(m_0, m_2) \uplus ms'_b$. It remains to show $syn^{\overline{\oplus}}(\oplus(*(m_0, m_2), \hat{m}'), ms_b \uplus ms_p)$ by contradiction: The assumed negation of $syn^{\overline{\oplus}}(\oplus(*(m_0, m_2), \hat{m}'), ms_b \uplus ms_p)$ and $syn^{\overline{\oplus}}(\hat{m}', ms'_b \uplus ms'_p)$ imply that $*(m_0, \ominus(m_2))$ is a basic $\oplus$-part of $\overline{\oplus}(ms'_b \uplus ms'_p)$. Using the equivalence resulting by the application of the induction hypothesis, we deduce that $\ominus(m_2)$ is a basic $\oplus$-part of $\overline{\oplus}(ms^2_b \uplus ms^1_p)$ and this refutes the assumption $syn^{\overline{\oplus}}(m_1, m_2 \uplus ms^2_b \uplus ms^1_p)$.

3. $ms^1_b = \emptyset$, $ms^1_p = m_2 \uplus ms^2_p$ and $m_3 = \overline{\oplus}(ms^2_p)$ is a $\oplus$-object: Here, we first deduce $syn^{\overline{\oplus}}(m_3, ms^1_b \uplus ms^2_p)$ and $\Im_1(ms^1_b, ms^2_p, \hat{n}_1, ik)$ to apply the induction hypothesis. For $\hat{m}' = *(m_0, m_3)$, we obtain $syn^{\overline{\oplus}}(\hat{m}', ms'_b \uplus ms'_p)$, $\Im_1(ms'_b, ms'_p, \hat{n}', ik)$ and the equivalence $m \in ms^1_b \uplus ms^2_p \Leftrightarrow *(m_0, m) \in ms'_b \uplus ms'_p$.

   Here, we set $ms_b = ms'_b$ and $ms_p = *(m_0, m_2) \uplus ms'_p$. We obtain immediately $\Im_1(ms_b, ms_p, \hat{n}, ik)$ for some $\hat{n}$, and $m \in ms^1_b \uplus ms^1_p \Leftrightarrow *(m_0, m) \in ms_b \uplus ms_p$ follows from $m \in ms^1_b \uplus ms^2_p \Leftrightarrow *(m_0, m) \in ms'_b \uplus ms'_p$ and from the equalities $ms^1_p = m_2 \uplus ms^2_p$ and $ms_p = *(m_0, m_2) \uplus ms'_p$. It remains to show $syn^{\overline{\oplus}}(\oplus(*(m_0, m_2), \hat{m}'), ms_b \uplus ms_p)$ by contradiction: The assumed negation of $syn^{\overline{\oplus}}(\oplus(*(m_0, m_2), \hat{m}'), ms_b \uplus ms_p)$ and $syn^{\overline{\oplus}}(\hat{m}', ms'_b \uplus ms'_p)$ provide a decomposition of m_2 into two $\oplus$-parts m_4 and m_5 such that $*(m_0, \ominus(m_4))$ is a basic $\oplus$-part of $\overline{\oplus}(ms'_b \uplus ms'_p)$. Using the equivalence resulting by the application of the induction hypothesis, we deduce that $\ominus(m_4)$ is a basic $\oplus$-part of $\overline{\oplus}(ms^1_b \uplus ms^2_p)$ and this refutes $syn^{\overline{\oplus}}(m_1, ms^1_b \uplus \oplus(m_4, m_5) \uplus ms^2_p)$. □

Handling of Case (2): According to definition 57, $\oplus(m_0, m_1) \in ccl^{\oplus}_2(\hat{m})$ holds when

(a) $\oplus(m_0, m_1) = \hat{m}$,

(b) $\oplus(m_0, m_1) = \ominus(\hat{m})$

(c) or when there is a proper substructure m' of $\oplus(m_0, m_1)$ with $m' \in ccl^{\oplus}_2(\hat{m})$.

Case (c) corresponds to a recursive case of $ccl^{\oplus}_2$, where $\oplus(m_0, m_1) \in ccl^{\oplus}_2(\hat{m})$ reduces to $m_0 \in ccl^{\oplus}_2(\hat{m})$ or $m_1 \in ccl^{\oplus}_2(\hat{m})$. This permits to conclude this case by the application of IH_{DYl}.

Cases (a) and (b) can be straightforwardly shown equivalent based on the definition of $ccl^{\oplus}_2$. For that reason, we focus on case (a), which we handle by the following lemma.

Lemma$^{\mathrm{VSE}}$ 60 (Generation of $\oplus$-Objects ; TC-AMP ; $\oplus$-case):
Let $\hat{m}$ be an arbitrary $\oplus$-object and let m_0 and m_1 belong to $DYl(ik, n)$. Then, we have

$$
\begin{aligned}
&(\oplus(m_0, m_1) = \hat{m} \wedge \\
&\quad (\neg isObj^{\oplus}(m_0) \vee (syn^{\overline{\oplus}}(m_0, ms^0_b \uplus ms^0_p) \wedge \Im_1(ms^0_b, ms^0_p, \hat{n}_0, ik))) \wedge \\
&\quad (\neg isObj^{\oplus}(m_1) \vee (syn^{\overline{\oplus}}(m_1, ms^1_b \uplus ms^1_p) \wedge \Im_1(ms^1_b, ms^1_p, \hat{n}_1, ik)))) \Rightarrow \\
&\exists ms_b, ms_p, \hat{n} : (syn^{\overline{\oplus}}(\hat{m}, ms_b \uplus ms_p) \wedge \Im_1(ms_b, ms_p, \hat{n}, ik)).
\end{aligned}
$$

Proof: The proof is by induction on $|m_0|_{\oplus} + |m_1|_{\oplus}$.

In the base case, m_0 and m_1 are basic $\oplus$-parts. This implies $obj^{\oplus}(\hat{m}, m_0, m_1)$, which permits to conclude using $ms_b = \{m_0, m_1\}$, $ms_p = \emptyset$ and $\hat{n} = n$.

In the step case, m_0 or m_1 is a $\oplus$-object and we distinguish the following cases:

1. m_0 is a basic $\oplus$-part, $syn^{\overline{\oplus}}(m_1, ms_b^1 \uplus ms_p^1)$ and $\Im_1(ms_b^1, ms_p^1, \hat{n}_1, ik)$: We distinguish two cases:
   (a) $\ominus(m_0) \in ms_b^1$: That is, $ms_b^1 = \ominus(m_0) \uplus ms_b^2$ and $syn^{\overline{\oplus}}(\oplus(m_0, m_1), ms_b^2 \uplus ms_p^1)$. Here, $\Im_1(ms_b^1, ms_p^1, \hat{n}_1, ik)$ implies $\Im_1(ms_b^2, ms_p^1, \hat{n}_1, ik)$, which permits to conclude by setting $ms_b = ms_b^2$ and $ms_p = ms_p^1$.
   (b) $\ominus(m_0) \notin ms_b^1$: As $\Im_1(ms_b^1, ms_p^1, \hat{n}_1, ik)$ ensures that $\ominus(m_0)$ cannot be a basic $\oplus$-part of any element in ms_p^1, we get $syn^{\overline{\oplus}}(\oplus(m_0, m_1), m_0 \uplus ms_b^1 \uplus ms_p^1)$ and $\Im_1(m_0 \uplus ms_b^1, ms_p^1, \hat{n}, ik)$ for some $\hat{n}$. Hence, we conclude by setting $ms_b = m_0 \uplus ms_b^1$ and $ms_p = ms_p^1$.

2. m_1 is a basic $\oplus$-part, $syn^{\overline{\oplus}}(m_0, ms_b^0 \uplus ms_p^0)$ and $\Im_1(ms_b^0, ms_p^0, \hat{n}_0, ik)$: Here, the proof is similar to case (1).

3. m_0 and m_1 satisfy $syn^{\overline{\oplus}}(m_0, ms_b^0 \uplus ms_p^0)$, $\Im_1(ms_b^0, ms_p^0, \hat{n}_0, ik)$, $syn^{\overline{\oplus}}(m_1, ms_b^1 \uplus ms_p^1)$ and $\Im_1(ms_b^1, ms_p^1, \hat{n}_1, ik)$: We distinguish the following cases:
   (a) $ms_b^0 = m_2 \uplus ms_b^2$, $\overline{\oplus}(ms_b^2 \uplus ms_p^0) = m_3$ and $\oplus(m_1, m_3) = m_4$ for a basic $\oplus$-part m_4: Here, we have $obj^{\oplus}(\oplus(m_0, m_1), m_2, m_4)$ for two basic $\oplus$-parts m_2 and m_4. This permits to proceed as in the base case.
   (b) $ms_b^0 = m_2 \uplus ms_b^2$, $\overline{\oplus}(ms_b^2 \uplus ms_p^0) = m_3$ and $\oplus(m_1, m_3) = \hat{m}'$ for a $\oplus$-object $\hat{m}'$: Here, the induction hypothesis provides us with ms_b', ms_p' and $\hat{n}'$ satisfying $syn^{\overline{\oplus}}(\hat{m}', ms_b' \uplus ms_p')$ and $\Im_1(ms_b', ms_p', \hat{n}', ik)$. Then, we proceed as in case (1) to combine m_2 with $\hat{m}'$ and conclude with $syn^{\overline{\oplus}}(\hat{m}, ms_b \uplus ms_p)$ and $\Im_1(ms_b, ms_p, \hat{n}, ik)$ for some $\hat{n}$.
   (c) $ms_b^0 = \emptyset$, $ms_p^0 = m_2 \uplus ms_p^2$ and $\overline{\oplus}(m_1 \uplus ms_p^2) = \infty$: Here, $\oplus(m_0, m_1) = m_2$ and we conclude by setting $ms_b = \emptyset$ and $ms_p = \{m_2\}$.
   (d) $ms_b^0 = \emptyset$, $ms_p^0 = m_2 \uplus ms_p^2$ and $\overline{\oplus}(m_1 \uplus ms_p^2) = m_3$ for a basic $\oplus$-part m_3: Here, $obj^{\oplus}(\oplus(m_0, m_1), m_3, m_2)$ holds and we conclude by setting $ms_b = \{m_3\}$ and $ms_p = \{m_2\}$.
   (e) $ms_b^0 = \emptyset$, $ms_p^0 = m_2 \uplus ms_p^2$ and $\overline{\oplus}(m_1 \uplus ms_p^2) = \hat{m}'$ for a $\oplus$-object $\hat{m}'$: If $ms_p^2 \neq \emptyset$, we have $\hat{m}' = \oplus(\overline{\oplus}(ms_p^2), m_1)$ and the induction hypothesis provides us with ms_b', ms_p' and $\hat{n}'$ satisfying $syn^{\overline{\oplus}}(\hat{m}', ms_b' \uplus ms_p')$ and $\Im_1(ms_b', ms_p', \hat{n}', ik)$. Otherwise, we have $\hat{m}' = m_1$ and we continue the proof by setting $ms_b' = ms_b^1$ and $ms_p' = ms_p^1$. The structure of $\hat{m} = \oplus(m_2, \hat{m}')$ is obtained by combining m_2 with the structure of $\hat{m}'$ in the following case distinction:
      i. $syn^{\overline{\oplus}}(\oplus(m_2, \hat{m}'), ms_b' \uplus m_2 \uplus ms_p')$: Here, we conclude by setting $ms_b = ms_b'$ and $ms_p = m_2 \uplus ms_p'$.
      ii. $\ominus(m_2) \in ms_p'$: Here, we have $ms_p' = \ominus(m_2) \uplus ms_p^3$, $syn^{\overline{\oplus}}(\oplus(m_0, m_1), ms_b', ms_p^3)$ and $\Im_1(ms_b', ms_p^3, \hat{n}', ik)$. Hence, we conclude by setting $ms_b = ms_b'$ and $ms_p = ms_p^3$.
      iii. $\ominus(m_2) \notin ms_p'$ and there is $m_3 \in ms_p'$ with $mrg_{\oplus}^{\oplus}(\oplus(m_2, m_3), m_2, m_3)$: Based on $\Im_1(ms_b^0, ms_p^0, \hat{n}_0, ik)$ and $\Im_1(ms_b', ms_p', \hat{n}', ik)$, we obtain $syn^{\overline{*}}(m_2, ms_2, m_4)$, $ms_2 \subset DYl(ik, n_2)$, $\mathcal{P}case_1(m_4, ik) \vee \ldots \vee \mathcal{P}case_{N_{\mathcal{P}}}(m_4, ik)$, $syn^{\overline{*}}(m_3, ms_3, m_5)$, $ms_3 \subset DYl(ik, n_3)$, and $\mathcal{P}case_1(m_5, ik) \vee \ldots \vee \mathcal{P}case_{N_{\mathcal{P}}}(m_5, ik)$.
      W.l.o.g., $mrg_{\oplus}^{\oplus}(\oplus(m_2, m_3), m_2, m_3)$, $syn^{\overline{*}}(m_2, ms_2, m_4)$ and $syn^{\overline{*}}(m_3, ms_3, m_5)$ imply $ms_2 = ms_{23} \uplus ms_2'$, $ms_3 = ms_{23} \uplus ms_3'$, $obj^{\oplus}(m_4, m_{24}, \overline{*}(ms_3', m_{45}))$ and

$obj^{\oplus}(m_5, m_{35}, \ominus(\overline{*}(ms'_2, m_{45})))$. We are interested in the messages $m'_4 = \oplus(\overline{*}(\overline{inv}(ms'_3), m_{24}), m_{45})$ and $m'_5 = \oplus(\overline{*}(\overline{inv}(ms'_2), m_{35}), \ominus(m_{45}))$, because tey clearly satisfy $mrg^{\oplus}_{\oplus}(\oplus(m'_4, m'_5), m'_4, m'_5)$. Furthermore, $\aleph_1$ permits to use $ms_3 \subset DYl(ik, n_3)$ and $ms_2 \subset DYl(ik, n_2)$ to obtain

$$\begin{aligned}
&\mathcal{P}case_1(m'_4, ik) \vee \ldots \vee \mathcal{P}case_{N_{\mathcal{P}}}(m'_4, ik) \text{ from} \\
&\mathcal{P}case_1(m_4, ik) \vee \ldots \vee \mathcal{P}case_{N_{\mathcal{P}}}(m_4, ik) \text{ and} \\
&\mathcal{P}case_1(m'_5, ik) \vee \ldots \vee \mathcal{P}case_{N_{\mathcal{P}}}(m'_5, ik) \text{ from} \\
&\mathcal{P}case_1(m_5, ik) \vee \ldots \vee \mathcal{P}case_{N_{\mathcal{P}}}(m_5, ik).
\end{aligned}$$

Now, we have all we need to use $\aleph_2$ and obtain $ms'_{45} \subset DYl(ik, n_{45})$ and m'_{45} satisfying $syn^{\overline{*}}(\oplus(m'_4, m'_5), ms'_{45}, m'_{45})$ and $\mathcal{P}case_1(m'_{45}, ik) \vee \ldots \vee \mathcal{P}case_{N_{\mathcal{P}}}(m'_{45}, ik)$.
Additionally, the equality $\oplus(m_2, m_3) = \overline{*}(ms_{23} \uplus ms'_2 \uplus ms'_3 \uplus ms'_{45}, m'_{45})$ follows from the above transformations and equalities. This allows us to get $ms'_{23} \subset DYl(ik, n_{23})$ satisfying $syn^{\overline{*}}(\oplus(m_2, m_3), ms'_{23}, m'_{45})$ and then $\Im_1(\emptyset, \{m_{23}\}, n_{23}, ik)$ for $m_{23} = \oplus(m_2, m_3)$.
Based on $\oplus(m_0, m_1) = \oplus(m_{23}, \overline{\oplus}(ms'_b, ms^3_p))$ for $ms'_p = m_3 \uplus ms^3_p$, we proceed according to the content of ms^3_p:

- If $ms^3_p = \emptyset$, we conclude by setting $ms_b = ms'_b$ and $ms_p = \{m_{23}\}$.
- Otherwise, $\Im_1(ms'_b, ms'_p, \hat{n}', ik)$ implies $\Im_1(ms'_b, ms^3_p, \hat{n}^3, ik)$ and we have $\Im_1(\emptyset, \{m_{23}\}, n_{23}, ik)$ with $|\overline{\oplus}(ms'_b \uplus ms^3_p)|_{\oplus} + |m_{23}|_{\oplus} < |m_0|_{\oplus} + |m_1|_{\oplus}$. This permits to conclude with the help of the induction hypothesis. □

6.5.1.3 Usage Example

In this section, we sketch how the invariant 58 permits to show that the attacker is not able to derive a composed message

$$\widehat{k}_B = \oplus(\oplus(*(nc_1, x_1), \oplus(\ldots, *(nc_1, x_n))), \oplus(*(\widehat{m}_B, x_1), \oplus(\ldots, *(\widehat{m}_B, x_n)))).$$

Recall that nc_1 and $\widehat{m}_B$ are left $*$-parts of $*(nc_1, x_i)$ and respectively $*(\widehat{m}_B, x_i)$ for $1 \leq i \leq n$ and that $\widehat{m}_B$ equals $\oplus(*(nc_1, g_1), \ominus(*(\pi, g_2)))$.

Assuming that $\widehat{k}_B$ belongs to $DY(ik)$, the invariant 58 yields $syn^{\overline{\oplus}}(\widehat{k}_B, ms_b \uplus ms_p)$ with $\Im_1(ms_b, ms_p, \hat{n}, ik)$ for some ms_b, ms_p and $\hat{n}$. This permits to distinguish the following cases:

- A basic $\oplus$-part $*(nc_1, x_i)$ of $\widehat{k}_B$ belongs to ms_b and thus to $DY(ik)$: This assumption permits to deduce $nc_1 \in DY(ik)$ or $*(nc_1, g_1) \in DY(ik)$, and both cases are refuted by the confidentiality of these message parts shown with the help of ccl_1 as a basic check-function.

- The complementary case yields that $\widehat{k}_B$ matches a $\oplus$-object $\oplus(*(nc_1, x_1), m_1)$ or has this as a $\oplus$-part in ms_p: This $\oplus$-object satisfies $syn^{\overline{*}}(\oplus(*(nc_1, x_1), m_1), ms_1, m_2)$ for $ms_1 \subset DY(ik)$ and for m_2 fulfilling the protocol-specific cases. Here, we focus on case TCAMP_$msg_1(m_2, ik)$, where we obtain $m_2 = \oplus(*(nc_1, g_1), \ominus(*(\pi, g_2)))$, $*(nc_1, x_1) = \overline{*}(ms_1, *(nc_1, g_1))$ and thus $m_1 = \ominus(\overline{*}(ms_1, *(\pi, g_2)))$ for $g_1 \neq g_2$.

  In case $\widehat{k}_B = \oplus(*(nc_1, x_1), m_1)$, we get $\overline{*}(ms_1, *(\widehat{m}_B, g_1)) = \ominus(\overline{*}(ms_1, *(\pi, g_2)))$, which clearly does not hold.

  In case $\oplus(*(nc_1, x_1), m_1)$ is a $\oplus$-part of $\widehat{k}_B$, we get $\ominus(\overline{*}(ms_1, *(\pi, g_2)))$ as a basic $\oplus$-part of

$$m_3 = \oplus(\oplus(*(nc_1, x_2), \oplus(\ldots, *(nc_1, x_n))), \oplus(\overline{*}(ms_1, *(\widehat{m}_B, g_1)), \oplus(\ldots, *(\widehat{m}_B, x_n)))).$$

This implies that the nonce nc_1 is in ms_1 or this nonce occurs in a basic $\oplus$-part of the form $\ominus(\overline{*}(ms_2,*(nc_1,*(\pi,g_2))))$ for $ms_1 = \widehat{m}_B \uplus ms_2$. The former case is refuted by the confidentiality of nc_1 and the latter case by the binding of nc_1 with g_1.

6.5.2 A Basic Check-Function Dealing with Derivations by Merging

In this section, we demonstrate how our proof technique of basic confidentiality properties can be extended to cope with selected secrets that are derivable by merging operations. For that purpose, we use an example (key establishment) protocol (in Sec. 6.5.2.1) based on the TC-AMP algebra, so that the confidentiality of the session key necessitates to use $\oplus$-objects as selected secrets. In Sec. 6.5.2.2, we propose a *ccl*-function ccl_3 that can be used as a basic check-function for this kind of basic confidentiality proofs. We sketch the corresponding correctness proof in Sec. 6.5.2.3, including the main used lemmata and their detailed proofs.

6.5.2.1 A Running Example

To introduce the basic check-function ccl_3 that permits to deal with $\oplus$-objects as selected secrets, we use the following (factitious) example protocol. It is a key establishment protocol initiated by a (trusted) server *srv*, which selects two agents (from a greater set of participants) to provide them with a session key. The two agents shall use this session key afterwards, e.g., for an anonymous conversation in a certain dialogue game, where the agent in role A shall be the initiator of the conversation and the agent in role B the responder.

To exemplify the different kinds of $\oplus$-objects as selected secrets (see below), we use two kinds of (long-term) shared keys as initial knowledge in this protocol: For each participant ag, there is a shared key $ik(ag)$, which *srv* uses to talk to ag in the role A (initiator of the conversation), and a second shared key $rk(ag)$, which *srv* uses to talk to ag in the role B (responder in the conversation).

1. $srv \longrightarrow A \; : \; \oplus(ik(A),\oplus(N_1,N_2)),\; *(\oplus(N_1,N_2),pair(A,\oplus(N_3,\ominus(N_2))))$
   $\%\; \oplus(ik(A),M_1),\; *(M_1,pair(A,M_2))$
2. $A \longrightarrow srv \; : \; \oplus(ik(A),\oplus(M_1,M_2))\; \%\; \oplus(ik(A),\oplus(N_1,N_3))$
3. $srv \longrightarrow B \; : \; N_4,\; *(\oplus(rk(B),N_4),pair(\oplus(N_1,N_3),B))$
   $\%\; N_4,\; *(\oplus(rk(B),N_4),pair(M_3,B))$

The protocol consists of the following three steps, in the "Alice-and-Bob" notation as described in Sec. 2.1:

- In step 1, the server (*srv*) generates three nonces nc_1, nc_2, nc_3 (for N_1, N_2, N_3) and uses a shared key, e.g., $ik(a)$, for $ik(A)$ to send a message

  $$pair(\oplus(ik(a),\oplus(nc_1,nc_2)),*(\oplus(nc_1,nc_2),pair(a,\oplus(nc_3,\ominus(nc_2)))))$$

  to the owner a of this shared key. This first message permits for instance to ask the receiver, whether she/he is willing to start a conversation with another party using the derivable session key.

  The receiver (in role) A derives M_1 from the first message part by the $dec^{\oplus}_{\oplus}$-operation using $\ominus(ik(A))$ as crypt-key, obtained from the own shared key $ik(A)$. After that, A applies the dec^{*}_{*}-operation to $inv(M_1)$ and to the second message part. Only if the result is a *pair*-message containing (the identity of) A as first entry, the receiver can accept the request of the server *srv* and uses M_1 and the second entry M_2 in the resulting *pair*-message to compute the session key $\oplus(M_1,M_2)$.

- In step 2, A replies with $\oplus(ik(A),\oplus(M_1,M_2))$ computed using the accepted session key and the own shared key $ik(A)$. This permits the server to check a positive answer of A by comparing the replied message with $\oplus(ik(a),\oplus(nc_1,nc_3))$ (for $\oplus(ik(A),\oplus(N_1,N_3))$) that the server computes using the nonces nc_1,nc_3 generated in the first step together with the shared key $ik(a)$ of a.

- In step 3, the server (srv) generates a fourth nonce nc_4 (for N_4) and uses a shared key, e.g., $rk(b)$, for $rk(B)$ to send a message $pair(nc_4,*(\oplus(rk(b),nc_4),pair(\oplus(nc_1,nc_3),b)))$ to the owner b of this shared key. This third message permits for instance to engage the receiver in participating as responder in some conversation by using the derivable session key.

  The receiver (in role) B uses the first message part as a nonce (N_4) and composes this with the own shared key $rk(B)$ to obtain $inv(\oplus(rk(B),N_4))$. After that, B applies the dec^*_*-operation to the obtained inv-object and to the second message part. Only if the result is a $pair$-message containing (the identity of) B as second entry, the receiver will act as responder in a subsequent conversation using the first entry M_3 in the resulting $pair$-message as session key.

A successful run of this protocol yields to a new session key $\oplus(nc_1,nc_3)$ (for $\oplus(N_1,N_3)$), which shall be known only for the server srv and for the chosen agents a and b (in the roles A and respectively B).

Independent of the anonymity aspect, we want to discuss in the following how the confidentiality of the established session key $s = \oplus(nc_1,nc_3)$ can be shown.

First, the techniques described in the previous sections are not applicable, because of the following reasons:

- It is not possible to use the canonical basic check-function ccl_1, because the sets S of selected secrets include $\oplus$-objects (s and further selected secrets), which would violate the necessary condition $\Re_1(S)$ in the correctness theorem.

- Showing the confidentiality of s by reduction with the help of the check-function ccl_2 does not work, because s is a $\oplus$-object.

- Coming up with a protocol-specific invariant providing a complete case distinction about the derivable $\oplus$-objects and using this to show that they differ from s seems complicated without relying on provably confidential basic message parts like $ik(a)$, $rk(b)$ and the used nonces nc_1,nc_2,nc_3: For instance, without knowing whether $\oplus(nc_1,nc_2)$ is confidential we will obtain a case where $\oplus(nc_1,nc_2)$ and $\oplus(nc_3,\ominus(nc_2))$ are public, which permits to derive $\oplus(nc_1,nc_3)$ by merging. Furthermore, the basic $\oplus$-parts $ik(a)$ and nc_1,nc_2,nc_3 occur protected with $\oplus$-objects $\oplus(nc_1,nc_2)$, $\oplus(nc_1,nc_3)$, $\oplus(ik(a),nc_2)$, $\oplus(ik(a),nc_3)$ and $\oplus(ik(a),nc_1)$, and the use of these $\oplus$-objects as selected secrets precludes the application of the proof technique by ccl_1 as a basic check-function.

Consequently, we need to come up with a basic check-function that permits the use of $\oplus$-objects as selected secrets. Before we provide the definition of this check-function (in Sec. 6.5.2.2), we introduce the different kinds of the relevant $\oplus$-objects:

1. Following the f-part inclusion principle and selecting additional secrets to preserve confidentiality protection against derivations by decrypt-type operations (using the above messages of a complete protocol run), we define the set S of the selected secrets to include $s = \oplus(nc_1,nc_3)$, nc_1, $\oplus(ik(a),nc_2)$, $ik(a),nc_2$, $\oplus(ik(a),nc_1)$, $\oplus(nc_1,nc_2)$, $\oplus(ik(a),nc_3)$, nc_3, $\oplus(rk(b),nc_4)$ and $rk(b)$.

   In contrast to $\oplus(ik(a),nc_2)$, $\oplus(ik(a),nc_1)$, $\oplus(nc_1,nc_2)$ and $\oplus(ik(a),nc_3)$, which occur as $\oplus$-parts in public $\oplus$-objects, $\oplus(rk(b),nc_4)$ does not occur as a $\oplus$-part of a public

$\oplus$-object. Moreover, none of the basic $\oplus$-parts of $\oplus(rk(b),nc_4)$ occurs as a $\oplus$-part of a public $\oplus$-object. For that reason, the protection of $\oplus(rk(b),nc_4)$, contrarily to $\oplus(ik(a),nc_2),\ldots,\oplus(ik(a),nc_3)$, can be checked without incorporating derivations by merging. So, we distinguish the protected $\oplus$-parts in S into two sets: $S_\oplus$ consists of the $\oplus$-parts whose protection needs to involve derivations by merging, and $S \setminus S_\oplus$ includes the protected $\oplus$-objects and corresponding $\oplus$-parts whose protection does not necessitate to involve derivations by merging.

2. In Sec. 6.5.1, we have proved that derivations by merging do not violate the confidentiality of a targeted $\oplus$-object s' with the help of a protocol-specific invariant providing a complete case distinction about all derivable $\oplus$-objects m. The provided cases cover in particular *all effective* derivations by merging, and the resulting messages neither match s' nor violate the protection of its $\oplus$-parts. For a similar handling in our new basic check-function, we use a set P including all public $\oplus$-objects that occur (also implicitly) as or in regular protocol messages and that can be involved in derivations by merging relevant for the protection of $\oplus$-objects in S. This set must be *closed under merging* (see below) and disjoint with S.

   For example, P includes $\oplus(ik(a),\oplus(nc_1,nc_2))$, $\oplus(ik(a),\oplus(nc_1,nc_3))$, $\oplus(nc_2,\ominus(nc_3))$ and $\oplus(nc_3,\ominus(nc_2))$. Each of these $\oplus$-objects is not critical for the protection of S, because every $dec^{\oplus}_{\oplus}$-derivation of some $s_i \in S$ from these $\oplus$-objects necessitates a crypt-key corresponding mainly to a selected secret in S.

   Note that the elements in P are allowed to appear as public messages. For that reason, all their $\oplus$-parts must belong to S. More precisely, the subset $S_\oplus$ of S must consist of the $\oplus$-parts of the elements in P (see below).

3. In contrast to the $\oplus$-objects in $S_\oplus$, other $\oplus$-objects in S, e.g., $\oplus(rk(b),nc_4)$, are handled like any other selected secret in S whose protection is not affected by merging derivations. So, it is sufficient to include just $rk(b)$ (without including nc_4) as selected secret. This is in principle permissible only if $\oplus(rk(b),nc_4)$ and its protected $\oplus$-part $rk(b)$ do not occur as (implicit) $\oplus$-parts of public message parts of regular messages.

Summing up, the new basic check-function shall be defined relative to three sets $S, S_\oplus, P$. For a session key $s = \oplus(nc_1,nc_3)$ fixed by a first message to an agent a, the contents of these sets are defined according to the following principles:

- A first protocol message that fixes $s = \oplus(nc_1,nc_3)$ necessitates to include

  1. $s,s_1,s_2,s_3,s_4,nc_1,nc_3,ik(a),nc_2$ in $S_\oplus$, for

     $$s_1 = \oplus(ik(a),nc_2), s_2 = \oplus(ik(a),nc_1), s_3 = \oplus(nc_1,nc_2), s_4 = \oplus(ik(a),nc_3),$$

  2. and p_1,p_2,p_3 in P for

     $$p_1 = \oplus(ik(a),\oplus(nc_1,nc_2)), p_2 = \oplus(ik(a),\oplus(nc_1,nc_3)), p_3 = \oplus(nc_2,\ominus(nc_3)).$$

  In particular, s_4 in $S_\oplus$ (and accordingly p_2 and p_3 in P) must be included at this first protocol step, in order to get a negative check for the message of the corresponding second protocol step. This is required, because the second protocol step does not include new items unused before.

- A first message

  $$pair(\oplus(ik(a),\oplus(nc'_1,nc'_2)),*(\oplus(nc'_1,nc'_2),pair(a,\oplus(nc'_3,\ominus(nc'_2)))))$$

  that includes $ik(a)$ and where the nonces differ from the nonces used at the establishment of s necessitates to include

1. $s_5, s_6, s_7, s_8, nc'_1, nc'_3, nc'_2$, in $S_\oplus$, for
$$s_5 = \oplus(ik(a), nc'_2),\ s_6 = \oplus(ik(a), nc'_1),\ s_7 = \oplus(nc'_1, nc'_2),\ s_8 = \oplus(ik(a), nc'_3),$$

2. $p_4, p_5, p_6, p_7, p_8, p_9, p_{10}$ in P, for
$$\begin{aligned}
&p_4 = \oplus(ik(a), \oplus(nc'_1, nc'_2)),\quad p_5 = \oplus(ik(a), \oplus(nc'_1, nc'_3)),\\
&p_6 = \oplus(nc'_2, \ominus(nc'_3)),\quad p_7 = \oplus(\oplus(nc_1, nc_2), \oplus(\ominus(nc'_1), \ominus(nc'_2))),\\
&p_8 = \oplus(\oplus(nc_1, nc_2), \oplus(\ominus(nc'_1), \ominus(nc'_3))),\\
&p_9 = \oplus(\oplus(nc_1, nc_3), \oplus(\ominus(nc'_1), \ominus(nc'_2))),\\
&p_{10} = \oplus(\oplus(nc_1, nc_3), \oplus(\ominus(nc'_1), \ominus(nc'_3))),
\end{aligned}$$

3. $s_9, s_{10}, s_{11}, s_{12}$ in $S_\oplus$, (as complementary to nc_1 in p_7, p_8, p_9, p_{10}), for
$$\begin{aligned}
&s_9 = \oplus(nc_2, \oplus(\ominus(nc'_1), \ominus(nc'_2))),\quad s_{10} = \oplus(nc_2, \oplus(\ominus(nc'_1), \ominus(nc'_3))),\\
&s_{11} = \oplus(nc_3, \oplus(\ominus(nc'_1), \ominus(nc'_2))),\quad s_{12} = \oplus(nc_3, \oplus(\ominus(nc'_1), \ominus(nc'_3))),
\end{aligned}$$

4. s_{13}, s_{14} in $S_\oplus$, (as complementary to nc_2 (resp. nc_3) in p_7, p_8 (resp. p_9, p_{10})), for
$$s_{13} = \oplus(nc_1, \oplus(\ominus(nc'_1), \ominus(nc'_2))),\ s_{14} = \oplus(nc_1, \oplus(\ominus(nc'_1), \ominus(nc'_3))),$$

5. $s_{15}, s_{16}, s_{17}, s_{18}$ in $S_\oplus$, (as complementary to $\ominus(nc'_1)$ in p_7, p_8, p_9, p_{10}), for
$$\begin{aligned}
&s_{15} = \oplus(\oplus(nc_1, nc_2), \ominus(nc'_2)),\quad s_{16} = \oplus(\oplus(nc_1, nc_2), \ominus(nc'_3)),\\
&s_{17} = \oplus(\oplus(nc_1, nc_3), \ominus(nc'_2)),\quad s_{18} = \oplus(\oplus(nc_1, nc_3), \ominus(nc'_3)),
\end{aligned}$$

6. s_{19}, s_{20} in $S_\oplus$, (as complementary to $\ominus(nc'_2)$ (resp. $\ominus(nc'_3)$) in p_7, p_8 (resp. p_9, p_{10})), for
$$s_{19} = \oplus(\oplus(nc_1, nc_2), \ominus(nc'_1)),\ s_{20} = \oplus(\oplus(nc_1, nc_3), \ominus(nc'_1)),$$

7. and $s_{21}, s_{22}, s_{23}, s_{24}, s_{25}, s_{26}, s_{27}, s_{28}, s_{29}, s_{30}$ in $S_\oplus$ (as all possible $\oplus$-parts m of p_7, p_8, p_9, p_{10} satisfying $|m|_\oplus = 1$), for
$$\begin{aligned}
&s_{22} = \oplus(nc_1, \ominus(nc'_1)),\quad s_{23} = \oplus(nc_1, \ominus(nc'_2)),\quad s_{24} = \oplus(nc_1, \ominus(nc'_3)),\\
&s_{25} = \oplus(nc_2, \ominus(nc'_1)),\quad s_{26} = \oplus(nc_2, \ominus(nc'_2)),\quad s_{27} = \oplus(nc_2, \ominus(nc'_3)),\\
&s_{28} = \oplus(nc_3, \ominus(nc'_1)),\quad s_{29} = \oplus(nc_3, \ominus(nc'_2)),\quad s_{30} = \oplus(nc_3, \ominus(nc'_3)).
\end{aligned}$$

Actually, we intended to ensure that P is closed under merging, e.g., p_7 results by merging p_1 and $\ominus(p_4)$. But, P includes messages like p_1 and p_{10} whose merging yields a new message (here $\oplus(\oplus(ik(a), nc_2), \oplus(\ominus(nc_3), \oplus(nc'_1, nc'_3)))$) that have $\oplus$-parts (here p_5, p_3) in P. That is, P must include only the merging results that do not possess $\oplus$-parts in P (see condition $\Re_2$ below). Merging results like $\oplus(p_5, p_3)$ must not be included in P, because otherwise we would need to include p_5 or p_3 in S and this violates the necessary condition $P \cap S = \emptyset$.

For the correctness proof, it was necessary to further specify the elements in P whose merging results possess $\oplus$-parts in P. This is done by the sole protocol-specific condition $\Re_P$ on the set P:

$$\begin{aligned}
&(obj^\oplus(p_1, u, v) \wedge obj^\oplus(p_2, u, w) \wedge obj^\oplus(\oplus(v, \ominus(w)), p_3, p_4) \wedge p_1, p_2, p_3, p_4 \in P) \Rightarrow\\
&((|u|_\oplus = 0 \wedge |v|_\oplus = 1 \wedge |w|_\oplus = 2 \wedge |p_3|_\oplus = 1 \wedge |p_4|_\oplus = 2) \vee\\
&\ (|u|_\oplus = 1 \wedge |v|_\oplus = 1 \wedge |w|_\oplus = 1 \wedge |p_3|_\oplus = 1 \wedge |p_4|_\oplus = 1))
\end{aligned}$$

$\Re_P$ mainly limits the $\oplus$-structure (number of basic $\oplus$-parts) of the involved elements in P. So, we expect that such a protocol-specific condition can be abandoned when dealing (in other message algebras) with merging results having a fixed top-structure, as opposed to the top-structure of $\oplus$-objects given by arbitrary many basic $\oplus$-parts.

- A third message $pair(nc_4, *(\oplus(rk(b), nc_4), pair(\oplus(nc_1, nc_3), b)))$ where the nonces nc_1, nc_3 belong to $S_\oplus$ necessitates to include $\oplus(rk(b), nc_4)$ and $rk(b)$ in S only (and not in $S_\oplus$).

After sketching the content of the parameter sets S, $S_\oplus$ and P, we describe how they are used in the definition of the basic check-function ccl_3.

6.5.2.2 Basic Check-Function ccl_3

For arbitrary given messages m, the function ccl_3 checks whether m is "critical" for the protection of the selected secrets in S, using the auxiliary sets $S_\oplus$ and P as described in the following.

- Except of $\oplus$-objects m, all other types of messages are checked relative to S as it is done by $ccl_1(S)$.
- In case of a $\oplus$-object m, the checks performed by ccl_3 and the additional conditions on S, $S_\oplus$ and P shall deal with the following derivations.
  - In dealing with $dec^{\oplus}_{\oplus}$-derivations, ccl_3 has to ensure
    1. that no (implicit) $\oplus$-part of m occurs in $S \setminus S_\oplus$, because these selected secrets may not occur as $\oplus$-parts of public $\oplus$-objects,
    2. and that, in presence of an (implicit) $\oplus$-part b of m in $S_\oplus$, a corresponding complementary $\oplus$-part u exists such that $\oplus(b, u)$ is in P and the remaining $\oplus$-part (if any) is not critical.
  - In dealing with $sel^{\ominus}_{\oplus}$-, $syn^{\ominus}_{\oplus}$- and $trs^{\ominus}_{\oplus}$-derivations, ccl_3 has to handle $\oplus$-objects m and $\ominus(m)$ as public variations of each other, as defined below.
  - In dealing with $dec^{*}_{\oplus}$-derivations, ccl_3 has to handle for instance $\oplus(g_1, \ominus(g_2))$ as a public variation of $\oplus(*(c_i, g_1), \ominus(*(c_i, g_2)))$ when c_i is not protected in S.
  - In dealing with derivations by $trs^{*}_{\oplus}$-operations, ccl_3 has to handle for instance $\oplus(g_1, \ominus(*(inv(c_i), g_2)))$ as a public variation of $\oplus(*(c_i, g_1), \ominus(g_2))$ when c_i is not protected in S.
  - Protection violation by $trs^{*}_{\oplus}$-derivations as keyed trans-operations is excluded by a corresponding additional condition, which ensures for instance in presence of $\oplus(*(c_i, g_1), \ominus(g_2))$ in S that either c_i or $\oplus(g_1, \ominus(*(inv(c_i), g_2)))$ must be in S (see $\Re_3$ below).
  - To exclude protection violation by $mrg^{\oplus}_{\oplus}$-derivations, the additional conditions ensure that P and S are disjoint and that P is closed under merging, in the above discussed sense (see $\Re_2$ below and the above introduced condition $\Re_P$).
  - To exclude protection violation by $obj^{\oplus}$-derivations, the additional conditions ensure that for each possible pair of $\oplus$-parts of a $\oplus$-object in S, at least one of these $\oplus$-parts is included in S (see $\Re_1$ below).
  - Protection violation by derivations through $syn^{*}_{\oplus}$-operations is excluded by a corresponding additional condition, which ensures for instance in presence of $\oplus(*(c_i, g_1), \ominus(*(c_i, g_2)))$ in S that c_i or $\oplus(g_1, \ominus(g_2))$ must be in S (see $\Re_4$ below).
- To meet the above introduced principles, the definition of ccl_3 is obtained by the following adaptations of ccl_1:
  - In the base case, we focus on all possible *public variations of m relative to S*.
    The public variations of m (relative to S) are all messages that can result from m by an application of $\ominus$ and/or by successive $dec^{*}_{\oplus}$- and $trs^{*}_{\oplus}$-derivations where

the used crypt-keys and trans-keys m_k satisfy $m_k, inv(m_k) \notin S$. Examples are given above.

The check of the public variations is performed with the help of an auxiliary function $ccl_3^{\oplus}$ relative to $S, S_{\oplus}, P$ (see below). If a public variation of m is in $ccl_3^{\oplus}(S, S_{\oplus}, P)$, the message m is "critical" for the protection of S (in the context given by $S_{\oplus}$ and P).

$ccl_3^{\oplus}$ checks whether the given public variation is in S. In case of a $\oplus$-object, it additionally checks the above introduced requirements in dealing with $dec_{\oplus}^{\oplus}$-derivations.

– In the (standard) recursion case, m is checked exactly as in ccl_1 if m is not a $\oplus$-object. Otherwise, ccl_3 simply checks whether a basic $\oplus$-part of m is critical.

- The auxiliary function $ccl_3^{\oplus}$ is defined by

$$
\begin{aligned}
& m \in ccl_3^{\oplus}(S, S_{\oplus}, P) \Leftrightarrow \\
& (m \in S \vee \\
& \quad (\exists m_0, m_1 : obj^{\oplus}(m, m_0, m_1) \wedge m_0 \in S \wedge m_0 \notin S_{\oplus}) \vee \\
& \quad (\exists b, m_b : obj^{\oplus}(m, b, m_b) \wedge \neg isObj^{\oplus}(b) \wedge b \in S_{\oplus} \wedge m \notin P \wedge \\
& \qquad (\forall u, m_u : obj^{\oplus}(m_b, u, m_u) \Rightarrow \oplus(b, u) \notin P)) \vee \\
& \quad (\exists u, m_u : obj^{\oplus}(m, u, m_u) \wedge u \in P \wedge m_u \in ccl_3^{\oplus}(S, S_{\oplus}, P))).
\end{aligned}
$$

The definition of $ccl_3^{\oplus}$ includes three base cases and one recursive case:

– Case 1: $m \in S$ corresponds to the simple base case.

– Case 2: m has a $\oplus$-part m_0 from $S \setminus S_{\oplus}$, which may not occur as (implicit) $\oplus$-part of a public message part derivable from a regular message.

– Case 3: m has a *basic* $\oplus$-part b from $S_{\oplus}$, m is not in P and the complementary $\oplus$-part m_b does not have any $\oplus$-part u such that $\oplus(b, u) \in P$ holds.

– Case 4: m has two complementary $\oplus$-parts u and m_u such that $u \in P$ and (recursively) $m_u \in ccl_3^{\oplus}(S, S_{\oplus}, P)$ hold.

The recursion in case 4 is wrt. "non-critical" $\oplus$-parts of m, which belong to P.

6.5.2.3 Correctness of ccl_3 as Basic Check-Function

In this section, we prove that ccl_3 can be used as basic check-function with sets S of selected secrets and with sets $S_{\oplus}, P$ about the context of merging derivations. The correct use of ccl_3 relies on the following additional conditions on $S, S_{\oplus}$ and P, as introduced above.

- First, condition $\Re_1$ of ccl_1 is adapted to permit the occurrence of $\oplus$-objects in S. Accordingly, the "f-part inclusion" principle in $\Re_1$ applies also to $\oplus$-objects.

- $\Re_2$ consists of the following conditions about S, $S_{\oplus}$ and P to enforce a consistent context for checking $\oplus$-objects:

  1. $p \in P \Rightarrow (p \notin S_{\oplus} \wedge \ominus(p) \in P \wedge (obj^{\oplus}(p, s_0, s_1) \Rightarrow s_0, s_1 \in S_{\oplus}))$
  2. $(obj^{\oplus}(p_1, u, v) \wedge obj^{\oplus}(p_2, u, w) \wedge v \neq w \wedge p_1, p_2 \in P) \Rightarrow$
     $(\oplus(v, \ominus(w)) \in P \vee (\exists p_3, p_4 \in P : obj^{\oplus}(\oplus(v, \ominus(w)), p_3, p_4)))$
  3. $(s \in S \setminus S_{\oplus} \wedge isObj^{\oplus}(s)) \Rightarrow (obj^{\oplus}(s, u, v) \Rightarrow u, v \notin S_{\oplus})$

  Parts 1 and 3 exclude that $\oplus$-parts of $\oplus$-objects in $S \setminus S_{\oplus}$ occur as $\oplus$-parts of elements in P. They also ensure that P and S are disjoint. Part 2 ensures that P is closed under merging, in the above introduced sense.

- Protection violation by $trs^*_\oplus$-derivations as keyed trans-operations is excluded by a corresponding additional condition $\Re_3$, which ensures for instance in presence of $\oplus(*(c_i,g_1),\ominus(g_2))$ in S that either c_i or $\oplus(g_1,\ominus(*(inv(c_i),g_2)))$ must be in S.
- Protection violation by $syn^*_\oplus$-derivations is excluded by a corresponding additional condition $\Re_4$, which ensures for instance in presence of $\oplus(*(c_i,g_1),\ominus(*(c_i,g_2)))$ in S that c_i or $\oplus(g_1,\ominus(g_2))$ must be in S.

Theorem[VSE] 61 (Correctness of ccl_3):
Let ik, S, $S_\oplus$, and P be finite message sets, where the sets S, $S_\oplus$, and P fulfill the properties $\Re_1$–$\Re_4$ and $\Re_P$. Then we have

$$ik \cap ccl_3(S,S_\oplus,P) = \emptyset \Rightarrow DY(ik) \cap ccl_3(S,S_\oplus,P) = \emptyset.$$

Proof: The proof is similar to that of theorem 46 in Sec. 6.2.4. In the step case, we consider the possible f-s (in the TC-AMP algebra) applied to derive $f(m_0,\ldots,m_{n-1})$ in $ccl_3(S,S_\oplus,P)$ from $m_0,\ldots,m_{n-1}$ in $DYl(ik,n)$. Recall that the proof goal consists in providing some m_i for $0 \leq i < n$ such that $m_i \in ccl_3(S,S_\oplus,P)$ holds and allows us to conclude with the induction hypothesis.

- The cases "$f = h_1$", "$f = h_2$", "$f = pair$", "$f = fst$", "$f = snd$" and "$f = inv$" are handled like in the proof of theorem 46 in Sec. 6.3.4, based on the additional conditions on S and on the ccl_1-similar part in the definition of ccl_3.
- In case "$f = \ominus$", the proof task consists in showing that $m_0 \in ccl_3(S,S_\oplus,P)$ follows from $\ominus(m_0) \in ccl_3(S,S_\oplus,P)$. The cases where m_0 is not a $\oplus$-object is handled like in the proof of theorem 46 in Sec. 6.3.4.

  The complementary case, where m_0 is a $\oplus$-object, is trivial: Since $\ominus(m_0)$ is a $\oplus$-object, $\ominus(m_0) \in ccl_3(S,S_\oplus,P)$ yields two cases:

  1. There is a public variation m of $\ominus(m_0)$ (relative to S) such that $m \in ccl_3^\oplus(S,S_\oplus,P)$: m must be also a public variation of m_0, which implies $m_0 \in ccl_3(S,S_\oplus,P)$.
  2. There is a basic $\oplus$-part m of $\ominus(m_0)$ such that $m \in ccl_3(S,S_\oplus,P)$: $\ominus(m)$ is a basic $\oplus$-part of m_0, which permits to ensue $\ominus(m) \in ccl_3(S,S_\oplus,P)$ (and thus $m_0 \in ccl_3(S,S_\oplus,P)$) from $m \in ccl_3(S,S_\oplus,P)$.
- In case "$f = *$", the proof task consists in showing that $m_1 \in ccl_3(S,S_\oplus,P)$ follows from $*(m_0,m_1) \in ccl_3(S,S_\oplus,P)$. The cases where m_1 is not a $\oplus$-object is handled like in the proof of theorem 46 in Sec. 6.3.4.

  The complementary case, where m_1 is a $\oplus$-object, is trivial: Since $*(m_0,m_1)$ is a $\oplus$-object, $m_0 \notin ccl_3(S,S_\oplus,P)$ and $*(m_0,m_1) \in ccl_3(S,S_\oplus,P)$ yield two cases:

  1. There is a public variation m of $*(m_0,m_1)$ (relative to S) with $m \in ccl_3^\oplus(S,S_\oplus,P)$: $m_0 \notin ccl_3(S,S_\oplus,P)$ implies that $m_0 \notin S$ and that m must be also a public variation of m_1, too, which immediately implies $m_1 \in ccl_3(S,S_\oplus,P)$.
  2. There is a basic $\oplus$-part m of $*(m_0,m_1)$ such that $m \in ccl_3(S,S_\oplus,P)$: $*(inv(m_0),m)$ is a basic $\oplus$-part of the message m_1. Using m_0,$inv(m_0) \notin ccl_3(S,S_\oplus,P)$, we deduce $*(inv(m_0),m) \in ccl_3(S,S_\oplus,P)$ (and thus $m_1 \in ccl_3(S,S_\oplus,P)$) as a consequence of $m \in ccl_3(S,S_\oplus,P)$.
- In case "$f = \oplus$", the proof task consists in showing that $\oplus(m_0,m_1) \in ccl_3(S,S_\oplus,P)$ implies $m_0 \in ccl_3(S,S_\oplus,P) \vee m_1 \in ccl_3(S,S_\oplus,P)$. According to the definition of ccl_3, we distinguish two cases:

1. In the recursion case where $\oplus(m_0,m_1) \in ccl_3(S,S_\oplus,P)$ implies that some basic $\oplus$-part m of $\oplus(m_0,m_1)$ satisfies $m \in ccl_3(S,S_\oplus,P)$, the same m must be (equal or) a basic $\oplus$-part of m_0 (respectively m_1). This immediately allows us to conclude with $m_0 \in ccl_3(S,S_\oplus,P)$ (respectively $m_1 \in ccl_3(S,S_\oplus,P)$).
2. A public variation of $\oplus(m_0,m_1)$ is in $ccl_3^\oplus(S,S_\oplus,P)$: According to the notion of public variation, if m' is a public variation of $\oplus(m_0,m_1)$ then there must be public variations m_0' and m_1' of m_0 and respectively m_1 such that $m' = \oplus(m_0',m_1')$. This allows us to handle this case with the help of lemma 63, in Sec. 6.5.2.3.2. □

6.5.2.3.1 Lemma for $\oplus$-object Case: The lemma described in this section is central in the proof of lemma 63, applied for case "$f = \oplus$" in the proof of theorem 61.

Lemma[VSE] 62 ($ccl_3^\oplus$; TC-AMP ; $\oplus$-object case):
Let m_0 and m_1 be arbitrary messages, and let S, $S_\oplus$ and P be finite sets of messages that fulfill $\Re_1$–$\Re_4$ and $\Re_P$. Then we have

$$(\oplus(m_0,m_1) \in ccl_3^\oplus(S,S_\oplus,P) \wedge obj^\oplus(\oplus(m_0,m_1),m_0,m_1))$$
$$\Rightarrow (m_0 \in ccl_3^\oplus(S,S_\oplus,P) \vee m_1 \in ccl_3^\oplus(S,S_\oplus,P)).$$

Proof: The proof of this lemma is by induction on $|\oplus(m_0,m_1)|_\oplus$, i.e. on the $\oplus$-structure of $\oplus(m_0,m_1)$.

Base Case: Using the definition of $ccl_3^\oplus$, the assumption $\oplus(m_0,m_1) \in ccl_3^\oplus(S,S_\oplus,P)$ yields four cases. Cases 1–3 are handled according to the same principles as in the Step Case (see below). Case 4 is closed by refutation, because it implies that $\oplus(m_0,m_1)$ possesses at least three basic $\oplus$-parts.

Step Case: According to the definition of $ccl_3^\oplus$, $\oplus(m_0,m_1) \in ccl_3^\oplus(S,S_\oplus,P)$ yields to the following cases:

1. (Case 1), $\oplus(m_0,m_1) \in S$: According to $\Re_1$, we obtain $m_0 \in S$ or $m_1 \in S$, which means that m_0 or m_1 is in $ccl_3^\oplus(S,S_\oplus,P)$, as required.
2. (Case 2), $\oplus(m_0,m_1)$ possesses a $\oplus$-part m_2 in $S \setminus S_\oplus$: If $m_2 = m_0$ or $m_2 = m_1$, $m_2 \in S$ permits to conclude as in (1). Otherwise, conditions $\Re_1$ and $\Re_2$ allow us to obtain a $\oplus$-part m_3 (of m_2 and) of m_0 or m_1 and to conclude by case 2 in the definition of $ccl_3^\oplus$.
3. (Case 3), $\oplus(m_0,m_1)$ possesses a basic $\oplus$-part b in $S_\oplus$, $\oplus(m_0,m_1)$ is not in P and the complementary $\oplus$-part m_b of $\oplus(m_0,m_1)$ does not possess any $\oplus$-part u satisfying $\oplus(b,u) \in P$: If $b = m_0$ or $b = m_1$, $b \in S_\oplus$ and $S_\oplus \subseteq S$ permit to conclude as in (1). Otherwise, b must be a basic $\oplus$-part of m_0 (or m_1), which may not be in P and whose complementary $\oplus$-part may not possess any $\oplus$-part u satisfying $\oplus(b,u) \in P$. This permit to conclude by case 3 in the definition of $ccl_3^\oplus$.
4. (Case 4), $\oplus(m_0,m_1)$ possesses u, m_u as complementary $\oplus$-parts satisfying $u \in P$ and $m_u \in ccl_3^\oplus(S,S_\oplus,P)$: W.l.o.g., we assume that (a $\oplus$-part of) u matches or occurs in m_0. This provides us with the following case distinction.
   (a) $m_0 = u$ and $m_1 = m_u$: This immediately yields $m_1 \in ccl_3^\oplus(S,S_\oplus,P)$.
   (b) $m_0 = \oplus(u,m_{u0})$ and $m_u = \oplus(m_{u0},m_1)$: Applying the induction hypothesis to $m_u \in ccl_3^\oplus(S,S_\oplus,P)$ permits to obtain $m_{u0} \in ccl_3^\oplus(S,S_\oplus,P)$ or $m_1 \in ccl_3^\oplus(S,S_\oplus,P)$. The latter case corresponds to our proof goal. In the former case, $u \in P$ permits to apply case 4 in the definition of $ccl_3^\oplus$ and to deduce that $\oplus(u,m_{u0})$, i.e. m_0, is in $ccl_3^\oplus(S,S_\oplus,P)$.

(c) $m_0 = u_0$ and $m_1 = \oplus(u_1, m_u)$: $\Re_2$ ensures that $m_0 \in S_\oplus \subseteq S$, which permits to conclude as in (1).

(d) $m_0 = \oplus(u_0, m_{u0})$, $m_1 = \oplus(u_1, m_{u1})$ and $m_{u1} \neq \infty$ (to exclude a pendant to (c)): This case is shown by contradiction, assuming that m_0 and m_1 are not in $ccl_3^\oplus(S, S_\oplus, P)$. According to $\Re_2$, we have $u_0 \in S_\oplus$, so that $m_0 \notin ccl_3^\oplus(S, S_\oplus, P)$ implies $\oplus(u_0, m_{u0}) \in P$ or $\oplus(u_0, v_0) \in P$ and $m_{v0} \notin ccl_3^\oplus(S, S_\oplus, P)$ for $m_{u0} = \oplus(v_0, m_{v0})$. Similarly, $m_1 \notin ccl_3^\oplus(S, S_\oplus, P)$ implies $\oplus(u_1, m_{u1}) \in P$ or $\oplus(u_1, v_1) \in P$ and $m_{v1} \notin ccl_3^\oplus(S, S_\oplus, P)$ for $m_{u1} = \oplus(v_1, m_{v1})$. This yields (mainly) to the following three cases:

(A) $\oplus(u_0, m_{u0}) \in P$ and $\oplus(u_1, m_{u1}) \in P$: According to the additional condition $\Re_2$, we deduce $\oplus(\ominus(u_1), m_{u0}) \in P$ or $\oplus(\ominus(u_{11}), m_{u01}), \oplus(\ominus(u_{12}), m_{u02}) \in P$ from $\oplus(u_0, m_{u0}), \oplus(u_0, u_1) \in P$. In the latter case, the protocol-specific condition $\Re_P$ provides us (mainly) with the following possible structures on the involved $\oplus$-objects:

i. $u_0 = b_1, u_1 = \oplus(b_2, b_3), m_{u0} = \oplus(b_4, \oplus(b_5, b_6))$ for basic $\oplus$-parts b_1–b_6 such that $\oplus(b_2, \ominus(b_4)), \oplus(b_3, \oplus(\ominus(b_5), \ominus(b_6)))$ belongs to P: This permits to obtain $\oplus(\oplus(b_2, b_3), m_{u1}) \in P$ from $\oplus(u_1, m_{u1}) \in P$ and to get $m_u = \oplus(\oplus(b_4, \oplus(b_5, b_6)), m_{u1})$ from $m_u = \oplus(m_{u0}, m_{u1})$.
First, we use that $\oplus(\oplus(b_2, b_3), m_{u1}), \oplus(b_2, \ominus(b_4))$ are in P to deduce $\oplus(\oplus(b_4, b_3), m_{u1}) \in P$ by $\Re_2$ and $\Re_P$.
Then, we use $\oplus(\oplus(b_4, b_3), m_{u1}), \oplus(b_3, \oplus(\ominus(b_5), \ominus(b_6))) \in P$ combined with $\Re_2$ and $\Re_P$ to obtain three possible cases:
– P includes $\oplus(b_4, b_5)$ and $\oplus(b_6, m_{u1})$.
– $m_{u1} = \oplus(b_7, b_8)$ for basic $\oplus$-parts b_7, b_8 and P includes $\oplus(b_7, b_5)$ and $\oplus(b_6, \oplus(b_4, b_8))$.
– P includes $\oplus(\oplus(b_4, m_{u1}), \oplus(b_5, b_6))$.
All three cases permit us to show $m_u \notin ccl_3^\oplus(S, S_\oplus, P)$ by the definition of $ccl_3^\oplus$. Hence, we conclude by refuting $m_u \in ccl_3^\oplus(S, S_\oplus, P)$.

ii. $u_0 = b_1, m_{u0} = \oplus(b_2, b_3), u_1 = \oplus(b_4, \oplus(b_5, b_6))$ for basic $\oplus$-parts b_1–b_6 with $\oplus(b_2, \ominus(b_4)), \oplus(b_3, \oplus(\ominus(b_5), \ominus(b_6))) \in P$: This permits to obtain $\oplus(\oplus(b_4, \oplus(b_5, b_6)), m_{u1}) \in P$ from $\oplus(u_1, m_{u1}) \in P$ and to get $m_u = \oplus(\oplus(b_2, b_3), m_{u1})$ from $m_u = \oplus(m_{u0}, m_{u1})$.
First, we use that $\oplus(\oplus(b_4, \oplus(b_5, b_6)), m_{u1}), \oplus(b_2, \ominus(b_4))$ are in P to deduce $\oplus(\oplus(b_2, \oplus(b_5, b_6)), m_{u1}) \in P$ by $\Re_2$ and $\Re_P$.
Then, we use $\oplus(\oplus(b_2, \oplus(b_5, b_6)), m_{u1}), \oplus(b_3, \oplus(\ominus(b_5), \ominus(b_6))) \in P$ combined with $\Re_2$ and $\Re_P$ to obtain $\oplus(\oplus(b_2, b_3), m_{u1}) \in P$, i.e. $m_u \in P$. This allows us to show $m_u \notin ccl_3^\oplus(S, S_\oplus, P)$ by the definition of $ccl_3^\oplus$. Hence, we conclude by refuting $m_u \in ccl_3^\oplus(S, S_\oplus, P)$.

iii. $u_0 = \oplus(b_1, b_2), u_1 = \oplus(b_3, b_4), m_{u0} = \oplus(b_5, b_6)$ for basic $\oplus$-parts b_1–b_6 such that $\oplus(b_3, \ominus(b_5)), \oplus(b_4, \ominus(b_6))$ are in the set P: This permits to obtain $\oplus(\oplus(b_3, b_4), m_{u1}) \in P$ from $\oplus(u_1, m_{u1}) \in P$ and to get $m_u = \oplus(\oplus(b_5, b_6), m_{u1})$ from $m_u = \oplus(m_{u0}, m_{u1})$.
First, we use that $\oplus(\oplus(b_3, b_4), m_{u1}), \oplus(b_3, \ominus(b_5))$ are in the set P to deduce $\oplus(\oplus(b_5, b_4), m_{u1}) \in P$ by $\Re_2$ and $\Re_P$.
Then, we use $\oplus(\oplus(b_5, b_4), m_{u1}), \oplus(b_4, \ominus(b_6)) \in P$ combined with $\Re_2$ and $\Re_P$ to obtain $\oplus(\oplus(b_5, b_6), m_{u1}) \in P$, i.e. $m_u \in P$. This allows us to show $m_u \notin ccl_3^\oplus(S, S_\oplus, P)$ by the definition of $ccl_3^\oplus$. Hence, we conclude by refuting $m_u \in ccl_3^\oplus(S, S_\oplus, P)$.

In case $\oplus(\ominus(u_1), m_{u0})$ is in P, we use $\oplus(u_1, m_{u1}) \in P$ and condition $\Re_2$ to obtain $\oplus(m_{u0}, m_{u1}) \in P$ or $\oplus(m_{u01}, m_{u11}), \oplus(m_{u02}, m_{u12}) \in P$ for m_{u01}, m_{u02} $\oplus$-parts of m_{u0} and for m_{u11}, m_{u12} $\oplus$-parts of m_{u1}. In both cases, the definition of

$ccl_3^{\oplus}$ permits us to obtain $\oplus(m_{u0}, m_{u1}) \notin ccl_3^{\oplus}(S, S_{\oplus}, P)$ for $m_u = \oplus(m_{u0}, m_{u1})$. Hence, we conclude by refuting $m_u \in ccl_3^{\oplus}(S, S_{\oplus}, P)$.

(B) $\oplus(u_0, m_{u0}) \in P$, $\oplus(u_1, v_1) \in P$ and $m_{v1} \notin ccl_3^{\oplus}(S, S_{\oplus}, P)$ for $m_{u1} = \oplus(v_1, m_{v1})$: According to the additional condition $\Re_2$, we deduce $\oplus(\ominus(u_1), m_{u0}) \in P$ or $\oplus(\ominus(u_{11}), m_{u01}), \oplus(\ominus(u_{12}), m_{u02}) \in P$ from $\oplus(u_0, m_{u0}), \oplus(u_0, u_1) \in P$.
First, we proceed as in (A) using v_1 for m_{u1} to get $\oplus(m_{u0}, v_1) \notin ccl_3^{\oplus}(S, S_{\oplus}, P)$. Then, we use $m_u = \oplus(\oplus(m_{u0}, v_1), m_{v1})$ together with $m_u \in ccl_3^{\oplus}(S, S_{\oplus}, P)$ to apply the induction hypothesis and obtain that $\oplus(m_{u0}, v_1)$ or m_{v1} is in $ccl_3^{\oplus}(S, S_{\oplus}, P)$. This allows us to conclude by refuting $\oplus(m_{u0}, v_1), m_{v1} \notin ccl_3^{\oplus}(S, S_{\oplus}, P)$.

(C) $\oplus(u_0, v_0) \in P$, $m_{v0} \notin ccl_3^{\oplus}(S, S_{\oplus}, P)$ for $m_{u0} = \oplus(v_0, m_{v0})$, $\oplus(u_1, v_1) \in P$ and such that $m_{v1} \notin ccl_3^{\oplus}(S, S_{\oplus}, P)$ for $m_{u1} = \oplus(v_1, m_{v1})$:
According to the additional condition $\Re_2$, we deduce $\oplus(\ominus(u_1), v_0) \in P$ or $\oplus(\ominus(u_{11}), v_{01}), \oplus(\ominus(u_{12}), v_{02}) \in P$ from $\oplus(u_0, v_0), \oplus(u_0, u_1) \in P$.
First, we proceed as in (A) using v_0 and v_1 instead of m_{u0} and respectively m_{u1} to obtain $\oplus(v_0, v_1) \notin ccl_3^{\oplus}(S, S_{\oplus}, P)$. Then, we use $m_u = \oplus(\oplus(v_0, v_1), \oplus(m_{v0}, m_{v1}))$ together with $m_u \in ccl_3^{\oplus}(S, S_{\oplus}, P)$ to apply the induction hypothesis and obtain that $\oplus(v_0, v_1)$ or $\oplus(m_{v0}, m_{v1})$ is in $ccl_3^{\oplus}(S, S_{\oplus}, P)$. This allows us to deduce $\oplus(m_{v0}, m_{v1}) \in ccl_3^{\oplus}(S, S_{\oplus}, P)$ using $\oplus(v_0, v_1) \notin ccl_3^{\oplus}(S, S_{\oplus}, P)$. Finally, we again apply the induction hypothesis to $\oplus(m_{v0}, m_{v1}) \in ccl_3^{\oplus}(S, S_{\oplus}, P)$, in order to get that m_{v0} or m_{v1} is in $ccl_3^{\oplus}(S, S_{\oplus}, P)$. This permits to conclude by refuting $m_{v0}, m_{v1} \notin ccl_3^{\oplus}(S, S_{\oplus}, P)$. □

6.5.2.3.2 Lemma for $\oplus$-case: The lemma described in this section allows us to handle directly case "$f = \oplus$" in the proof of theorem 61.

Lemma[VSE] **63 (ccl_3 ; TC-AMP ; $\oplus$-case):**
Let m_0 and m_1 be arbitrary messages, and let S, $S_{\oplus}$ and P be finite sets of messages that fulfill $\Re_1$–$\Re_4$ and $\Re_P$. Then we have

$$\oplus(m_0, m_1) \in ccl_3^{\oplus}(S, S_{\oplus}, P) \Rightarrow (m_0 \in ccl_3^{\oplus}(S, S_{\oplus}, P) \vee m_1 \in ccl_3^{\oplus}(S, S_{\oplus}, P)).$$

Proof: The proof is by induction on the $\oplus$-structure of m_0 and m_1, i.e. on $|m_0|_{\oplus} + |m_1|_{\oplus}$.

Base Case: Neither m_0 nor m_1 is a $\oplus$-object. Here, almost all cases resulting by the axiom about the operations of $\oplus(m_0, m_1)$ are closed by refutation (e.g., $\oplus(m_0, m_1) = \infty$ is refuted by $\infty \notin ccl_3^{\oplus}(S, S_{\oplus}, P)$). Only case (11), where $obj^{\oplus}(\oplus(m_0, m_1), m_0, m_1)$ holds, is shown by lemma 62.

Step Case: m_0 or m_1 is a $\oplus$-object. After the application of the axiom about the operations of $\oplus(m_0, m_1)$, we obtain certain cases ((1)–(4)) handled by refutation, case (11), where $obj^{\oplus}(\oplus(m_0, m_1), m_0, m_1)$ holds, handled by lemma 62, and six more cases ((5)–(10)).

First, the proof for case (5), which is also replayed for cases (6)–(8), is trivial: In case (5), where $obj^{\oplus}(m_1, \ominus(b), m_b)$, $obj^{\ominus}(\ominus(b), m_0)$ and $\oplus(m_0, m_1) = m_b$ hold, we assume that $m_0 \notin ccl_3^{\oplus}(S, S_{\oplus}, P)$ holds and want to show $m_1 \in ccl_3^{\oplus}(S, S_{\oplus}, P)$: Since m_0, i.e. b, is a basic $\oplus$-part, $b \notin ccl_3^{\oplus}(S, S_{\oplus}, P)$ means according to the definition of $ccl_3^{\oplus}$ that b and $\ominus(b)$ are not in S. This permits to use $m_b \in ccl_3^{\oplus}(S, S_{\oplus}, P)$, to transfer all cases obtained by the definition of $ccl_3^{\oplus}$ to $\oplus(\ominus(b), m_b)$, i.e. m_1, and thus to show $m_1 \in ccl_3^{\oplus}(S, S_{\oplus}, P)$.

Next, we describe the proof for case (9), which is also replayed for case (10). In case (9), where $obj^{\oplus}(m_0,b,m_{b0})$, $obj^{\oplus}(m_1,\ominus(b),m_{b1})$, $obj^{\ominus}(\ominus(b),b)$ and $\oplus(m_0,m_1) = \oplus(m_{b0},m_{b1})$ hold, $\oplus(m_0,m_1) \in ccl_3^{\oplus}(S,S_{\oplus},P)$ implies $\oplus(m_{b0},m_{b1}) \in ccl_3^{\oplus}(S,S_{\oplus},P)$. We proceed by a case distinction:

- $b \notin ccl_3^{\oplus}(S,S_{\oplus},P)$: We have as well $\ominus(b) \notin ccl_3^{\oplus}(S,S_{\oplus},P)$. Applying the induction hypothesis to $\oplus(m_{b0},m_{b1}) \in ccl_3^{\oplus}(S,S_{\oplus},P)$, we obtain $m_{b0} \in ccl_3^{\oplus}(S,S_{\oplus},P)$ or the consequence $m_{b1} \in ccl_3^{\oplus}(S,S_{\oplus},P)$. Replaying the proof of case (5) above, we transfer the former case to $m_0 \in ccl_3^{\oplus}(S,S_{\oplus},P)$ and the latter case to $m_1 \in ccl_3^{\oplus}(S,S_{\oplus},P)$.

- $b \in ccl_3^{\oplus}(S,S_{\oplus},P)$: Only case 1 in the definition of $ccl_3^{\oplus}$ applies, since b is not a $\oplus$-object. If $b \notin S_{\oplus}$, we trivially show $m_0 \in ccl_3^{\oplus}(S,S_{\oplus},P)$ by case 2 in the definition of $ccl_3^{\oplus}$. Otherwise, we have $b \in S_{\oplus}$ and we handle this case by refuting the assumption $m_0,m_1 \notin ccl_3^{\oplus}(S,S_{\oplus},P)$ as described below.

  Since $b \in S_{\oplus}$, having m_0, i.e. $\oplus(b,m_{b0})$, not in $ccl_3^{\oplus}(S,S_{\oplus},P)$ implies $\oplus(b,m_{b0}) \in P$ or $\oplus(b,u_0) \in P$ and $m_{u0} \notin ccl_3^{\oplus}(S,S_{\oplus},P)$ for $m_{b0} = \oplus(u_0,m_{u0})$. Similarly, having m_1, i.e. $\oplus(\ominus(b),m_{b1})$, not in $ccl_3^{\oplus}(S,S_{\oplus},P)$ implies $\oplus(\ominus(b),m_{b1}) \in P$ or $\oplus(\ominus(b),u_1) \in P$ and $m_{u1} \notin ccl_3^{\oplus}(S,S_{\oplus},P)$ for $m_{b1} = \oplus(u_1,m_{u1})$. This yields (mainly) to the following three cases:

  1. $\oplus(b,m_{b0}) \in P$ and $\oplus(\ominus(b),m_{b1}) \in P$: According to $\Re_2$, this permits to obtain $\oplus(m_{b0},m_{b1}) \in P$ or $\oplus(m_{b01},m_{b11}),\oplus(m_{b02},m_{b12}) \in P$ for $\oplus$-parts m_{b01},m_{b02} of m_{b0} and for $\oplus$-parts m_{b11},m_{b12} of m_{b1}. In both cases, the definition of $ccl_3^{\oplus}$ allows us to obtain $\oplus(m_{b0},m_{b1}) \notin ccl_3^{\oplus}(S,S_{\oplus},P)$, which permits to conclude by refuting $\oplus(m_{b0},m_{b1}) \in ccl_3^{\oplus}(S,S_{\oplus},P)$.
  2. $\oplus(b,m_{b0}) \in P$, $\oplus(\ominus(b),u_1) \in P$ and $m_{u1} \notin ccl_3^{\oplus}(S,S_{\oplus},P)$ for $m_{b1} = \oplus(u_1,m_{u1})$:
     First, we proceed as in (1) using u_1 for m_{b1} to deduce $\oplus(m_{b0},u_1) \notin ccl_3^{\oplus}(S,S_{\oplus},P)$. Then, we use $\oplus(m_{b0},m_{b1}) = \oplus(\oplus(m_{b0},u_1),m_{u1})$ and $\oplus(m_{b0},m_{b1}) \in ccl_3^{\oplus}(S,S_{\oplus},P)$ to apply the induction hypothesis and obtain that the message $\oplus(m_{b0},u_1)$ or m_{u1} is in $ccl_3^{\oplus}(S,S_{\oplus},P)$. Finally, this permits us to deduce $m_{u1} \in ccl_3^{\oplus}(S,S_{\oplus},P)$ from $\oplus(m_{b0},u_1) \notin ccl_3^{\oplus}(S,S_{\oplus},P)$ and then to conclude by refuting $m_{u1} \notin ccl_3^{\oplus}(S,S_{\oplus},P)$.
  3. $\oplus(b,u_0) \in P$, $m_{u0} \notin ccl_3^{\oplus}(S,S_{\oplus},P)$ for $m_{b0} = \oplus(u_0,m_{u0})$, $\oplus(\ominus(b),u_1) \in P$ and $m_{u1} \notin ccl_3^{\oplus}(S,S_{\oplus},P)$ for $m_{b1} = \oplus(u_1,m_{u1})$:
     First, we proceed as in case (1) using u_0 and u_1 for m_{b0} and respectively m_{b1} to deduce that the message $\oplus(u_0,u_1)$ is not in $ccl_3^{\oplus}(S,S_{\oplus},P)$. Then, we use $\oplus(m_{b0},m_{b1}) = \oplus(\oplus(u_0,u_1),\oplus(m_{u0},m_{u1}))$ and $\oplus(m_{b0},m_{b1}) \in ccl_3^{\oplus}(S,S_{\oplus},P)$ to apply the induction hypothesis and deduce that $\oplus(u_0,u_1)$ or $\oplus(m_{u0},m_{u1})$ is in $ccl_3^{\oplus}(S,S_{\oplus},P)$. This permits to obtain $\oplus(m_{u0},m_{u1}) \in ccl_3^{\oplus}(S,S_{\oplus},P)$ from $\oplus(u_0,u_1) \notin ccl_3^{\oplus}(S,S_{\oplus},P)$. Finally, we again apply the induction hypothesis to $\oplus(m_{u0},m_{u1}) \in ccl_3^{\oplus}(S,S_{\oplus},P)$ and get that m_{u0} or m_{u1} is in $ccl_3^{\oplus}(S,S_{\oplus},P)$, which allows us to conclude by refuting $m_{u0},m_{u1} \notin ccl_3^{\oplus}(S,S_{\oplus},P)$. □

Chapter 7

Automated Inductive Verification

In this chapter, we describe the VSE framework for the inductive verification of cryptographic protocols that are based on message algebras. It includes specification support in form of re-usable (abstract data type (ADT)) theories and proof construction support in form of heuristics.

7.1 Protocol Formalization

We formalize protocols in the specification language of VSE (VSE-SL) using abstract data types (ADTs), where the corresponding axioms and properties are expressed in first order logic with equality. The VSE tool provides us with means to structure the specification of ADTs into a development graph of VSE-theories. In particular, "satisfies"-links can be established between the axioms and the theorems of any ADT yielding two adjoining VSE-theories. The theorems can thus be proven *locally*, i.e. relative to the axioms and to the properties from the *imported* VSE-theories, if any. These structuring means are important as they permit us to separate protocol formalizations into

1. *common* VSE-theories,
2. *algebra-specific* VSE-theories,
3. and *protocol-specific* VSE-theories.

The common VSE-theories belong to our implementation of the original inductive method, [30], and can be re-used independent from the considered message algebra. The algebra-specific VSE-theories include the axioms about the message structures in Chap. 5 and the corresponding formal notions of the proof techniques in Chap. 6, 10 and 11. They can be re-used for protocols that are based on the same message algebra.

In this section, we give an overview on the content of these VSE-theories, focusing more on the first category. Instead of listing the ADTs, we describe the corresponding mathematical objects.

7.1.1 Agents and Messages

We start with the basic constituents of the events in protocol traces: agents (participants) and messages. The ADTs for agent identifiers and for messages are declared in the VSE-theories `bagent` and `tmsg`, respectively, which are shown in Fig. 7.1. While the former is common, the latter is algebra-specific, as it contains the axioms about the message structures according to the considered equations.

The ADT in `bagent` provides us with the set of agent identifiers and the ADT in `tmsg` fixes the set $Mes_0(\Sigma, At)$ of message terms (see Def. (11)).

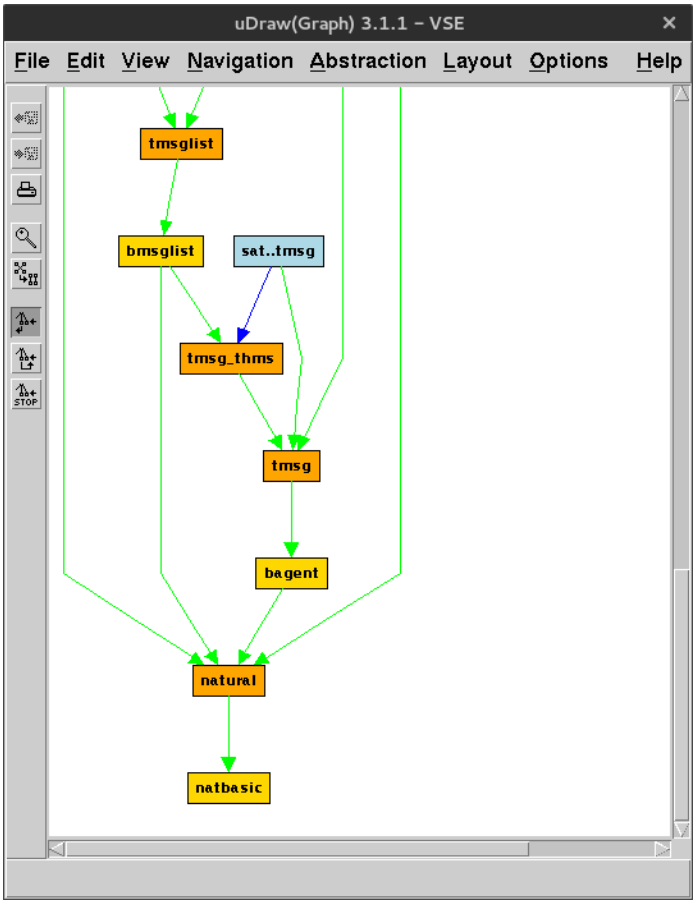

Figure 7.1: An excerpt from a development graph enclosing the VSE-theories `bagent`, `tmsg` and `tmsglist`

Definition$^{\text{VSE}}$ 64 (Agents, Messages):

1. The set Ags of all *agent identifiers* is the disjoint union of
   (a) $\{spy\}$, and
   (b) $\{agt(n) \mid n \in \mathbb{N}\}$.
2. We associate every agent identifier with an *atomic message* $ag(i)$ according to the *bijective* relation
   $$\{(ag(0), spy)\} \cup \{(ag(i+1), agt(i)) \mid i \in \mathbb{N}\}.$$
   The $ag(i)$-s are called *agent names*. They are the counterparts of agent identifiers in messages.
3. The set $\{nc(i) \mid i \in \mathbb{N}\}$ contains the *nonces*.
4. The set $Mes_0(\Sigma, At)$ of *message terms* is defined as in Def. 11 relative to a signature Σ and the set of atomic messages
   $$At = \{ag(i) \mid i \in \mathbb{N}\} \uplus \{nc(i) \mid i \in \mathbb{N}\} \uplus \overline{At},$$
   where $\overline{At}$ contains the atomic messages that differ from nonces, agent names and from all $f \in \Sigma\langle 0 \rangle$.

The distinction between agent identifiers and names is due to the structure of events, as described in Sec. 7.1.2.

We determine the signature Σ of the function symbols for the cryptographic primitives, according to the message algebra that the verified protocol is based on. Similarly, we determine the kinds of atomic messages that are used in addition to nonces and agent names according to the protocol: The set $\overline{At}$ is supposed to contain infinitely, but countably many atomic messages for each additional kind.

Except of the function symbols (in Σ) and their associated equations, which constitute in fact the considered message algebra, the set At of the atomic messages is not really algebra-specific. The axioms about the message structures are practically the same, independent from the different kinds of the elements in At. So, it is straightforward to re-use them in the verification of a new protocol if it makes use of other kinds of atomic messages, provided it is based on the same cryptographic primitives (in Σ) with the same algebraic equations. For instance, the axioms of the PACE algebra can be re-used for the verification of any protocol where no other equations than those for the DH exchange and for the enc/decryption are necessary.

7.1.2 Events and Traces

Fig. 7.2 shows an excerpt with many common VSE-theories of a development graph. These include `bprotocolevent` and `bprotocoltrace`, where the ADTs for events and traces are declared, respectively. The latter is extended in (a common VSE-theory) `tprotocoltrace` with an append function on traces and with a predicate to express the membership of events. We will use the mathematical notations "#" and "$\in$" for the append function and the membership predicate, respectively.

The ADTs in `bprotocolevent` and `bprotocoltrace` provide us with the sets of events and traces, respectively.

Definition$^{\mathbf{VSE}}$ 65 (Events, Traces):

1. The set Evs of all protocol events is the disjoint union of
   (a) $\{says(ag,ag',m) \mid ag,ag' \in Ags \text{ and } m \in Mes_0(\Sigma, At)\}$,
   (b) $\{gets(ag,m) \mid ag \in Ags \text{ and } m \in Mes_0(\Sigma, At)\}$,
   (c) $\{note(ag,m) \mid ag \in Ags \text{ and } m \in Mes_0(\Sigma, At)\}$, and
   (d) $\{send(ag,ag',m_0,m_1) \mid ag,ag' \in Ags \text{ and } m_0,m_1 \in Mes_0(\Sigma, At)\}$.
2. The set Trs of all protocol traces is defined inductively by
$$\epsilon \in Trs$$
$$(ev \in Evs \wedge tr \in Trs) \Rightarrow ev.tr \in Trs.$$

In addition to the standard events, i.e., *says*, *gets* and *note* for message sending, receiving and recording, respectively, we use *send* events to represent *simultaneous says* and *note* events. That is, $send(ag,ag',m_0,m_1)$ means that ag has sent message m_1 to ag' and has *at the same time* recorded message m_0. Unlike the loose use of "*note*" and "*says*", where the recorded message m_0 is not necessarily bound to the sent message m_1, the use of "*send*" binds both messages. This is helpful to model chronological order between different events, such as m_0 is received before sending m_1. Examples are in Sec. 8.1.2 and 9.1.2.

We use ϵ and $x.l$ to denote the empty list and a list l extended by an element x, respectively. In lists $ev.tr$ that represent protocol traces the events are listed in the opposite chronological order. That is, ev is the last event in the trace $ev.tr$.

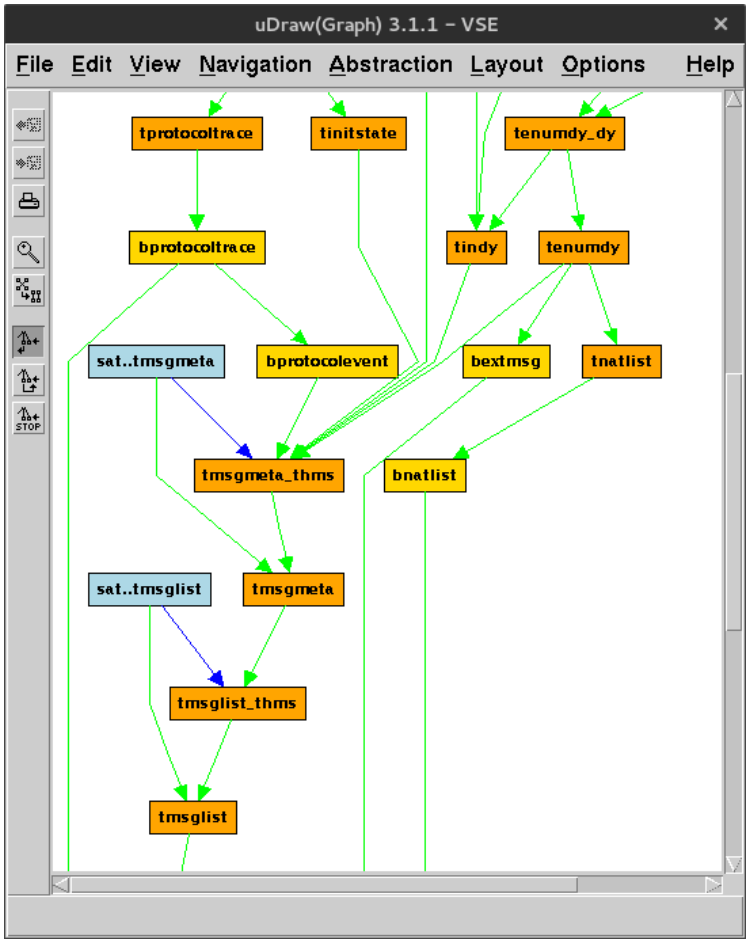

Figure 7.2: An excerpt from a development graph enclosing the VSE-theories `tmsgmeta`, `bprotocolevent`, `bprotocoltrace`, `tprotocoltrace`, `tindy` and `tenumdy`

7.1.3 Observable Message Sets

The events in a given trace *tr* determine the messages involved agents and intruder may observe from this trace. In case of *note* and *send* events, the observed messages by the intruder depend on whether the involved agents are *compromised*.

We formalized the notions about the immediately observable message sets in the common VSE-theories `tinitstate` and `ttraceinfo`, which are shown in Fig. 7.3. The former includes the following axioms on static corruption:

Axiom$^{\text{VSE}}$ 66 (Static Corruption):

1. $bad \subseteq Ags$ is the set of *compromised* agents.
2. The intruder belongs to the set of compromised agents, i.e.

$$spy \in bad.$$

Using the *bad* set, we are able to incorporate the notion of static corruption in the definition of the immediately observable messages by the attacker.

Definition$^{\mathbf{VSE}}$ 67 (Observable Messages):
The list $obset(ag,tr)$ of (*immediately*) observable messages for an agent ag and a trace tr is defined recursively by

$$obset(ag,\epsilon) = \epsilon$$

$$obset(ag,says(ag',ag'',m)\bullet tr) = \begin{cases} m\bullet obset(ag,tr) & : \ ag = ag' \text{ or } ag = spy \\ obset(ag,tr) & : \ \text{otherwise} \end{cases}$$

$$obset(ag,gets(ag',m)\bullet tr) = \begin{cases} m\bullet obset(ag,tr) & : \ ag = ag' \text{ and } ag \neq spy \\ obset(ag,tr) & : \ \text{otherwise} \end{cases}$$

$$obset(ag,note(ag',m)\bullet tr) = \begin{cases} m\bullet obset(ag,tr) & : \ ag = ag' \text{ or} \\ & \quad ag = spy \text{ and } ag' \in bad \\ obset(ag,tr) & : \ \text{otherwise} \end{cases}$$

$$obset(ag, send(ag',ag'',m,m')\bullet tr) = \begin{cases} m\bullet m'\bullet obset(ag,tr) & : \ ag = ag' \text{ or} \\ & \quad ag = spy \text{ and } ag' \in bad \\ m'\bullet obset(ag,tr) & : \ ag \neq ag', ag = spy \\ & \quad \text{and } ag' \notin bad \\ obset(ag,tr) & : \ \text{otherwise} \end{cases}$$

This definition is contained in the VSE-theory `ttraceinfo`. $obset(ag,tr)$ results in a list that represents a *finite* set of messages. It models the *local view* of (the immediately observable messages by) the agent ag on (from) the trace tr. In particular, it does not a priori include the *initial* knowledge of ag; *obset* results in the empty list when the given trace is empty. It is not possible to include the initial knowledge right from the beginning, since this can be infinite, as discussed in Sec. 7.1.4. Instead of that, we will have *note* events that model an access to initially known information before it is used in protocol messages.

We will use $spies(tr)$ to abbreviate $obset(spy,tr)$. Note that in case of a new *gets* event, the list $spies(tr)$ is not extended. It is expected that the received message was sent earlier in a *says* or *send* event, and it is thus already observed by the intruder (see the generic protocol rules in section 7.1.6).

Finally, we want to point out that the observed messages from a *send* event are equivalent to those given by a *note* and a *says* event.

7.1.4 Initial Knowledge

Protocols are deployed in application scenarios where the participants are assumed to possess (or to access) certain kinds of initial knowledge before starting any run. This includes long-term keys and parameters required for use with certain cryptographic primitives like static generators in DH exchange. The configuration of the initial knowledge determines the possible roles and communication partners of the participants. Unlike in the bounded session models where the initial knowledge is given by *finite* sets, participants are allowed in the trace models to start unlimited number of sessions and thus possess (or access) as much initial knowledge items as required. Instead of finite sets, we need for this reason to associate the participants with their initial knowledge using the predicate $initHas(ag,m)$. It is declared in the common VSE-theory `tinitstate`, shown in Fig. 7.3. It allows us to express that the knowledge items in m are initially known by agents ag.

As it is introduced in Sec 7.1.3, we do not include the entire initial knowledge in the list of the observable messages. Instead of that, every participant is free to extend her observable messages with any message part that she initially has. This is carried out through a *note* event. Hence, we obtain two *generic* extension rules for protocol traces, which are independent from the considered protocol.

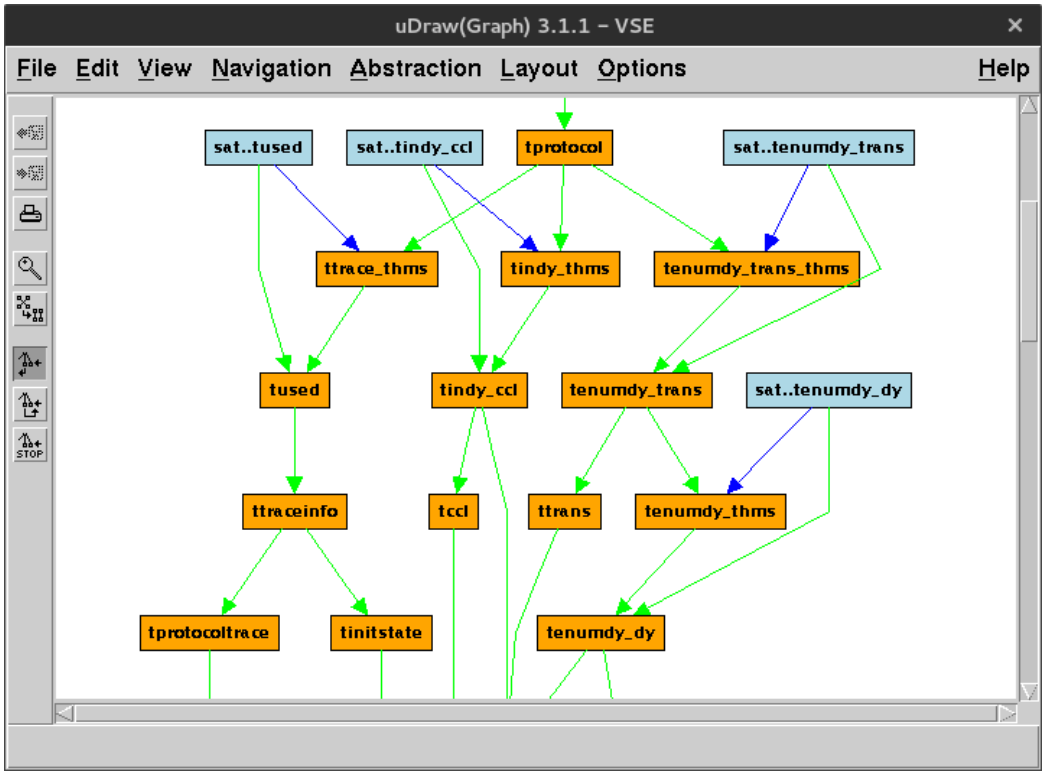

Figure 7.3: An excerpt from a development graph enclosing the VSE-theories `tinitstate`, `tprotocoltrace`, `ttraceinfo`, `tprotocol`, `tused`, `ttrace_thms`, `tindy_thms` and tccl

Definition$^{\text{VSE}}$ 68 (Access):

$$Access_{reg}(ev) \Leftrightarrow (\exists ag, m : initHas(ag, m) \wedge ev = note(ag, m))$$
$$Access_{bad}(ev) \Leftrightarrow (\exists ag, m : ag \in bad \wedge initHas(ag, m) \wedge ev = note(spy, m))$$

This definition is included in the common VSE-theory `tprotocol`, shown in Fig. 7.3. $Access_{reg}(ev)$ and $Access_{bad}(ev)$ will be used as two additional admissible alternatives for extending any given protocol trace with an event ev (see Sec. 7.1.6). They represent the access to the initial knowledge: Every participant may access her own initial knowledge; The intruder may access additionally the initial knowledge of compromised agents.

Note that the initial knowledge item m in Def. 68 must be further specified according to the protocol. Simple axioms are used (as part of the protocol model) to fix the content of the initial knowledge and the corresponding assumptions. Examples are in Sec. 8.1.1 and 9.1.1.

7.1.5 Freshness

The generation of a *new* nonce (or of another kind of atomic messages) is expressed with the help of a predicate *used*, as defined below. A message is used in a trace, iff it occurs in the initial knowledge of any agent or in some event message of this trace. The algebra-specific part of the definition is expressed through *uses*, whose definition belongs to the algebra-specific VSE-theory `tmsgmeta`, shown in Fig. 7.2.

Definition$^{\mathbf{VSE}}$ 69 (used):
For all messages m and all traces tr we have

$$\begin{aligned} used(m,tr) \Leftrightarrow ((&\exists ag,m' : initHas(ag,m') \wedge uses(m',m)) \\ &\vee (\exists ag,m' : (gets(ag,m') \in tr \vee note(ag,m') \in tr) \wedge uses(m',m)) \\ &\vee (\exists ag,ag',m' : says(ag,ag',m') \in tr \wedge uses(m',m)) \\ &\vee (\exists ag,ag',m',m'' : send(ag,ag',m',m'') \in tr \\ &\qquad \wedge (uses(m',m) \vee uses(m'',m)))), \end{aligned}$$

where $uses(m_0,m_1)$ means that m_1 occurs in m_0, as defined in Sec. 5.5.4.

This definition is included in the common VSE-theory `tused`, shown in Fig. 7.3.

7.1.6 Protocol Traces

In this section we describe the specification of the traces given by the possible runs of the considered protocol. As a convention, we refer to this subset of *Trs* by the name $\mathcal{P}$ of the protocol. Its definition is included in a VSE-theory named `t_`$\mathcal{P}$. This theory is imported by a VSE-theory `t_`$\mathcal{P}$`_properties` which we reserve for the protocol properties. Justifiably, the latter is linked through "satisfies"-link to the former. Both VSE-theories are obviously protocol-specific and their content is formalized using the available ADTs in `tprotocol`, which we introduced in the previous sections. Fig. 7.4 shows all three VSE-theories in case of the formalization of PACE.

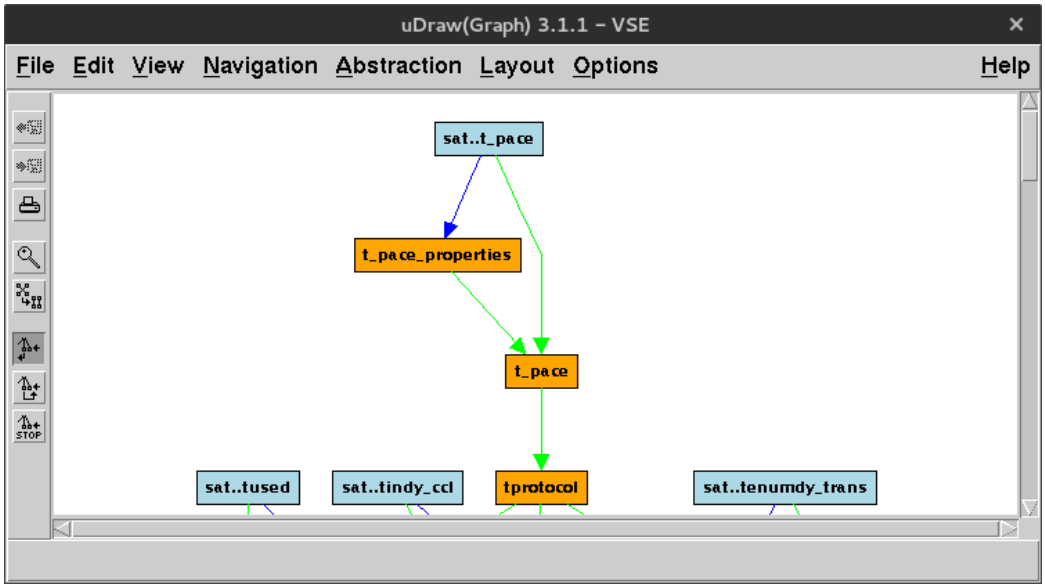

Figure 7.4: An excerpt from the development graph in the verification of PACE, enclosing the VSE-theories `tprotocol`, `t_pace` and `t_pace_properties`

The set $\mathcal{P}$ is defined according to the following schema:

Definition Schema$^{\mathbf{VSE}}$ 70 ($\mathcal{P}$):
The set $\mathcal{P}$ of protocol traces is defined by

$$\begin{aligned}
&\epsilon \in \mathcal{P}, \text{ and}\\
&ev.tr \in \mathcal{P} \Leftrightarrow\\
&(tr \in \mathcal{P} \wedge (Access_{reg}(ev) \vee Access_{bad}(ev) \vee \mathcal{P}_1(ev,tr) \vee \ldots\\
&\qquad \vee \mathcal{P}_n(ev,tr) \vee Oops_{ev}(ev,tr) \vee Gets_{ev}(ev,tr) \vee Fake_{ev}(ev,tr))).
\end{aligned}$$

The set $\mathcal{P}$ contains the empty trace ϵ. It contains inductively any extension of its elements tr with an *admissible* event ev. The predicates in the disjunction on the right-hand side provide us with a complete case distinction on all *admissible*, alternative extensions. Their definitions determine the structure of the corresponding events ev and express the corresponding conditions about prior events from the extended trace tr and about the constituents of ev.

In addition to $Access_{reg}$ and $Access_{bad}$, we have two more *generic* cases: message receiving and fake steps. The corresponding definitions are included likewise in the common VSE-theory `tprotocol`.

Definition$^{\mathbf{VSE}}$ 71 ($Gets_{ev}$ and $Fake_{ev}$):

$$\begin{aligned}
&Gets_{ev}(ev,tr) \Leftrightarrow\\
&(\exists ag,ag',m : (says(ag,ag',m) \in tr \vee (\exists m' : send(ag,ag',m',m) \in tr))\\
&\qquad \wedge ev = gets(ag',m))\\
&Fake_{ev}(ev,tr) \Leftrightarrow\\
&(\exists ag,m : m \in DY(spies(tr)) \wedge ev = says(spy,ag,m))
\end{aligned}$$

A message m can be received by some participant, only if she is the intended receiver (of m) in a prior *says* or *send* event. The attacker can send any message from her own knowledge $DY(spies(tr))$ to any participant. Our definitions of "DY" and "DYl" in Def. 13 are replaced with equivalent definitions of (set) predicates in the VSE-theory `tindy`, which is shown in Fig. 7.2. Only the corresponding predicate to "DYl" is algebra-specific, but its definition can be adapted straightforwardly to any other set Op of the available function symbols.

The definitions of the remaining cases in Def. 70 are protocol-specific, since they include the conditions of the corresponding protocol steps according to the protocol rules. For a protocol $\mathcal{P}$ with n steps, we have to provide the definition of n predicates $\mathcal{P}_1,\ldots,\mathcal{P}_n$. There is additionally the predicate $Oops_{ev}$, which allows us to model *dynamic corruption* where certain secrets are disclosed after being used in previous protocol runs or events. The definition of $Oops_{ev}$ is expected to bind the (dynamically) disclosed secrets with (partial or completed) prior protocol runs. Examples are in Sec. 8.1.2 and 9.1.2.

The VSE-theory `t_`$\mathcal{P}$ includes not only the definitions of $\mathcal{P}$, $\mathcal{P}_1,\ldots,\mathcal{P}_n$, and $Oops_{ev}$, but also axioms about the available initial knowledge items and corresponding assumptions. Examples are in Sec. 8.1.1 and 9.1.1.

7.1.7 Protocol Properties

As introduced in Sec. 7.1.6, we reserve the VSE-theory `t_`$\mathcal{P}$`_properties` for the protocol properties that we want to verify. In addition to confidentiality, authentication, and other trace properties, which serve as (proof-)structuring lemmata, we are also interested in in-

distinguishability properties. But, we focus in this section on the trace properties. The latter are discussed in part III of the thesis.

7.1.7.1 Confidentiality Properties

The confidentiality of a message $\hat{s}$ means that this message does not belong to the intruder knowledge. It is formalized according to the following schema:

$$(tr \in \mathcal{P} \wedge \Phi(\hat{s}, tr)) \Rightarrow \hat{s} \notin DY(spies(tr))$$

The premise $\Phi(\hat{s}, tr)$ puts $\hat{s}$ into context with protocol runs. It determines the protocol steps where $\hat{s}$ or parts of $\hat{s}$ are exchanged between the *regular* participants who are intended to *exclusively* obtain $\hat{s}$. Achieving that $\hat{s}$ is available during or at the end of the protocol run *only for* the intended participants necessitates to protect (at least some of) the transmitted message parts that are required for its derivation. As mentioned in Chap. 6, the confidentiality proof is conducted according to the structure of $\hat{s}$ and to the protection of its message parts in regular protocol messages:

1. When $\hat{s}$ and its protected substructures are not derivable by merging operations, we investigate the additional selected secrets that are necessary to protect the confidentiality against derivations by decrypt-type operations from regular protocol messages. If these additional secrets and their protected substructures are as well not derivable by merging operations, we are able to provide sets S of selected secrets satisfying $\Re_1(S)$ for a confidentiality proof by the canonical basic check-function ccl_1 (cp. the use of ccl_1 in Sec. 6.2 and 6.3). Here, the confidentiality of $\hat{s}$ is proven by applying a corresponding lemma, which is of the form

   $$(tr \in \mathcal{P} \wedge \Phi(\hat{s}, tr)) \Rightarrow (\exists S : \Re_1(S) \wedge spies(tr) \cap ccl_1(S) = \emptyset),$$

   where $\Re_1(S)$ corresponds to the additional (closure) conditions on S required to employ the correctness theorem of the used ccl_1 (cp. theorem 46 for ccl_1 in PACE).

2. For a composed message $\hat{s}$ that can be derived out of protocol messages *only* by composition, we reduce the confidentiality of $\hat{s}$ to the confidentiality of message parts required for its composition. This is done using the canonical check-function ccl_2 and the generic reduction theorem appropriate to the type of $\hat{s}$ (see Sec. 6.4), together with a corresponding lemma of the form:

   $$(tr \in \mathcal{P} \wedge \Phi(\hat{s}, tr) \wedge \Psi(\hat{s}, DY(spies(tr)))) \Rightarrow spies(tr) \cap ccl_2(\hat{s}) = \emptyset$$

   $\Psi(\hat{s}, DY(spies(tr)))$ expresses that for each alternative to compose $\hat{s}$ one of the required message parts is not available in $DY(spies(tr))$. It is obtained, as described in Sec. 6.4.1, by a straightforward transformation of the additional condition in the generic reduction theorem appropriate to the type of $\hat{s}$.

   If none of the resulting substructures belongs to the basic confidential message parts, which are already (or can be) shown according to (1), the proof task consists in selecting a message part s' to handle by reduction (if possible) or as described below.

   In certain (confidentiality) properties, the handling of $\hat{s}$ is by induction on the top-structure of $\hat{s}$. Here, the reduction theorem yields in the step case some s' permitting to conclude by the induction hypothesis. In the base case, the substructures resulting by the reduction theorem need to be treated as described above, i.e. by identifying a basic confidential message part or by selecting some s' to be handled again by reduction or as described below.

3. The mentioned reduction technique in (2) is not applicable when $\hat{s}$ could be derived by a merging operation. In this case, we try to prove the confidentiality of $\hat{s}$ with the help of a protocol-specific invariant that provides a complete case distinction on all derivable messages $\hat{m}$ having the same type like $\hat{s}$ (see Sec. 6.5.1). In particular, this invariant links $\hat{m}$ to the public regular message parts and identifies their confidential substructures. If all these basic confidential substructures can be established as described in (1), the invariant may make use of these properties. Otherwise, we have basic confidential message parts derivable by merging, which necessitates the use of another basic check-function for their proof (see below) before using these results as part of our invariant.

   Having an appropriate invariant, the confidentiality of $\hat{s}$ is proved by showing that none of the derivable $\hat{m}$ matches $\hat{s}$ (cp. the example in Sec. 6.5.1.3).

4. If the confidentiality proof of $\hat{s}$ necessitates to use selected secrets that are derivable by merging, we need a new basic check-function permitting such selected secrets. Here, the use of ccl_3 for our example in Sec. 6.5.2 provides good guidance, to define an analog basic check-function and to prove its correctness in a similar way.

7.1.7.2 Authentication Properties

The authentication by some protocol participant ag is principally formalized from her point of view. The premise fixes the events by ag in some protocol run until the state is reached where the authentication guarantee can be checked. It is generally an equality check of a received message (part) $m_{\overline{ag}}$ with the corresponding counterpart m_{ag}, as can be computed by ag. It implies the *authenticity* of the received message $m_{\overline{ag}}$: The message $m_{\overline{ag}}$ originates from the peer $\overline{ag}$ who was somehow involved with ag in the same protocol run. The conclusion fixes then the events by $\overline{ag}$, including the originating event of $m_{\overline{ag}}$.

Basically, authenticity properties can be distinguished according to the strength of the corresponding guarantees: The tighter the guarantee, the stronger the authenticity property. We want to judge the strength of the considered authenticity properties relative to the following features:

1. The peer $\overline{ag}$ equals the protocol partner addressed by ag, or is arbitrary.
2. The peer $\overline{ag}$ addresses her messages to ag, or to a participant who may be different.
3. The events by the peer $\overline{ag}$ are associated with the intended protocol role *only*, or a second interpretation equivalent with the role of ag is possible.
4. Whether guarantees about the structure and the content of the messages originating from the peer $\overline{ag}$ are provided.

An authentication guarantee from the point of view of ag where the peer $\overline{ag}$ is not arbitrary and talks to ag in a fixed role is generally formalized according to the schema:

$$(tr \in \mathcal{P} \wedge \Phi(ag,\overline{ag},nc,tr) \wedge gets(ag,m[nc]) \in tr) \Rightarrow$$
$$(\Psi(\overline{ag},ag,nc,tr) \wedge says(\overline{ag},ag,m[nc]) \in tr)$$

$\Phi(ag,\overline{ag},nc,tr)$ includes prior events by ag where the nonce nc, which identifies a new run, is transmitted to the peer $\overline{ag}$. Similarly, $\Psi(\overline{ag},ag,nc,tr)$ includes events by $\overline{ag}$ where she receives the message containing nc and uses this to generate the reply message $m[nc]$ for the (last) *says* event.

Usually, we prove any authentication property immediately using the corresponding authenticity property. It is formalized by a simple adaptation of the premise: The occurrence of a gets event in the trace ($gets(.,m) \in tr$) is substituted with the occurrence of the

received message in the corresponding observable messages ($m \in spies(tr)$). The equivalent authenticity property of the above authentication schema is:

$$(tr \in \mathcal{P} \wedge \Phi'(ag, \overline{ag}, nc, tr) \wedge m[nc] \in spies(tr)) \Rightarrow$$
$$(\Psi(\overline{ag}, ag, nc, tr) \wedge says(\overline{ag}, ag, m[nc]) \in tr)$$

$\Phi'(ag, \overline{ag}, nc, tr)$ results by the replacement of the *gets* events in $\Phi(ag, \overline{ag}, nc, tr)$.

Generally, authenticity properties have the most involved proof argument in the class of the trace properties. It combines (basic) confidentiality properties with other trace properties that we call structuring lemmata, introduced below. The combination of the mentioned properties is required in particular to handle the fake case, where we prove that the attacker is not able to *forge* the authenticated message.

7.1.7.3 Structuring Lemmata

Except of confidentiality, authentication and authenticity properties all other kinds of trace properties serve generally to structure the proofs. The original inductive method introduced a categorization of the latter properties. For our purposes we employ mainly (i) binding (unicity) theorems and (ii) regularity lemmata. Briefly, binding properties are about the structure and the content of message parts that occur *together* with secure *nonces*. And regularity properties are guarantees about certain message parts that rely on the initial knowledge of *regular* protocol participants and on message generation by them.

The separation between binding theorems and regularity lemmata is not strict, and certain binding theorems can be also interpreted as regularity lemmata. We will provide certain examples for both kinds of properties in Sec. 8.5, 9.5.2, 9.6, 12.2 and 14.2.

Noteworthy, our implementation of the original inductive method in VSE allows for the automated transformation of protocol specifications from a specific Alice-and-Bob form (in a CAPSL-like language, [97]) into VSE-SL. It generates not only the definition of the protocol rules, but also the formalization of the properties and many structuring lemmata. But, the very involved lemmata like binding theorems about message parts across several protocol steps need to be provided during proof construction. Nevertheless, they could be (partly) speculated through the search for corresponding patterns in proof states.

Structuring lemmata can be also produced as a by-product during the application of certain proof strategy. This will be the case in our proof technique applied to indistinguishability properties, where a central theorem provides the necessary and sufficient conditions. These conditions are then shown with the help of regularity properties on different types of derivable messages and the protection of their substructures. See Chap. 12 and 14.

7.2 Proof Construction

In order to verify the protocol properties in `t_`$\mathcal{P}$`_properties`, we start the VSE prover by clicking on the "satisfies"-link to `t_`$\mathcal{P}$. The prover registers these properties as proof obligations, for which complete proofs in the (VSE) sequent calculus have to be constructed. For that purpose, the VSE prover allows the user

- to select basic inference rules,
- apply lemmata and axioms,
- and to invoke powerful simplification routines.

Simplification routines, e.g., `predlogic` for bringing proof goals in a canonical form, usually combine several inference rules *in one step*. Despite their frequent use proofs of protocol properties are very large. This fact and the number of the proof obligations definitely show that the inductive method is only successful in real-world applications if the burden of user

interaction is lowered drastically. Therefore we extended the VSE prover with *proof heuristics* exploiting the higher-level knowledge available from a systematic analysis of the given class of proof obligations.

Our approach is bottom up in the sense that starting with routine tasks whose automatization just saves some clicks we proceed by building heuristics that cover more complex proof decisions using the lower level ones as primitives. This complies with the tradeoff between achieving the verification of the protocols within an acceptable period and enhancing the automatization degree through the integration of more proof heuristics in the VSE prover. Hence, our primary goal was not to focus (from the beginning) on the fully automated verification of certain protocol classes. Instead of that, we integrated as much heuristics in the VSE prover as timely possible in our verification projects, preferring those heuristics that are oftener needed.

Except of few protocol properties, which are proven immediately with corresponding lemmata, the majority are verified by induction on protocol traces. The general scheme for structural induction on the traces of a particular protocol, shown in Fig. 7.5, is more or less straightforward. The heuristics discussed in the following implement an application-specific refinement of this scheme. This refinement takes into account both the type of the proof goals and the available lemmata and axioms. We want to describe the top-level proof scheme and some of the often used heuristics.

7.2.1 The Top-Level Proof Scheme

All inductive proofs of protocol properties are structured into the following (proof-) tasks. For each task there is a collection of heuristics that are potentially applicable in these situations.

1. Set up a proof by structural induction on traces.
2. Handle the base case.
3. Handle the step case:
   (a) Reduce the *protocol-independent* differences.
   (b) Add information about individual protocol steps.
   (c) Reduce the remaining differences and apply the induction hypothesis.

An inductive proof about traces of a given protocol is initialized by a heuristic (`traceInd`) which selects the induction variable representing the protocol trace. Afterwards, it transforms the proof goals representing the base case and the step case to a canonical form through the application of `predlogic` and other simplification inference rules.

Base Case: The proof goals of the base case are expected to contain assumptions that contradict properties of the empty trace ϵ. For that reason, we handle the base case by searching for such an assumption and then applying the corresponding lemmata and axioms to close the proof goal by contradiction. This is carried out by the heuristic `emptyTr`.

Step Case: Like in all other mechanized induction systems we try to reduce the given goals to a situation where the inductive hypothesis can be applied. In our proof scheme we separate the difference reduction to the induction hypothesis in two phases and include the protocol details in between. To explain this idea, we consider a proof goal

$$IH, \gamma_1, \ldots, \gamma_q \vdash \delta_1, \ldots, \delta_r,$$

in the canonical form as produced by `traceInd`. We have on the left-hand side the induction hypothesis IH and further hypotheses as *antecedents*, and on the right-hand side the

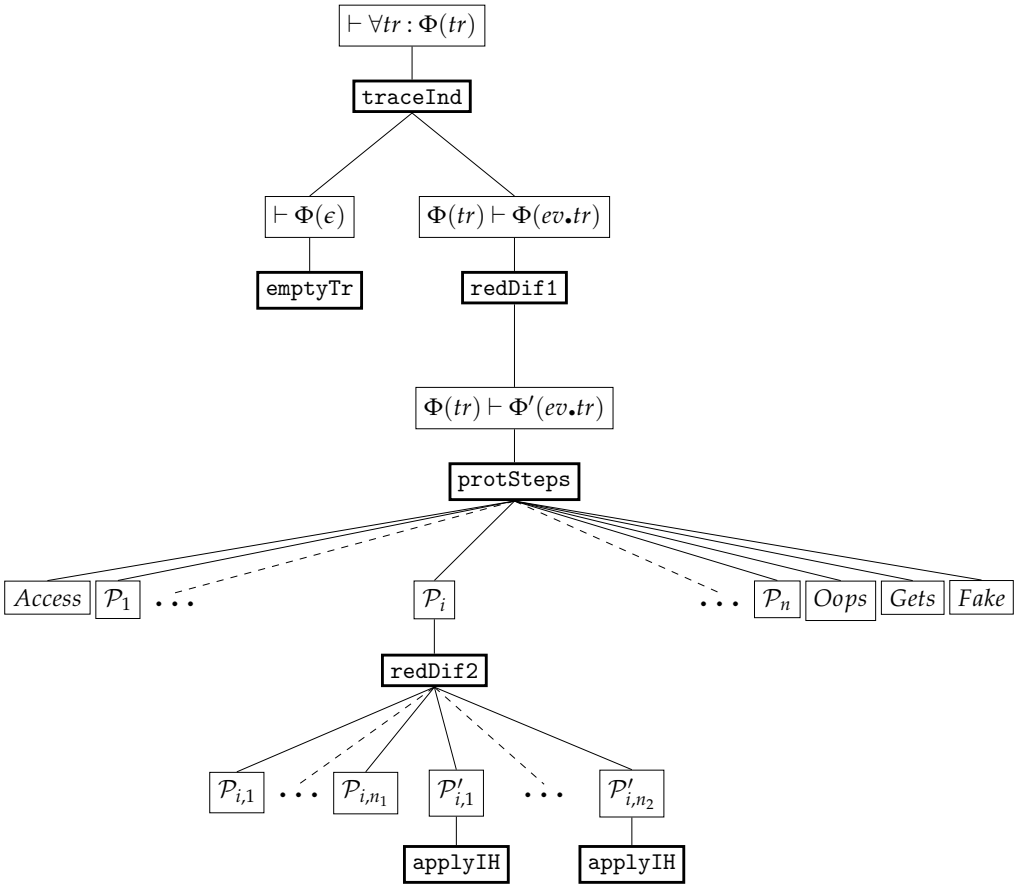

Figure 7.5: The top-level proof scheme

possible conclusions as *succedents*. For such a open goal we need to provide a proof for a succedent δ_i out of the antecedents, possibly using negations of the other succedents as well as available axioms and lemmata.

Reduce Protocol-Independent Differences: Protocol properties include generally premises that restrict the set of traces under consideration by excluding certain events or messages. For instance, most of the properties in PACE are defined about traces that do not include the access to the password of the card by a compromised terminal. This is formulated by the premise $(\forall ag : note(ag, \langle A, pwd(A), G\rangle) \in tr \Rightarrow ag \notin bad)$. We expect that every premise $\phi(tr)$ of this kind matches (modulo difference reduction) an antecedent $\gamma_i = \phi(ev.tr)$ or a negated succedent $\neg\delta_j = \phi(ev.tr)$. Furthermore, we know that the difference can be eliminated without considering any details about the event ev added to the trace (induction variable) tr. For that reason, we start the difference reduction (before including the protocol details) focusing on the antecedents with the mentioned features. This is done by the heuristic `redDif1`.

Include Protocol Details: In the next steps we exploit the assumptions under which a certain event ev can be added to a trace tr. This is achieved by a heuristic called `protSteps`. It applies first Def. 70 to the assumption $(ev.tr) \in \mathcal{P}$, before carrying out the corresponding *case split*. Next the conditions for each particular extension, as given by the corresponding generic or protocol-specific definition, are added to the goals. These are transformed finally

to a canonical form through the application of `predlogic` and other simplification inference rules.

Reduce Remaining Differences and Apply Induction Hypothesis: In each of the resulting proof goals we focus on the antecedents and the succedents that are of the form $\phi(ev.tr)$. We expect that every antecedent $\phi(ev.tr)$ occurs (after difference reduction) as a premise or in the negated conclusion of the induction hypothesis. For that reason, we eliminate first the corresponding differences (by the heuristic `redDif2`) aiming at antecedents $\phi(tr)$ that match their counterparts in the induction hypothesis.

In contrast to the canonical difference reduction before the inclusion of the protocol details, difference reduction in this phase amounts to two kinds of subgoals:

1. The majority of the resulting subgoals can be immediately handled by the application of the induction hypothesis. This is achieved by the heuristic `applyIH`, which provides the instantiation for the universally quantified variables. It tries afterwards to close the subgoal by `predlogic` and by appropriate lower level heuristics (see Sec. 7.2.2.4).

2. In the complementary cases it is necessary to apply appropriate *structuring lemmata* and to make use of specific *goal assumptions*. These goal assumptions can originate from the definition of the corresponding protocol step (by `protSteps` in (3-b)) or from the applied lemmata or axioms during the difference reduction by `redDef2`. The treatment of these subgoals requires user interaction. It often involves proof decisions, like:

   - Which goal assumptions should be considered?
   - Which structuring lemmata have to be applied?
   - Do we need a *new* structuring lemma?

However, also in those cases where *some* user interaction is necessary, heuristics are available to continue (and complete) the proof afterwards.

7.2.2 The Proof Heuristics

We extended the VSE prover in a first phase progressively with proof heuristics that handle localized tasks in our proof scheme. From the beginning, we put emphasis on implementation features that facilitate the composition and adaptation of our heuristics. Due to the local (wrt. single goals) and global (wrt. the proof tree) information slots provided by the VSE prover, it was possible to store the intermediate application states of single heuristics using simple information labels. So, our heuristics have access not only to the purely logical information, but also to the locally and globally stored labels that allow us to decide about the next step to be performed. This includes also the switch to another heuristic, which continues the proof construction using corresponding information labels as parameters. Thus, we used the information labels to model the internal control flow of a single heuristic as well as to organize the composition of heuristics.

Meanwhile, we implemented all the heuristics introduced in Sec. 7.2.1 and we composed them in a high-level heuristic `commonStps`, which constructs the major canonical proof part. This heuristic saves up in average more than 80 % of the required clicks in a step-by-step mode. The resulting subgoals require interactions of the user, who continues the proof construction in a step-by-step mode where further heuristics can be invoked. For this purpose, all heuristics composed in `commonStps` and further low-level heuristics are available for selection. In the rest of this section we give an overview on these low-level heuristics. Through their rigorous use, we reached an automation degree over 95 %.

7.2.2.1 The heuristic `emptyTr`

It is invoked by `commonStps` to handle the base case as explained in Sec. 7.2.1. It is composed (in the time being) out of two low-level heuristics:

1. `nullEvt` searches for an antecedent of the form $ev \in \epsilon$, stating that the event ev belongs to the empty trace. It contradicts obviously an axiom about traces, which belongs to the VSE-theory `tprotocoltrace`, shown in Fig. 7.2. Upon matching of $ev \in \epsilon$, `nullEvt` applies this axiom with the corresponding instantiation.

2. `nullMsg` searches for an antecedent of the form $m \in spies(\epsilon)$, stating that the message m occurs in the empty trace. It contradicts obviously the property that $x \notin spies(\epsilon)$, which belongs to the VSE-theory `ttrace_thms` shown in Fig. 7.3. Upon matching of $m \in spies(\epsilon)$, `nullMsg` applies this property with the corresponding instantiation.

Note, that this version of `emptyTr` is also provided in the VSE framework implementing the original inductive method.

7.2.2.2 The heuristic `redDif1`

It is invoked by `commonStps` in the first difference reduction phase, as explained in Sec. 7.2.1. The aim of this heuristic is to reduce antecedents and negated succedents to the premises (of the induction hypothesis) that exclude certain events or messages. If we denote the occurrence of an event or a message by ϕ, we want to cover all premises $\Psi[\phi]$ that correspond to a negation $\neg\phi$, a disjunction $(\neg\phi \vee \Psi')$, or to an implication $\phi \Rightarrow \Psi'$, possibly preceded by a universal quantification. This is carried out by the following low-level heuristics, which are invoked in the given order.

1. `evtNotInTr` searches for an antecedent or a negated succedent $\Psi[ev' \in (ev\bullet tr)]$ that can be associated with a premise $\Psi[ev' \in tr]$ of the induction hypothesis. Upon matching, it applies corresponding inference rules (according to Ψ) and an axiom from the VSE-theory `tprotocoltrace`, shown in Fig. 7.2, to eliminate the difference.

2. `msgNotInTr` searches for an antecedent or a negated succedent $\Psi[m \in spies(ev\bullet tr)]$ that can be associated with a premise $\Psi[m \in spies(tr)]$ of the induction hypothesis. Upon matching, it applies corresponding inference rules (according to Ψ) and a theorem from the VSE-theory `ttrace_thms`, shown in Fig. 7.3, to eliminate the difference.

3. The low-level heuristic `msgNotInDy` searches for an antecedent or a negated succedent $\Psi[m \in DY(spies(ev\bullet tr))]$ that can be associated with a premise $\Psi[m \in DY(spies(tr))]$ of the induction hypothesis. Upon matching, it applies corresponding inference rules (according to Ψ) and a theorem from the VSE-theory `tindy_thms`, shown in Fig. 7.3, to eliminate the difference.

7.2.2.3 The heuristic `redDif2`

It is invoked by `commonStps` in the second difference reduction phase, as explained in Sec. 7.2.1. The aim of this heuristic is to reduce the remaining differences in antecedents to corresponding premises or negated conclusions of the induction hypothesis. For the time being, we handle three typical kinds of antecedents by corresponding low-level heuristics, which are invoked in the given order.

1. `evtInTr` searches for an antecedent $ev' \in (ev\bullet tr)$ that has to be reduced to a premise $ev' \in tr$ in the induction hypothesis. Upon matching, it applies a corresponding axiom from the VSE-theory `tprotocoltrace`, shown in Fig. 7.2. This yields a case split, with a first subgoal where the difference is eliminated and a second subgoal corresponding to $ev' = ev$. `evtInTr` tries to close the latter by invoking a corresponding low-level heuristic.

2. `msgInTr` searches for an antecedent $m \in spies(ev.tr)$ that has to be reduced to a premise $m \in spies(tr)$ in the induction hypothesis. Upon matching, it applies a corresponding theorem from the VSE-theory `ttrace_thms`, shown in Fig. 7.3. In addition to a subgoal where the difference is eliminated, this results often in other subgoals corresponding to $m = m_{ev}$, for an event message m_{ev}. They are tried to be closed by invoking a corresponding low-level heuristic (see below).

3. `msgInCcl` searches for two antecedents $m \in spies(ev.tr)$ and $m \in ccl(mL)$ that have to be reduced to a conclusion $\forall m : m \in spies(tr) \Rightarrow m \notin ccl(mL)$ in the induction hypothesis. Upon matching, it applies the same theorem from the VSE-theory `ttrace_thms` as in `msgInTr`. We obtain a subgoal where the difference is eliminated, and often other subgoals where the antecedent $m \in ccl(mL)$ is replaced with $m_{ev} \in ccl(mL)$, for an event message m_{ev}. They are tried to be closed by invoking a corresponding low-level heuristic (see below).

The third heuristic is required in the proof of lemmata employed for confidentiality properties. The matched propositions originate actually from $spies(tr) \cap ccl(mL) = \emptyset$, which we usually formalize by $\forall m : m \in spies(tr) \Rightarrow m \notin ccl(mL)$.

7.2.2.4 The heuristic `applyIH`

It is invoked by `commonStps` in the subgoals of `redDif2` where the difference reduction succeeded. After the application of the induction hypothesis, including the instantiation of the universally quantified variables, it invokes `predlogic`. This might result in subgoals where further difference reduction is required, in particular when the verified property is existentially quantified. Such subgoals are tried to be closed with one of the following low-level heuristics:

1. `closeEvt` searches for an antecedent $ev \in tr$ and a succedent $ev \in (ev'.tr)$. Upon matching, it closes the subgoal with the help of a corresponding axiom from the VSE-theory `tprotocoltrace`, shown in Fig. 7.2.

2. The low-level heuristic `closeMsg` searches for an antecedent $m \in obset(ag,tr)$ and a succedent $m \in obset(ag,ev'.tr)$. Upon matching, it closes the subgoal with the help of a corresponding theorem from the VSE-theory `ttrace_thms`, shown in Fig. 7.3.

3. `closeMsgDY` searches for an antecedent $m \in DY(obset(ag,tr))$ and a corresponding succedent $m \in DY(obset(ag,ev'.tr))$. Upon matching, it closes the subgoal with the help of a corresponding theorem from the VSE-theory `tindy_thms`, shown in Fig. 7.3.

7.2.2.5 Further Low-Level Heuristics

The above introduced heuristics are partly composed from other low-level heuristics. For instance, the heuristic `msgInCcl`, which is usually applicable in confidentiality lemmata, obtains subgoals (with an antecedent $m_{ev} \in ccl_i(mL)$) that require the application of the definition of the *ccl*-function ccl_i (see Chap. 6). This is achieved by a heuristic `inCcl`, which applies the corresponding definition of ccl_i from the VSE-theory `tccl`, shown in Fig. 7.3. After a case split it handles the obtained subgoals either through a recursive call or by the invocation of other low-level heuristics. Typical subgoals where other low-level heuristics are invoked contain antecedents of the form $m_{ev} \in mL$ or $m_{ev} = m$. In the former case we invoke the heuristic `inLst` and in the latter case the heuristic `eqMsg`. The low-level heuristic `inLst` fetches the messages m_{mL} from the list mL successively (by the application of the corresponding definition from the VSE-theory `tmsglist`, shown in Fig. 7.1), yielding subgoals with equalities $m_{ev} = m_{mL}$ instead of the antecedent $m_{ev} \in mL$. All these subgoals are then closed or simplified by invoking `eqMsg`.

The low-level heuristic `eqMsg` is the most used one in the proof construction, usually within other heuristics. It applies our axioms and derived theorems about message structures to close or simplify proof goals with equalities as antecedents (see Sec. 5.5.2). It is parametrized through a list of triples, where each triple associates two term patterns with an axiom or a theorem. According to the preference fixed by this list, `eqMsg` searches then in the antecedents for an equality that matches the term patterns. Afterwards it applies the axiom or the theorem given by the corresponding triple. Using such a parameter list, facilitates the adaptation and the (progressive) extension of this central low-level heuristic.

Chapter 8

Verification of PACE's Trace Properties

In this chapter we give an overview on the verification of the trace properties of PACE in the VSE system. We start with an excerpt from the specification of the protocol model. Then we describe the proofs of the main protocol properties: the basic confidentiality property, the mutual authentication, and the forward secrecy of session key. We focus in particular on the formalization and we sketch the proof ideas as well as the use of our proof technique, described in Chap. 6.

8.1 Protocol Model

Following the conventions in Sec. 7.1.6, the set PACE, which contains all possible PACE traces, is defined according to the schema 70. Besides the generic cases, we obtain eight protocol-specific cases: $\text{PACE}_1(ev,tr)$–$\text{PACE}_7(ev,tr)$ cover the admissible trace extensions according to PACE and $Oops_{ev}(ev,tr)$ models dynamic corruption. In this section, we discuss the definition of these predicates. Before that, we describe the protocol-specific details regarding the initial knowledge.

8.1.1 Initial Knowledge

In Sec. 7.1.4 we introduced the predicate $initHas(ag,m)$ to express what participants initially know. We already mentioned that further axioms are required to specify the protocol-specific assumptions about the initial knowledge. In case of PACE, we use the following axioms.

Axiom$^{\text{VSE}}$ 72 (initHas; PACE):

1.
$$initHas(ag,m) \Rightarrow (\exists g : m = \langle pwd(ag), g\rangle \\ \vee (\exists ag' : ag \neq ag' \wedge m = \langle ag', pwd(ag'), g\rangle))$$

2.
$$(initHas(ag,\langle pwd(ag),g\rangle) \wedge initHas(ag',\langle ag,pwd(ag),g'\rangle)) \Rightarrow g = g'$$

3.
$$\exists g : initHas(ag,\langle pwd(ag),g\rangle)$$

4.
$$\exists ag, ag', g : ag \neq ag' \wedge initHas(ag, \langle ag', pwd(ag'), g \rangle)$$

The initial knowledge consists of pairs $\langle pwd(ag), g \rangle$ and triples $\langle ag', pwd(ag'), g \rangle$. To run PACE in the card role, ag needs to access $\langle pwd(ag), g \rangle$, i.e. the own password and generator.

In order to run PACE in the terminal role, ag has to access $\langle ag', pwd(ag'), g \rangle$. Note that the participant ag' (in the card role) *differs* from ag. This way, we have a kind of role separation, in the sense that no agent runs PACE with himself as a partner (see Sec. 8.3 and 8.4).

By the second axiom, every password is associated with the same generator, and that from the perspective of all agents. Every agent can run PACE in the card role and there is at least one agent that is able to run PACE in the terminal role. These axioms are needed to prove that the specification of the PACE rules is not restrictive. That is, complete PACE runs are not excluded through the conditions used in our formalization of these rules.

8.1.2 PACE Rules

In this section we discuss the specification of the PACE rules. The first rule includes the condition for initiating a new run by a card.

Definition$^{\text{VSE}}$ 73 (PACE_1):

$$\begin{aligned}&\text{PACE}_1(ev, tr) \Leftrightarrow \\ &(\exists A, B, g, s : A \neq B \wedge note(A, \langle pwd(A), g \rangle) \in tr \\ &\quad \wedge \neg used(s, tr) \wedge ev = says(A, B, enc(pwd(A), s)))\end{aligned}$$

A accesses the own password and generator before sending message 1.

Definition$^{\text{VSE}}$ 74 (PACE_2, PACE_3):

1.
$$\begin{aligned}&\text{PACE}_2(ev, tr) \Leftrightarrow \\ &(\exists B, A, g, z, x_1 : B \neq A \wedge note(B, \langle A, pwd(A), g \rangle) \in tr \\ &\quad \wedge gets(B, z) \in tr \wedge \neg used(x_1, tr) \wedge ev = send(B, A, \langle z, x_1 \rangle, dh(g, x_1)))\end{aligned}$$

2.
$$\begin{aligned}&\text{PACE}_3(ev, tr) \Leftrightarrow \\ &(\exists A, g, B, s, X_1, y_1 : note(A, \langle pwd(A), g \rangle) \in tr \wedge says(A, B, enc(pwd(A), s)) \in tr \\ &\quad \wedge gets(A, X_1) \in tr \wedge \neg used(y_1, tr) \wedge ev = send(A, B, \langle s, y_1 \rangle, dh(g, y_1)))\end{aligned}$$

Before sending message 2, B must have accessed the password and the generator of a card A and also must have received a message z that is expected to be a nonce encrypted by A. The recorded information in the *send* event allows us

1. to express *explicitly* that B knows the nonce x_1, and
2. to *bind* step 2 with the received message z.

If step 2 were formalized by a *says* event, the set of observable messages by B would neither contain x_1 nor permit us to derive this nonce by algebraic reasoning.

Similarly, step 3 is described by a *send* event where the nonce s from step 1 and the new nonce y_1 are recorded. The reasons for binding step 2 with z and step 3 with s are related to the definitions of steps 4 and 5, respectively.

Definition$^{\mathbf{VSE}}$ 75 (PACE$_4$, PACE$_5$):

1.

$$\begin{aligned}
&\text{PACE}_4(ev,tr) \Leftrightarrow \\
&(\exists B,A,g,z,x_1,Y_1,x_2 : note(B,\langle A,pwd(A),g\rangle) \in tr \wedge gets(B,z) \in tr \\
&\quad \wedge send(B,A,\langle z,x_1\rangle,dh(g,x_1)) \in tr \wedge gets(B,Y_1) \in tr \wedge \neg used(x_2,tr) \\
&\quad \wedge ev = send(B,A,x_2,dh(gen(dh(g,dec(pwd(A),z)),dh(Y_1,x_1)),x_2)))
\end{aligned}$$

2.

$$\begin{aligned}
&\text{PACE}_5(ev,tr) \Leftrightarrow \\
&(\exists A,B,s,X_1,g,y_1,X_2,y_2 : says(A,B,enc(pwd(A),s)) \in tr \wedge gets(A,X_1) \in tr \\
&\quad \wedge send(A,B,\langle s,y_1\rangle,dh(g,y_1)) \in tr \wedge gets(A,X_2) \in tr \wedge \neg used(y_2,tr) \\
&\quad \wedge ev = send(A,B,y_2,dh(gen(dh(g,s),dh(X_1,y_1)),y_2)))
\end{aligned}$$

In steps 4 and 5, the generator for the second Diffie-Hellman exchange is computed using the generator g, the nonce s from step 1 and the common Diffie-Hellman value that is established in steps 2 and 3. Note that B obtains s by decrypting the received z.

Actually, the conditions for step 4 do not impose an explicit chronological order of the preceding events. Such an order is implicitly given by the content of the events. By recording z in the *send* event that represents step 2, it follows that $gets(B,z)$ occurred prior to this step. If step 2 were formalized by a *says* event, we would not have this implicit ordering condition. In this case, the generator could be computed using an out-of-date Diffie-Hellman value, that is established in (steps 2 and 3 of) a previous PACE run, i.e. before receiving z. This problem is solved by binding z with step 2. Binding s with step 3 avoids the same problem from the perspective of the card A.

Definition$^{\mathbf{VSE}}$ 76 (PACE$_6$, PACE$_7$):

1.

$$\begin{aligned}
&\text{PACE}_6(ev,tr) \Leftrightarrow \\
&(\exists B,A,g,z,x_1,Y_1,x_2,Y_2 : note(B,\langle A,pwd(A),g\rangle) \in tr \wedge gets(B,z) \in tr \\
&\quad \wedge send(B,A,\langle z,x_1\rangle,dh(g,x_1)) \in tr \wedge gets(B,Y_1) \in tr \\
&\quad \wedge send(B,A,x_2,dh(gen(dh(g,dec(pwd(A),z)),dh(Y_1,x_1)),x_2)) \in tr \\
&\quad \wedge gets(B,Y_2) \in tr \\
&\quad \wedge (\forall x : send(B,A,x,dh(gen(dh(g,dec(pwd(A),z)),dh(Y_1,x_1)),x)) \in tr \\
&\qquad\quad \Rightarrow Y_2 \neq dh(gen(dh(g,dec(pwd(A),z)),dh(Y_1,x_1)),x)) \\
&\quad \wedge ev = says(B,A,mac(dh(Y_2,x_2),Y_2)))
\end{aligned}$$

2.

$$
\begin{aligned}
&\text{PACE}_7(ev,tr) \Leftrightarrow \\
&(\exists A,B,s,X_1,g,y_1,X_2,y_2 : says(A,B,enc(pwd(A),s)) \in tr \wedge gets(A,X_1) \in tr \\
&\quad \wedge send(A,B,\langle s,y_1\rangle,dh(g,y_1)) \in tr \wedge gets(A,X_2) \in tr \\
&\quad \wedge send(A,B,y_2,dh(gen(dh(g,s),dh(X_1,y_1)),y_2)) \in tr \\
&\quad \wedge gets(A,mac(dh(X_2,y_2),dh(gen(dh(g,s),dh(X_1,y_1)),y_2))) \in tr \\
&\quad \wedge ev = says(A,B,mac(dh(X_2,y_2),X_2)))
\end{aligned}
$$

B rejects the received message Y_2 in step 5, if this matches message 4, [60]. Basically, our formalization of step 4 in axiom 75 does not prevent B to repeat this step several times in a run. That is, it is possible that message 4 is generated, for instance twice by B using the same generator but different nonces. This explains the universally quantified form of the used condition in the definition of PACE_6.

In [60], the technical specification of PACE requires that A does not accept the received message X_2 in step 4, if A finds out (in the reply to step 6) that X_2 matches message 5. As pointed out in Sec. 2.3.1, this inequality check by A is redundant, since A generates message 5 after accepting X_2 and the equality of both values is excluded using the new nonce y_2. For this reason, we simplify the inequality check in our formalization of step 7. Note that the corresponding conditions include an explicit verification of the received MAC value in step 6 (cp. the *mac*-message in the *gets* event).

In addition to the regular protocol steps, we model the accidental loss of session keys for a stronger form of resistance against offline password testing (see Chap. 12).

Definition$^{\text{VSE}}$ 77 (Oops_{ev}; PACE):

$$
\begin{aligned}
&Oops_{ev}(ev,tr) \Leftrightarrow \\
&(\exists B,A,x_2,g,z,m,x_2,y_2 : send(B,A,x_2,dh(gen(dh(g,dec(pwd(A),z)),m),x_2)) \in tr \\
&\quad \wedge send(A,B,y_2,dh(gen(dh(g,dec(pwd(A),z)),m),y_2)) \in tr \\
&\quad \wedge ev = note(spy,dh(dh(gen(dh(g,dec(pwd(A),z)),m),y_2),x_2)))
\end{aligned}
$$

In Sec. 8.4 we will see that there is no guarantee for a terminal B that the communication partner is running PACE in the card role. Even more, the used message z must not be an encrypted nonce. Fortunately, this does not violate the confidentiality of the session key (see Sec. 8.6). Session keys remain confidential also when both communication partners have run PACE in the terminal role, using any common message z. In the given definition, the lost session key could be established between two terminals, as well as between a card and a terminal.

8.2 Basic Confidentiality Properties

The main objective of PACE is the *authenticated* establishment of a session key between a card A and a terminal B.

This requires in particular the mutual authentication of both participants, which comprises typically one authentication guarantee from the point of view of A (see Sec. 8.3) and another authentication guarantee from the point of view of B (see Sec. 8.4). Both authentication guarantees are not identical (in case of our formal model of PACE).

The authentication guarantees in PACE rely on *the confidentiality of the fresh generator*. Only the participants that are able to compute *the fresh generator* can participate in the DH

exchange of steps 4 and 5 to obtain the common DH value used as MAC-key in the subsequent steps 6 and 7. The fresh generator cannot be composed without obtaining the first argument of gen. In the authentication guarantee from the point of view of a card A, this message part is computed by $dh(G, N_1)$, where the nonce N_1 is sent encrypted to the peer using the password $pwd(A)$. Whereas in the authentication guarantee from the point of view of a terminal B, the first argument of gen is computed by $dh(G, dec(pwd(A), M_1))$, where M_1 is an arbitrary message sent by the peer. This includes also the case where M_1 differs from any encryption with $pwd(A)$ as a key.

Recapitulating, we need the confidentiality of two kinds of fresh generators for the proof of the authentication guarantees by PACE:

1. In both authentication guarantees, we will have proof situations where the fresh generator matches $gen(dh(G, N_1), M)$.
2. The proof of the authentication guarantee from the point of view of B comprises additional cases where the fresh generator matches $gen(dh(G, dec(pwd(A), M_1)), M)$ for some peer's message M_1 that differs from any encryption with $pwd(A)$.

This yields to the following basic confidentiality property.

Property$^{\mathbf{VSE}}$ 78 (Basic Confidentiality):

1. $$\begin{aligned}&(tr \in \text{PACE} \wedge A \notin bad \wedge (\forall ag : note(ag, \langle A, pwd(A), G\rangle) \in tr \Rightarrow ag \notin bad)\\ &\quad \wedge enc(pwd(A), N_1) \in spies(tr))\\ &\Rightarrow gen(dh(G, N_1), M) \notin DY(spies(tr))\end{aligned}$$

2. $$\begin{aligned}&(tr \in \text{PACE} \wedge A \notin bad \wedge (\forall ag : note(ag, \langle A, pwd(A), G\rangle) \in tr \Rightarrow ag \notin bad)\\ &\quad \wedge obj^{dec}(dec(pwd(A), M_1), pwd(A), M_1))\\ &\Rightarrow gen(dh(G, dec(pwd(A), M_1)), M) \notin DY(spies(tr))\end{aligned}$$

According to our proof technique from Chap. 6, we prove these properties with the help of the canonical ccl-function ccl_1 and the correctness theorem 46. For that purpose, we have to provide in each case a set S that fulfills $\Re_1(S)$. And we employ the lemmas obtained from both confidentiality properties by replacing the conclusions with $spies(tr) \cap ccl_1(S) = \emptyset$.

In the proof of the first property, we make use of the set

$$S = \{gen(dh(G, N_1), M), dh(G, N_1), N_1, pwd(A)\}.$$

Note that $pwd(A)$ is required in S, in order to handle $enc(pwd(A), N_1)$ as not critical, i.e. $enc(pwd(A), N_1) \notin ccl_1(S)$.

The proof of the second property requires to make use of the set

$$\begin{aligned}S = \{&gen(dh(G, dec(pwd(A), M_1)), M), dh(G, dec(pwd(A), M_1)),\\ &dec(pwd(A), M_1), pwd(A)\}.\end{aligned}$$

It is easy to check in both cases that $\Re_1(S)$ holds, which allows us to handle the fake cases in the proofs of the mentioned lemmas simply by theorem 46. The rest of these proofs consists of ensuring that the protocol messages (and accidentally obtained session keys) do not belong to $ccl_1(S)$. This follows in the second lemma simply from the structure of the PACE messages, as no message matches any element of S. Contrarily, in the first lemma we have to deal with the case where a message $dh(G', N')$ (in steps 2 and 3) matches $dh(G, N_1)$ from S: The definition of PACE_2 and PACE_3 (in 74) requires that N' (matching N_1) does not occur in prior events. This contradicts the assumption that N_1 originates in a first PACE message (cp. $enc(pwd(A), N_1) \in spies(tr)$ in the premises of (78-1)).

8.3 Authentication by the Card

The mutual authentication by a two-party protocol consists of two authentication guarantees: authentication of the responder by the initiator and vice-versa. In this section, we discuss the authentication of the terminal (responder) by the card (initiator).

According to the general principles introduced in Sec. 7.1.7.2, an authentication guarantee for a participant ag relies on the authenticity of a message received and checked by ag. This requires that ag reaches a corresponding state in the protocol run where the authenticity verification can be carried out. In case of PACE, the card A checks the authenticity guarantees in step 6 by verifying the equality between the received message and the counterpart, which A computes as given in the receiver's view (see Sec. 2.3.1).

Property$^{\text{VSE}}$ 79 (Auth_byA):

$$
\begin{aligned}
&(tr \in \text{PACE} \wedge A \notin bad \wedge (\forall ag : note(ag, \langle A, pwd(A), G\rangle) \in tr \Rightarrow ag \notin bad)\\
&\quad \wedge (\forall N : note(spy, dh(dh(gen(dh(G,N_1),dh(M_2,N_3)),N_5),N)) \notin tr)\\
&\quad \wedge says(A,B,enc(pwd(A),N_1)) \in tr \wedge send(A,B,\langle N_1,N_3\rangle,dh(G,N_3)) \in tr\\
&\quad \wedge gets(A,M_4) \in tr \wedge send(A,B,N_5,dh(gen(dh(G,N_1),dh(M_2,N_3)),N_5)) \in tr\\
&\quad \wedge gets(A,mac(dh(M_4,N_5),dh(gen(dh(G,N_1),dh(M_2,N_3)),N_5))) \in tr)\\
&\Rightarrow\\
&(\exists B', N_2', MS, N_4' : note(B', \langle A, pwd(A), G\rangle) \in tr \wedge gets(B', enc(pwd(A), N_1)) \in tr\\
&\quad \wedge send(B', A, \langle enc(pwd(A), N_1), N_2'\rangle, dh(G, N_2')) \in tr\\
&\quad \wedge gets(B', map_{dh}(dh(G,N_3), MS)) \in tr \wedge M_2 = map_{dh}(dh(G, N_2'), MS)\\
&\quad \wedge send(B', A, N_4', dh(gen(dh(G,N_1), dh(map_{dh}(dh(G,N_3), MS), N_2')), N_4')) \in tr\\
&\quad \wedge M_4 = dh(gen(dh(G,N_1), dh(map_{dh}(dh(G,N_3), MS), N_2')), N_4')\\
&\quad \wedge gets(B', dh(gen(dh(G,N_1), dh(M_2,N_3)), N_5)) \in tr\\
&\quad \wedge says(B', A, mac(dh(dh(gen(dh(G,N_1), dh(M_2,N_3)), N_5), N_4'),\\
&\qquad\qquad dh(gen(dh(G,N_1), dh(M_2,N_3)), N_5))) \in tr)
\end{aligned}
$$

The events by A in tr fixes a PACE run from the point of view of A. In order to authenticate the communication partner of A in this run, the password of A and the MAC key may not be (accidentally) revealed. For this, A and every participant that has accessed the password of A may not be bad, and there may not exist an oops event for the MAC key.

The authentication guarantees for A are:

1. There is an agent B' that has run PACE in the *terminal role* with A.

   That is, the peer is indefinite (any device) from the point of view of A. But, her protocol role is definite.

2. B' has received the messages of A.

3. B' has generated the messages for A.

   That is, the peer addresses her messages to A.

Note that both messages in steps 2 and 3 can be modified by the attacker, by arbitrary many applications of "dh". This is expressed with the help of the map_{dh} function and an arbitrary message list MS: It is possible to modify the DH value $dh(G, N_3)$ that is sent by A in step 3. The peer would then receive $map_{dh}(dh(G, N_3), MS)$ (cp. the fourth event in the conclusion). In order for this modification to remain undetected, it is necessary to enforce the *same* modification for the DH value $dh(G, N_2')$ that is sent to A by the peer in step 2 (cp. the third event and the first equality in the conclusion).

When dh corresponds to exponentiation, MS contains arbitrary many exponents. Fortunately, the result is again an exponent of the initial generator G, and it is thus cryptographically as strong as G (see section 2.3.3).

Here, we have a typical situation where the guaranteed structure and content of the peer's messages M_2 and M_4 are security-relevant. This confirms the significance of the fourth feature among the features that we introduced in Sec. 7.1.7.2 to judge the strength of authenticity properties.

The authentication property 79 is proven immediately with the help of the corresponding authenticity lemma, which we obtain canonically as described in Sec. 7.1.7.2. Hence, the main effort is spent for the proof of the authenticity lemma, which is carried out by induction on PACE traces. The most involved part belongs to the fake case, where we prove that the attacker is not able to *forge* the authenticated message. It belongs to the typical proof states where we need to apply appropriate structuring lemmata. Before we provide (in Sec. 8.5) more insight in this central part of the authenticity proofs, we discuss the authenticity guarantees from the point of view of the terminal.

8.4 Authentication by the Terminal

The terminal B checks the authenticity guarantees in step 7 by verifying the equality between the received message and the counterpart, which B computes as given in the receiver's view (see Sec. 2.3.1).

Property$^{\mathbf{VSE}}$ 80 (Auth_byB):

$$
\begin{aligned}
&(tr \in \text{PACE} \wedge A \notin bad \wedge (\forall ag : note(ag, \langle A, pwd(A), G\rangle) \in tr \Rightarrow ag \notin bad)\\
&\quad \wedge (\forall N : note(spy, dh(dh(gen(dh(G, dec(pwd(A), M_1)), dh(M_3, N_2)), N_4), N)) \notin tr)\\
&\quad \wedge (\exists tr_0, tr_1 : tr = tr_1 \# tr_0 \wedge note(B, \langle A, pwd(A), G\rangle) \in tr_0\\
&\qquad \wedge send(B, A, \langle M_1, N_2\rangle, dh(G, N_2)) \in tr_1\\
&\qquad \wedge send(B, A, N_4, dh(gen(dh(G, dec(pwd(A), M_1)), dh(M_3, N_2)), N_4)) \in tr_1\\
&\qquad \wedge gets(B, M_5) \in tr_1\\
&\qquad \wedge gets(B, mac(dh(M_5, N_4), dh(gen(dh(G, dec(pwd(A), M_1)), dh(M_3, N_2)), N_4)))\\
&\qquad\quad \in tr_1)))\\
\Rightarrow\\
&(\exists ag_1, MS, ag_2, \overline{M}_1, \overline{N}, \widehat{N} : ag_1 \neq B \wedge \langle pwd(A), G\rangle \in DY(got\_info(ag_1, tr))\\
&\quad \wedge gets(ag_1, map_{dh}(dh(G, N_2), MS)) \in tr \wedge send(ag_1, ag_2, \langle \overline{M}_1, \overline{N}\rangle, dh(G, \overline{N})) \in tr\\
&\quad \wedge M_3 = map_{dh}(dh(G, \overline{N}), MS)\\
&\quad \wedge \overline{M}_1 \in DY(\{pwd(A), M_1\}) \wedge M_1 \in DY(\{pwd(A), \overline{M}_1\})\\
&\quad \wedge gets(ag_1, dh(gen(dh(G, dec(pwd(A), M_1)), dh(M_3, N_2)), N_4)) \in tr\\
&\quad \wedge send(ag_1, ag_2, \widehat{N}, M_5) \in tr\\
&\quad \wedge M_5 = dh(gen(dh(G, dec(pwd(A), M_1)), dh(map_{dh}(dh(G, N_2), MS), \overline{N})), \widehat{N})\\
&\quad \wedge says(ag_1, ag_2, mac(dh(dh(gen(dh(G, dec(pwd(A), M_1)), dh(M_3, N_2)), N_4), \widehat{N}),\\
&\qquad\qquad dh(gen(dh(G, dec(pwd(A), M_1)), dh(M_3, N_2)), N_4))) \in tr)
\end{aligned}
$$

The events by B in tr fixes a PACE run from the point of view of B. To express that the access to the password $pwd(A)$ and the generator G happens prior to the other events in tr, we subdivide tr into two chronologically ordered parts tr_0 and tr_1. The *note* event that represents the access to the password and the generator belongs to tr_0, and the remaining

events by B occur in tr_1.

Similar to the authentication property in 79, we need that the password and the MAC key are not (accidentally) revealed.

As it is shown in Fig. 8.1, there is no guarantee for a terminal B that the peer is running PACE in the card role. In this figure we have a complete PACE run between a terminal B and a second agent B' acting also in the terminal role. The used M_1 is an arbitrary message, which is not necessary an encrypted nonce. Fortunately, this does not pose a real security threat in the application scenario, since the card will not reach a state where access to anyone of the terminals is granted (see Sec. 2.2).

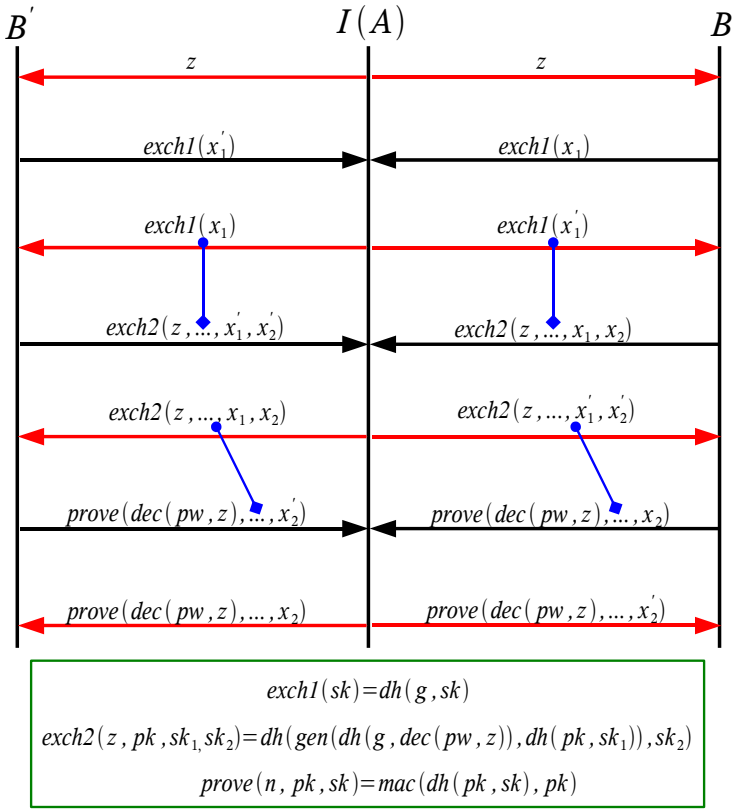

Figure 8.1: False Authentication of a Card

Hence, the authentication guarantees for B include both cases, i.e. the peer has run PACE in the card role or in the terminal role. This means the following:

1. There is an agent ag_1 that has received the messages of B. She is not necessarily the participant A addressed by B.
2. ag_1 has generated the messages that were received by B. But, the messages of ag_1 are not necessarily addressed to B.
3. ag_1 and B have used the same password.

These guarantees are given in both cases: (i) ag_1 has run PACE in the card role and (ii) ag_1 has run PACE in the terminal role. In the former case, the intended partner of ag_1 is

any terminal ag_2. The message $\overline{M}_1$ that is recorded in the *send* event by ag_1 is a nonce, and M_1 must be equal to the encryption of this nonce by $pwd(A)$.

In case (ii), the intended partner of ag_1 is A. The message $\overline{M}_1$ that is recorded in the *send* event by ag_1 equals M_1.

In both cases, we have a similar guarantee about the structure and the content of the peer's messages M_3 and M_5 as in property 79.

8.5 Protection of the MAC Key

Both authenticity properties rely on the same central argument, that the peer is not able to generate the MAC without possessing the *fresh* generator used in the second DH exchange (steps 4 and 5). According to PACE, a card A computes the fresh generator $\widehat{g}_A$ by $\widehat{g}_A = gen(dh(g,nc_1),dh(M_2,nc_3))$, where

- nc_1 and nc_3 are arbitrary nonces, generated by A in steps 1 and 3, respectively,
- g is the static generator of A,
- and M_2 is a arbitrary message received by A in step 2.

Note that nc_1, nc_3, and g can be seen as arbitrary instances of the variables N_1, N_3 and G in 79.

In step 6, A checks the authenticity of the received message M_6 by comparison with the own *mac*-message $mac(dh(M_4,nc_5),dh(\widehat{g}_A,nc_5))$, where

- M_4 is a arbitrary message received by A in step 4,
- and nc_5 is a arbitrary nonce, generated by A and sent as part of $\widehat{m}_5 = dh(\widehat{g}_A,nc_5)$ in step 5.

Here, nc_5 can be seen as an arbitrary instance of the variable N_5 in 79.

Since nc_5 is a local secret of A and was sent *bound with* the fresh generator $\widehat{g}_A$ in $\widehat{m}_5$, the only way to *embed* nc_5 in the MAC-key is to use $\widehat{m}_5$ in its computation. This implies that $dh(M_4,nc_5) = dh(\ldots,dh(\widehat{m}_5,x_1),\ldots,x_n)$, i.e. $M_4 = dh(\ldots,dh(\widehat{g}_A,x_1),\ldots,x_n)$, and that the peer has used $x_1,\ldots,x_n$ to compute the MAC-key. According to the *regular* structure of the *dh*-messages in the protocol, anyone who generates M_4 and possesses at the same time *all* $x_1,\ldots,x_n$, would need to use $\widehat{g}_A$. This entails that the peer was able to compute the fresh generator $\widehat{g}_A$. As $\widehat{g}_A$ is unknown by the attacker (see Sec. 8.2), the peer is honest and has participated in the computation of $\widehat{g}_A$ using the password $pwd(A)$.

The same argumentation applies to the authenticity of message 7, as required for 80. The considered message parts are obviously from the point of view of a terminal B. For instance, the fresh generator is computed by $\widehat{g}_B = gen(dh(g,dec(pwd(A),M_1),dh(M_3,nc_2))$, where

- g is the static generator of A and $pwd(A)$ her password,
- M_1 and M_3 are arbitrary messages received by B in steps 1 and 3, respectively,
- and nc_2 is a arbitrary nonce, generated by B in step 2.

Here, nc_2 and g can be seen as arbitrary instances of the variables N_2 and G in 80.

The above mentioned binding and regularity arguments are in particular required when handling the fake cases in the authenticity proofs. Considering the authenticity of message 6, the fake case yields to a proof state where

1. the *mac*-message $\widehat{m}_6 = mac(dh(M_4,nc_5),dh(\widehat{g}_A,nc_5))$ does not belong to the observable messages ($\widehat{m}_6 \notin ik$),

2. and $\widehat{m}_6$ is *assumed* to be forged by the attacker ($\widehat{m}_6 \in DY(ik)$).

The proof goal is handled then by tracing back the assumption (in 2) to the negation of the basic confidentiality property. This is done in two major steps where we need to apply our proof technique from Chap. 6.

Step 1: We reduce the occurrence of the *mac*-message to the occurrence of the MAC-key in the intruder knowledge. Following our proof technique, we use ccl_2 and apply theorem 54 in combination with the following *regularity lemma*. It holds, when the protocol messages do not include the fixed *mac*-object (neither as a crypt-part nor as a select-part).

Property$^{\mathbf{VSE}}$ 81 (Absence of protected mac-Objects):

$$\begin{aligned}&(tr \in \text{PACE} \wedge mac(m_0, m_1) \notin spies(tr) \wedge \neg(m_0, m_1 \in DY(spies(tr))))\\ &\Rightarrow spies(tr) \cap ccl_2(mac(m_0, m_1)) = \emptyset\end{aligned}$$

It is easy to check that definition 51 (of ccl_2), the corresponding correctness theorem 54 and the protocol property 81 allow us to conclude from the assumption in (2) and the hypothesis in (1) that the MAC-key $dh(M_4, nc_5)$ belongs to $DY(ik)$.

Step 2: We reduce the occurrence of the MAC-key $dh(M_4, nc_5)$ to the occurrence of the fresh generator $\widehat{g}_A$ in the intruder knowledge. This is done by combining the following propositions on M_4, nc_5 and $\widehat{g}_A$:

1. As M_4 is the arbitrary message that is received by A in step 4, it belongs to the intruder knowledge.
2. The nonce nc_5 is sent bound with the fresh generator $\widehat{g}_A$ in (the message $dh(\widehat{g}_A, nc_5)$ of) step 5. It belongs to the local secrets of A.

The key property in this part of the proof is a theorem that enforces a certain structure of M_4 due to the binding of nc_5 with $\widehat{g}_A$. In fact, the theorem holds not only for the local secret nc_5 and the generator $\widehat{g}_A$ used in step 5, but also for the local secrets and the generators used in the other DH steps (2, 3 and 4) of PACE. According to the protocol, the used generators ($\widehat{m}$) by regular participants (in steps 2–5) are not *dh*-objects and the generated nonces ($\widehat{N}$) are *local secrets*, i.e. confidential. For a public DH value $dh(\widehat{m}, \widehat{N})$ that is sent by a regular participant, the following theorem provides us with all possible alternatives for the attacker to embed $\widehat{N}$ with an *arbitrary message* m in a *dh*-object $dh(m, \widehat{N})$. In the proof of this theorem, we need to reason on the derivation of $dh(m, \widehat{N})$ by induction on the *dh*-structure of m. For this reason, we generalize the occurrence of $dh(m, \widehat{N})$ to the occurrence of some message in $ccl_2(dh(m, \widehat{N}))$. More details are given while sketching the proof of the obtained binding property:

Property$^{\mathbf{VSE}}$ 82 (Binding local Secret with Generator):

$$\begin{aligned}&(tr \in \text{PACE} \wedge dh(\widehat{m}, \widehat{N}) \in spies(tr) \wedge (\forall m_1, m_2 : \neg obj^{dh}(\widehat{m}, m_1, m_2))\\ &\quad \wedge \widehat{N} \notin DY(spies(tr)) \wedge DY(spies(tr)) \cap ccl_2(dh(m, \widehat{N})) \neq \emptyset)\\ &\Rightarrow\\ &((\exists MS : m = map_{dh}(\widehat{m}, MS) \wedge MS \subset DY(spies(tr))) \vee\\ &\ (\exists N, MS : note(spy, dh(dh(\widehat{m}, \widehat{N}), N)) \in tr\\ &\qquad \wedge dh(m, \widehat{N}) = map_{dh}(dh(dh(\widehat{m}, \widehat{N}), N), MS)))\end{aligned}$$

This property states that there are three alternatives for the attacker to obtain a *dh*-object $dh(m,\widehat{N})$:

1. reusing the public DH value $dh(\widehat{m},\widehat{N})$, i.e. $m = \widehat{m}$,

2. computing a common DH value out of the public DH value $dh(\widehat{m},\widehat{N})$ and arbitrary many private values in a multiset $MS = \{x_1,\ldots,x_n\}$ that the attacker employed to generate a public DH value $dh(\ldots,dh(\widehat{m},x_1),\ldots,x_n)$,

3. or computing m out of an accidentally obtained common DH value $dh(dh(\widehat{m},\widehat{N}),N)$ and arbitrary many messages by successive applications of *dh*.

Note that the first alternative corresponds to the first part of the conclusion (in property 82) when MS is empty.

Before we give an overview on the proof of theorem 82, we briefly explain its use in the fake case of our authenticity proofs: Let us focus on the authenticity by A. Here, we apply theorem 82 substituting $\widehat{m}$, $\widehat{N}$ and respectively m by $\widehat{g}_A$, nc_5 and M_4, respectively. The above three cases yields the following proof states:

1. $m = \widehat{m}$ implies $M_4 = \widehat{g}_A$. The premise (in 1), i.e. M_4 belongs to the observable messages, means then that $\widehat{g}_A$ belongs to the intruder knowledge. This allows us to close the goal by contradiction with the confidentiality of $\widehat{g}_A$.

2. The generation of a public DH value $dh(\ldots,dh(\widehat{m},x_1),\ldots,x_n)$ using $x_1,\ldots,x_n$ by the attacker necessitates to use also the generator $\widehat{m}$, which we substituted with $\widehat{g}_A$. This allows us to proceed like in the first case.

3. The accidentally obtained common DH value implies that the trace tr includes the event $note(spy,dh(dh(\widehat{g}_A,nc_5),N))$. This allows us to close the goal by contradiction with the assumption that the MAC-key from the point of view of A is not revealed accidentally (cp. the fourth premise in 79).

The proof arguments in case (1) and (2) are actually based on a regularity lemma about the occurrence of *dh*-values $map_{dh}(\widehat{g},\{x_1,\ldots,x_n\})$ together with the elements of $\{x_1,\ldots,x_n\}$ in the intruder knowledge. The regularity lemma implies that the attacker needs (in such a case) to possess the generator $\widehat{g}$, too.

Now, we turn to the proof of theorem 82. This theorem is a typical protocol property that is proven by nested induction: The first induction is wrt. the *dh*-structure of m that allows us to trace back the occurrence of $dh(m,\widehat{N})$ (for all possible messages m) to the occurrence of smaller *dh*-objects $dh(m',\widehat{N})$.

In the base case, we prove the theorem conjecture for all messages m that are not *dh*-objects. More details are given below.

In the step case, we consider messages $dh(m,\widehat{N})$ with at least *two successive applications* of *dh* at the top-positions. Such a *dh*-message does not occur in any protocol step. It matches only the messages that can be gained by the attacker in oops events. These facts are proven by induction on PACE traces. In the obtained step case, the conjecture hypothesis includes one proposition of the form

$$DY(\{m_{PACE}\} \cup spies(tr)) \cap ccl_2(dh(m,\widehat{N})) \neq \emptyset,$$

where m_{PACE} originates from the last event in the extended trace.

Applying the correctness theorem 55 of ccl_2 wrt. *dh*-objects to this part of the conjecture hypothesis, we obtain three cases:

1. The observable messages allow us to derive $dh(m, \widehat{N})$ by *extraction*, i.e. they include $dh(m, \widehat{N})$ in clear-text, as select-part or as crypt-part, as formalized by

$$\{m_{PACE}\} \cup spies(tr) \cap ccl_2(dh(m, \widehat{N})) \neq \emptyset.$$

2. The *dh*-message $dh(m, \widehat{N})$ is generated by composition out of m and $\widehat{N}$, i.e.

$$m, \widehat{N} \in DY(\{m_{PACE}\} \cup spies(tr)).$$

3. The *dh*-message $dh(m, \widehat{N})$ is generated by composition out of $dh(m', \widehat{N})$ and some message m_i, i.e.

$$dh(m', \widehat{N}), m_i \in DY(\{m_{PACE}\} \cup spies(tr)) \text{ with } dh(dh(m', \widehat{N}), m_i) = dh(m, \widehat{N}).$$

The first proof state is handled with a further case distinction:

- The observable messages in $spies(tr)$ allow us to derive $dh(m, \widehat{N})$ by extraction, i.e. $spies(tr) \cap ccl_2(dh(m, \widehat{N})) \neq \emptyset$. This proof goal is closed with the help of the corresponding induction hypothesis.
- The message m_{PACE}, which originates from the last event in the extended trace, allows us to derive $dh(m, \widehat{N})$ by extraction, i.e. $m_{PACE} \in ccl_2(dh(m, \widehat{N}))$. Since $dh(m, \widehat{N})$ has at least two successive applications of dh at the top-positions, only a message $m_{PACE} = dh(dh(\widehat{G}, N), N')$ that originates from a oops event fulfills this property. This provides us with the second alternative in the conclusion of theorem 82. Here, we have the specific case where the list MS is empty.

The second proof state above is closed by contradiction with the assumption that $\widehat{N}$ is confidential, i.e. $\widehat{N} \notin DY(\{m_{PACE}\} \cup spies(tr))$.

In the third proof state, the occurrence of a smaller *dh*-object $dh(m', \widehat{N})$ in the knowledge set $DY(\{m_{PACE}\} \cup spies(tr))$ allows us to apply the original induction hypothesis (wrt. the *dh*-structure). The lists MS in the resulting consequences are then extended with m_i.

We terminate this section with an overview on the base case in the proof of our theorem 82. Since m is not a *dh*-object, we know that $dh(m, \widehat{N})$ can be generated only out of m and $\widehat{N}$. As $\widehat{N}$ is confidential, the only remaining option for the attacker to derive a message of the form $dh(m, \widehat{N})$ is to reuse $dh(\widehat{m}, \widehat{N})$. This means, $m = \widehat{m}$, which we prove in the following unicity theorem by induction on PACE traces.

Property$^{\text{VSE}}$ 83 (Unicity of DH Generator):

$$\begin{aligned}
&(tr \in \text{PACE} \wedge \widehat{N} \notin DY(spies(tr))) \\
&\Rightarrow \\
&(\exists \widehat{m} : (\forall m : ((\forall m_1, m_2 : \neg obj^{dh}(m, m_1, m_2)) \wedge spies(tr) \cap ccl_2(dh(m, \widehat{N})) \neq \emptyset) \\
&\qquad \Rightarrow m = \widehat{m}))
\end{aligned}$$

This unicity property of PACE, which is fundamental for the authenticity guarantees, means intuitively the following: Any occurrence of a *secret nonce* $\widehat{N}$ in $dh(m, \widehat{N})$ with a message m that is not a *dh*-object corresponds to the occurrence of a public DH value $dh(\widehat{m}, \widehat{N})$, where $\widehat{m}$ is used as generator.

8.6 Forward Secrecy of Session Keys

According to the technical document [60] that specifies PACE, the MAC-key is used as key material to generate the session key used afterwards for the secure transmission of the application data. Hence, the confidentiality of the session key follows immediately from the confidentiality of the MAC-key, which belongs in turn to the central arguments of the authenticity proofs. In this section, we are interested in a stronger confidentiality property, i.e. the forward secrecy of this key, which implies as well that of the session key.

According to the authentication properties, the MAC-key corresponds to the common DH value established in the second DH phase, where the public DH values are generated by two honest participants. Furthermore, it is necessary to assume additionally that the key is not disclosed accidentally. In this way, we obtain the following formalization for the intended forward secrecy property:

Property$^{\mathbf{VSE}}$ 84 (Forward Secrecy):

$$\begin{aligned}
&(tr \in \text{PACE} \wedge Ag_1 \notin bad \wedge Ag_2 \notin bad \wedge \\
&\quad \wedge send(Ag_1, Ag, \widehat{N}_1, dh(gen(M, \widehat{M}), \widehat{N}_1)) \in tr \\
&\quad \wedge send(Ag_2, Ag', \widehat{N}_2, dh(gen(M, \widehat{M}), \widehat{N}_2)) \in tr \\
&\quad \wedge note(spy, dh(dh(gen(M, \widehat{M}), \widehat{N}_1), \widehat{N}_2)) \notin tr) \\
&\Rightarrow dh(dh(gen(M, \widehat{M}), \widehat{N}_1), \widehat{N}_2) \notin DY(\{pwd(A)\} \cup spies(tr))
\end{aligned}$$

The premises cover all the possible scenarios given in the authentication properties.

The exclusion of the *note* event implies that we are dealing with a secret message that does not occur in any protocol message. According to our proof technique from Chap. 6, we may then prove the property 84 with the help of the *ccl*-function ccl_2 and its correctness theorem 55: We assume $dh(dh(gen(M, \widehat{M}), \widehat{N}_1), \widehat{N}_2) \in DY(\{pwd(A)\} \cup spies(tr))$ and apply the correctness theorem 55, which yields two cases:

1. $(\{pwd(A)\} \cup spies(tr)) \cap ccl_2(dh(dh(gen(M, \widehat{M}), \widehat{N}_1), \widehat{N}_2)) \neq \emptyset$
2. $\widehat{N}_2, dh(gen(M, \widehat{M}), \widehat{N}_1)$ are in $DY(\{pwd(A)\} \cup spies(tr))$ or $\widehat{N}_1, dh(gen(M, \widehat{M}), \widehat{N}_2)$ are in $DY(\{pwd(A)\} \cup spies(tr))$.

In the first case, we apply the definition of ccl_2 and a corresponding lemma, where we simply check that no message in $spies(tr)$ allows us to derive $dh(dh(gen(M, \widehat{M}), \widehat{N}_1), \widehat{N}_2)$ by extraction. The second case is refuted by the fact that the nonces $\widehat{N}_2$ and $\widehat{N}_1$ do not belong to $DY(\{pwd(A)\} \cup spies(tr))$, since they are generated locally by Ag_1 and Ag_2 and cannot be extracted from the observable messages.

Chapter 9

Verification of TC-AMP's Trace Properties

In this chapter we give an overview on the verification of the trace properties of TC-AMP. We start with an excerpt from the specification of the protocol model. Then we describe the proofs of the main protocol properties: the basic confidentiality property, the mutual authentication, and the forward secrecy of session key. We focus in particular on the formalization and we sketch the proof ideas as well as the use of our proof technique, described in Chap. 6.

9.1 Protocol Model

Following the conventions in Sec. 7.1.6, the set TCAMP, which contains all possible TC-AMP traces, is defined according to schema 70. Besides the generic cases, we obtain four protocol-specific cases: $\text{TCAMP}_1(ev,tr)$–$\text{TCAMP}_3(ev,tr)$ cover the admissible trace extensions according to TC-AMP and $Oops_{ev}(ev,tr)$ models dynamic corruption. In this section, we discuss the definition of these predicates. Before that, we describe the protocol-specific details regarding the initial knowledge.

9.1.1 Initial Knowledge

In Sec. 7.1.4 we introduced the predicate $initHas(ag,m)$ to express what participants initially know. The protocol-specific assumptions about this initial knowledge are specified in form of axioms.

Axiom[VSE] 85 (initHas; TC-AMP):

1.
$$initHas(ag,m) \Rightarrow (\exists g_1,g_2 : g_1 \neq g_2 \wedge (m = \langle pwd(ag),g_1,g_2\rangle \vee (\exists ag' : ag \neq ag' \wedge m = \langle ag',pwd(ag'),g_1,g_2\rangle)))$$

2.
$$(initHas(ag,\langle pwd(ag),g_1,g_2\rangle) \wedge initHas(ag',\langle ag,pwd(ag),g_1',g_2'\rangle)) \Rightarrow (g_1 = g_1' \wedge g_2 = g_2')$$

3.

$$\exists g_1, g_2 : initHas(ag, \langle pwd(ag), g_1, g_2 \rangle)$$

4.

$$\exists ag, ag', g_1, g_2 : ag \neq ag' \wedge initHas(ag, \langle ag', pwd(ag'), g_1, g_2 \rangle)$$

The initial knowledge consists of triples of the form $\langle pwd(ag), g_1, g_2 \rangle$ and quadruplets of the form $\langle ag', pwd(ag'), g_1, g_2 \rangle$, where g_1 and g_2 are two different atomic bases. To run TC-AMP in the card role, ag needs to access $\langle pwd(ag), g_1, g_2 \rangle$, i.e. the own password and two associated generators.

In order to run TC-AMP in the terminal role, ag has to access $\langle ag', pwd(ag'), g_1, g_2 \rangle$. Note that the participant ag' (in the card role) *differs* from ag. This way, we have a kind of role separation, in the sense that no agent runs TC-AMP with himself as a partner (see Sec. 9.3 and 9.4).

By the second axiom, every password is associated with the same pair of generators, and that from the perspective of all agents. Every agent can run TC-AMP in the card role and there is at least one agent that is able to run TC-AMP in the terminal role. These axioms are needed to prove that the specification of the TC-AMP rules is not restrictive. That is, complete TC-AMP runs are not excluded through the conditions used in our formalization of these rules.

9.1.2 TC-AMP Rules

In this section we discuss the specification of the TC-AMP rules. The first rule includes the condition for initiating a new run by a terminal.

Definition[VSE] 86 (TCAMP_1):

$$\begin{aligned}
&\text{TCAMP}_1(ev, tr) \Leftrightarrow \\
&(\exists B, A, g_1, g_2, nc_1 : B \neq A \wedge note(B, \langle A, pwd(A), g_1, g_2 \rangle) \in tr \\
&\quad \wedge \neg used(nc_1, tr) \wedge ev = send(B, A, nc_1, \oplus(*(nc_1, g_1), \ominus(*(pwd(A), g_2)))))
\end{aligned}$$

Before initiating a run, B must have accessed the password and the associated pair of generators that belong to a card A.

Definition[VSE] 87 (TCAMP_2):

$$\begin{aligned}
&\text{TCAMP}_2(ev, tr) \Leftrightarrow \\
&(\exists A, B, g_1, g_2, M_1, nc_2 : A \neq B \wedge note(A, \langle pwd(A), g_1, g_2 \rangle) \in tr \\
&\quad \wedge gets(A, M_1) \in tr \wedge \neg used(nc_2, tr) \wedge \\
&\quad ev = send(A, B, nc_2, \langle *(pwd(A), *(nc_2, g_1)), \\
&\qquad h_1(M_1, *(pwd(A), *(nc_2, g_1)), *(nc_2, \oplus(M_1, \oplus(*(pwd(A), g_2), *(M_1, g_1)))))))\rangle))
\end{aligned}$$

Before sending message 2, a card A must have accessed the own password and the associated pair of generators. It also must have received a message M_1 that is expected to be generated by a terminal B using the same password and generators and a new nonce.

Definition[VSE] 88 (TCAMP$_3$):

$$
\begin{aligned}
&\mathrm{TCAMP}_3(ev,tr) \Leftrightarrow \\
&(\exists B,A,g_1,g_2,nc_1,M_2 : send(B,A,nc_1,\oplus(*(nc_1,g_1),*(pwd(A),\ominus(g_2)))) \in tr \wedge \\
&\quad gets(B,\langle M_2,h_1(\oplus(*(nc_1,g_1),*(pwd(A),\ominus(g_2))),M_2, \\
&\qquad \oplus(*(inv(pwd(A)),*(nc_1,M_2)), \\
&\qquad\quad *(inv(pwd(A)),*(\oplus(*(nc_1,g_1),*(pwd(A),\ominus(g_2))),M_2))))\rangle) \in tr \wedge \\
&\quad ev = says(B,A,h_2(\oplus(*(nc_1,g_1),*(pwd(A),\ominus(g_2))),M_2, \\
&\qquad \oplus(*(inv(pwd(A)),*(nc_1,M_2)), \\
&\qquad\quad *(inv(pwd(A)),*(\oplus(*(nc_1,g_1),*(pwd(A),\ominus(g_2))),M_2))))))
\end{aligned}
$$

The terminal B gets the response of A in step 2, verifies its content, and sends in step 3 another hash value, which A uses to verify the authentication of B.

In addition to the regular protocol steps, we model the accidental loss of session keys for a stronger form of resistance against offline password testing (see Chap. 14).

Definition[VSE] 89 (Oops$_{ev}$; TC-AMP):

$$
\begin{aligned}
&Oops_{ev}(ev,tr) \Leftrightarrow \\
&(\exists B,A,g_1,g_2,nc_1,nc_2 : send(B,A,nc_1,\oplus(*(nc_1,g_1),*(pwd(A),\ominus(g_2)))) \in tr \\
&\quad \wedge send(A,B,nc_2,\langle *(pwd(A),*(nc_2,g_1)), \\
&\qquad h_1(\oplus(*(nc_1,g_1),*(pwd(A),\ominus(g_2))),*(pwd(A),*(nc_2,g_1)), \\
&\qquad\quad \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,\oplus(*(nc_1,g_1),*(pwd(A),\ominus(g_2))),g_1)))\rangle) \in tr \\
&\quad \wedge ev = note(spy, \\
&\qquad\quad \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(\oplus(*(nc_1,g_1),*(pwd(A),\ominus(g_2))),g_1)))))
\end{aligned}
$$

The first step by B and the second step by A fix the accidentally lost session key (the third h_1- and h_2-part) from the perspective of the participants.

9.2 Basic Confidentiality Properties

The main objective of TC-AMP is the *authenticated* establishment of a session key between a terminal B and a card A.

This requires in particular the mutual authentication of both participants, which comprises typically one authentication guarantee from the point of view of B (see Sec. 9.3) and another authentication guarantee from the point of view of A (see Sec. 9.4).

The proof of the first authentication guarantee is based on the protection of the third h_1-part in the second message part of step 2, which we describe in Sec. 9.5. This protection relies on the confidentiality of the message parts nc_1, π_A, $*(nc_1,g_1)$ and $*(\pi_A,g_2)$ that occur in the first message by B.

In parallel, the proof of the second authentication guarantee is based on the protection of the third h_2-part in step 3, which we describe in Sec. 9.6. This protection relies on the confidentiality of the message parts nc_2, π_A, $*(nc_2,g_1)$ and $*(\pi_A,g_1)$ that occur in the first message by A, i.e. in the second TC-AMP step.

In addition to the mutual authentication, TC-AMP guarantees the forward secrecy of the established session key (see Sec. 9.7). This property relies primarily on the local secrecy of the nonces nc_1 and nc_2 (generated by honest B and A), which are not accessible for the attacker even when the password π_A is disclosed.

Recapitulating, the proof of the TC-AMP standard properties requires the following basic confidentiality properties:

Property$^{\text{VSE}}$ 90 (Basic Confidentiality):

1. $(tr \in \text{TCAMP} \wedge A \notin bad \wedge (\forall ag : note(ag, \langle A, pwd(A), g_1, g_2 \rangle) \in tr \Rightarrow ag \notin bad)$
   $\wedge\, send(B, A, nc_1, (nc_1 * g_1) \oplus (\ominus(pwd(A) * g_2))) \in tr)$
   $\Rightarrow pwd(A), nc_1, *(nc_1, g_1), *(pwd(A), g_2) \notin DY(spies(tr))$

2. $(tr \in \text{TCAMP} \wedge A \notin bad \wedge (\forall ag : note(ag, \langle A, pwd(A), g_1, g_2 \rangle) \in tr \Rightarrow ag \notin bad)$
   $\wedge\, send(A, B, nc_2, \langle nc_2 * (pwd(A) * g_1), h_1(m_1, nc_2 * (pwd(A) * g_1),$
   $nc_2 * ((pwd(A) * g_2) \oplus m_1 \oplus (m_1 * g_1)))\rangle) \in tr)$
   $\Rightarrow pwd(A), nc_2, *(nc_2, g_1), *(pwd(A), g_1) \notin DY(spies(tr))$

3. $(tr \in \text{TCAMP} \wedge B \notin bad \wedge$
   $send(B, A, nc_1, (nc_1 * g_1) \oplus (\ominus(pwd(A) * g_2))) \in tr)$
   $\Rightarrow nc_1 \notin DY(spies(tr) \cup \{pwd(A)\})$

4. $(tr \in \text{TCAMP} \wedge A \notin bad \wedge$
   $send(A, B, nc_2, \langle nc_2 * (pwd(A) * g_1), h_1(m_1, nc_2 * (pwd(A) * g_1),$
   $nc_2 * ((pwd(A) * g_2) \oplus m_1 \oplus (m_1 * g_1)))\rangle) \in tr)$
   $\Rightarrow nc_2 \notin DY(spies(tr) \cup \{pwd(A)\})$

According to our proof technique from Chap. 6, we prove properties 1 and 2 with the help of the *ccl*-function ccl_1 and the correctness theorem 46 and properties 3 and 4 with the help of the *ccl*-function ccl_2 and the correctness theorem 53. We want to focus on the proof of properties 1 and 2, since the proofs of 3 and 4 are straightforward.

For the use of the *ccl*-function ccl_1 and the corresponding correctness theorem, we have to provide a set S that fulfills $\Re_1(S)$. Instead of using two tailored sets, one for each property, we prefer to use a single set S for both properties. This set is fixed through the non-compromised password $pwd(A)$. It contains (in addition to $pwd(A)$) all nonces and all confidential $\oplus$-parts that are bound with $pwd(A)$, including $\oplus$-parts in corresponding oops-events. So, we use $\Phi^{\text{TCAMP}}(S, pwd(A), tr)$ to express that S is the smallest set that fulfills:

1. if there is some event $note(ag, \langle A, pwd(A), g_1, g_2 \rangle)$ or $note(A, \langle pwd(A), g_1, g_2 \rangle)$ in tr where the password $pwd(A)$ is not accessed by a bad agent, then the set S includes $pwd(A), *(pwd(A), g_1)$ and $*(pwd(A), g_2)$,
2. if there is some event $send(B, A, nc_1, (nc_1 * g_1) \oplus (\ominus(pwd(A) * g_2)))$ in tr where the password $pwd(A)$ is not accessed by a bad agent, then the set S includes nc_1 and $*(nc_1, g_1)$, and
3. if there is some event $send(A, B, nc_2, \langle nc_2 * (pwd(A) * g_1), h_1(m_1, nc_2 * (pwd(A) * g_1), nc_2 * ((pwd(A) * g_2) \oplus m_1 \oplus (m_1 * g_1)))\rangle)$ in tr where the password $pwd(A)$ is not accessed by a bad agent, then the set S includes nc_2, $*(nc_2, g_1)$ and $*(nc_2, *(nc_1, g_1))$ for each $send(B, A, nc_1, (nc_1 * g_1) \oplus (\ominus(pwd(A) * g_2)))$ in tr.

Note that $\Phi^{\text{TCAMP}}(S, pwd(A), tr)$ implies the required condition $\Re_1(S)$ for the application of the correctness theorem.

Using $\Phi^{\text{TCAMP}}(S, pwd(A), tr)$, we formalize the lemma that combines the basic confidentiality properties 1 and 2 as follows:

Lemma[VSE] 91 (Confidentiality of $*$- and $\oplus$-Parts):

$$(tr \in \text{TCAMP} \wedge A \notin bad \wedge (\forall ag : note(ag, \langle A, pwd(A), g_1, g_2\rangle) \in tr \Rightarrow ag \notin bad)$$
$$\Rightarrow (\exists S : \Phi^{\text{TCAMP}}(S, pwd(A), tr) \wedge spies(tr) \cap ccl_1(S) = \emptyset)$$

The proof of this lemma is by induction on TC-AMP traces. The induction hypothesis provides us with a set S_{tr} that fulfills

C1: $\Phi^{\text{TCAMP}}(S_{tr}, pwd(A), tr)$, and

C2: $spies(tr) \cap ccl_1(S_{tr}) = \emptyset$.

The proof goal in the step case consists in providing an appropriate extension $S_{ev,tr}$ of the set S_{tr} such that we have

G1: $\Phi^{\text{TCAMP}}(S_{ev,tr} \uplus S_{tr}, pwd(A), ev_\bullet tr)$, and

G2: $spies(ev_\bullet tr) \cap ccl_1(S_{ev,tr} \uplus S_{tr}) = \emptyset$.

In the fake case, where m_{ev} the message of the last event ev belongs to $DY(spies(tr))$, the set $S_{ev,tr}$ is empty and proof goal G2 follows from C2 (in the induction hypothesis) by applying the correctness theorem 46 of ccl_1.

In the remaining cases, we need the sets $S_{ev,tr}$ and S_{tr} to satisfy (as explained in Sec. 6.1.2) the required extension condition in theorem 45, for some message part $\widehat{m}$ that is not used in the observable messages from tr (but in the last event ev). This means, $S_{ev,tr}$ and S_{tr} must satisfy

$$(\forall m \in S_{ev,tr} : uses(m, \widehat{m})) \wedge (\forall m \in S_{tr} : \neg uses(m, \widehat{m})),$$

in order to use C2 from the induction hypothesis and reduce the assumed negation of G2 to $m_{ev} \in ccl_2(S_{ev,tr} \uplus S_{tr})$, provided $\widehat{m}$ is not used in the observable messages from tr. Then, we refute this assumption by showing that $m_{ev} \notin ccl_1(S_{ev,tr} \uplus S_{tr})$ holds, (based on $\Phi^{\text{TCAMP}}(S_{tr}, pwd(A), tr)$).

To reach our proof goal, we set the set $S_{ev,tr}$ according to the following principle:

1. In case the event ev equals $note(ag, \langle A, pwd(A), g_1, g_2\rangle)$ or $note(A, \langle pwd(A), g_1, g_2\rangle)$ and $pwd(A)$ is non-compromised, we distinguish whether $pwd(A)$ is already included in S_{tr}.
   - If $pwd(A) \in S_{tr}$, we set $S_{ev,tr} = \emptyset$.
   - Otherwise, we set $S_{ev,tr} = \{pwd(A), *(pwd(A), g_1), *(pwd(A), g_2)\}$.
2. In case $ev = send(B, A, nc_1, (nc_1 * g_1) \oplus (\ominus(pwd(A) * g_2)))$ and $pwd(A)$ is non-compromised, we set $S_{ev,tr} = \{nc_1, *(nc_1, g_1)\}$.
3. When the last event ev equals
$$send(A, B, nc_2, \langle *(nc_2, *(pwd(A), g_1)), h_1(m_1, *(nc_2, *(pwd(A), g_1)), * (nc_2, \oplus(\oplus(*(pwd(A), g_2), m_1), *(m_1, g_1))))\rangle)$$
and $pwd(A)$ is non-compromised, we set
$$S_{ev,tr} = \{nc_2, *(nc_2, g_1)\} \cup \{*(nc_2, *(nc_1, g_1)) \mid * (nc_1, g_1) \in S_{tr}\}.$$
4. In all other cases, we set $S_{ev,tr} = \emptyset$.

The choice of $S_{ev,tr}$ according to this principle implies obviously that $\Phi^{\text{TCAMP}}(S_{ev,tr} \uplus S_{tr}, pwd(A), ev.tr)$ is a consequence of $\Phi^{\text{TCAMP}}(S_{tr}, pwd(A), tr)$. Furthermore, it permits us to obtain in each extension case the required conditions:

- In case (1), we use $\widehat{m} = pwd(A)$. Here, $\Phi^{\text{TCAMP}}(S_{tr}, pwd(A), tr)$ and $pwd(A) \notin S_{tr}$ imply that tr does not include any event $note(ag, \langle A, pwd(A), g_1, g_2\rangle)$ and any event $note(A, \langle pwd(A), g_1, g_2\rangle)$. Thus, the trace tr does not include any subsequent event where $pwd(A)$ is used. This permits us to show $\neg uses(m, pwd(A))$ for all m in $spies(tr)$.
- In case (2), we use $\widehat{m} = nc_1$. Here, the condition $\neg used(nc_1, tr)$ of the first TC-AMP step yields $\neg uses(m, nc_1)$ for all $m \in spies(tr)$.
- In case (3), we use $\widehat{m} = nc_2$. Here, the condition $\neg used(nc_2, tr)$ of the first TC-AMP step yields $\neg uses(m, nc_1)$ for all $m \in spies(tr)$.

9.3 Authentication by the Terminal

TC-AMP allows for the terminal B to authenticate the card A upon receiving and checking message 2. Formalizing this authentication guarantee according to the general principles introduced in Sec. 7.1.7.2 yields the following property.

Property$^{\text{VSE}}$ 92 (Auth_byB):

$$
\begin{aligned}
&(tr \in \text{TCAMP} \wedge A \notin bad \wedge (\forall ag : note(ag, \langle A, pwd(A), g_1, g_2\rangle) \in tr \Rightarrow ag \notin bad) \\
&\quad \wedge send(B, A, nc_1, (nc_1 * g_1) \oplus (\ominus(pwd(A) * g_2))) \in tr \\
&\quad \wedge gets(B, \langle m_2, h_1((nc_1 * g_1) \oplus (\ominus(pwd(A) * g_2)), m_2, \\
&\qquad inv(pwd(A)) * (nc_1 * m_2) \oplus \\
&\qquad inv(pwd(A)) * (((nc_1 * g_1) \oplus (\ominus(pwd(A) * g_2))) * m_2))\rangle) \in tr \\
&\quad \wedge (\forall nc : note(spy, nc * (nc_1 * g_1) \oplus \\
&\qquad\qquad nc * (((nc_1 * g_1) \oplus (\ominus(pwd(A) * g_2))) * g_1)) \notin tr)) \\
&\Rightarrow \\
&(\exists nc_2 : m_2 = nc_2 * pwd(A) * g_1 \wedge \\
&\quad gets(A, (nc_1 * g_1) \oplus (\ominus(pwd(A) * g_2))) \in tr \wedge \\
&\quad send(A, B, nc_2, \langle m_2, h_1((nc_1 * g_1) \oplus (\ominus(pwd(A) * g_2)), m_2, \\
&\qquad nc_2 * ((nc_1 * g_1) \oplus ((nc_1 * g_1) \oplus (\ominus(pwd(A) * g_2))) * g_1))\rangle) \in tr)
\end{aligned}
$$

Obviously, we need to exclude oops-events that disclose the third h_1-part.

This authenticity guarantee for the terminal B is stronger than that of PACE: The peer equals the intended partner and participates in the role of the card; She addresses her message to B.

Property 92 is proved immediately with the help of the corresponding authenticity lemma. The proof of the latter is by induction on TC-AMP traces. The most involved part, described in Sec. 9.5, deals (in the fake case) with the protection of the third h_1-part.

9.4 Authentication by the Card

The opposite authentication, i.e. of B by the card A, is ensued after receiving and checking message 3. Formalizing this authentication guarantee according to the general principles introduced in Sec. 7.1.7.2 yields the following property.

Property[VSE] 93 (Auth_byA):

$$
\begin{array}{l}
(tr \in TCAMP \wedge A \notin bad \wedge (\forall ag : note(ag, \langle A, pwd(A), g_1, g_2\rangle) \in tr \Rightarrow ag \notin bad) \\
\quad \wedge send(A, B, nc_2, \langle nc_2 * (pwd(A) * g_1), h_1(m_1, nc_2 * (pwd(A) * g_1), \\
\qquad nc_2 * ((pwd(A) * g_2) \oplus m_1 \oplus (m_1 * g_1)))\rangle) \in tr \\
\quad \wedge gets(A, h_2(m_1, nc_2 * (pwd(A) * g_1), nc_2 * ((pwd(A) * g_2) \oplus m_1 \oplus (m_1 * g_1)))) \in tr \\
\quad \wedge (\forall nc : note(spy, nc_2 * (nc * g_1) \oplus \\
\qquad\qquad nc_2 * (((nc_* g_1) \oplus (\ominus(pwd(A) * g_2))) * g_1)) \notin tr)) \\
\Rightarrow \\
(\exists nc_1 : m_1 = (nc_1 * g_1) \oplus (\ominus(pwd(A) * g_2)) \wedge \\
\quad gets(B, \langle nc_2 * (pwd(A) * g_1), h_1(m_1, nc_2 * (pwd(A) * g_1), \\
\qquad nc_2 * ((pwd(A) * g_2) \oplus m_1 \oplus (m_1 * g_1)))\rangle) \in tr \wedge \\
\quad says(B, A, h_2(m_1, nc_2 * (pwd(A) * g_1), (nc_1 * (nc_2 * g_1)) \oplus (m_1 * (nc_2 * g_1)))) \in tr)
\end{array}
$$

We need to exclude oops-events that disclose the third h_2-part.

This authenticity guarantee for the card A is stronger than that of PACE: The peer equals the intended partner and participates in the role of the terminal; She addresses her message to A.

Property 93 is proved immediately with the help of the corresponding authenticity lemma. The proof of the latter is by induction on TC-AMP traces. The most involved part, described in Sec. 9.6, deals (in the fake case) with the protection of the third h_2-part.

9.5 Protection of the Third h_1-Part

In this section we describe the main proof arguments to exclude that the h_1-message received by the terminal in the second TC-AMP step is faked by the attacker. We sketch the proof idea in Sec. 9.5.2, where the assumed fake message is reduced to available message parts. In particular, the assumed derivation of the h_1-message by the attacker necessitates to employ a certain $\oplus$-object (the third h_1-part) for its composition. More details to the partial structure of this $\oplus$-object is established with the help of the unicity theorem in Sec. 9.5.3. To prove that the attacker is not able to derive such a $\oplus$-object we employ the invariant 58 about the structure and the conditions of all derivable $\oplus$-objects from the immediately observable messages. Before we describe the proof steps based on this invariant in Sec. 9.5.4, we present (in Sec. 9.5.1) more details to the protocol-specific predicates used to link derivable $\oplus$-objects to regular TC-AMP messages.

9.5.1 Invariant about Derivable $\oplus$-Objects

In this section, we recall the invariant about the derivable $\oplus$-objects from TC-AMP messages, which is introduced in Sec. 6.5.1.2. We start with the definition of the predicates used to specify the different cases.

First, predicate $\text{TCAMP}\_msg_1(m, ik)$ holds for every $\oplus$-object m that originates from a first regular TC-AMP step:

$$
\begin{array}{l}
\text{TCAMP}\_msg_1(m, ik) \Leftrightarrow \\
(\exists nc_1, g_1, g_2, pw(j) : \oplus(*(nc_1, g_1), \ominus(*(pw(j), g_2))) \in ik \wedge \\
\quad g_1 \neq g_2 \wedge nc_1, pw(j), *(nc_1, g_1), *(pw(j), g_2) \notin DY(ik) \wedge \\
\quad (m = \oplus(*(nc_1, g_1), \ominus(*(pw(j), g_2))) \vee m = \oplus(\ominus(*(nc_1, g_1)), *(pw(j), g_2))))
\end{array}
$$

Second, predicate $\text{TCAMP\_}oops(m,ik)$ holds for all $\oplus$-objects m originating from oops events:

$$
\begin{aligned}
&\text{TCAMP\_}oops(m,ik) \Leftrightarrow \\
&(\exists nc_2,nc_1,g_1,g_2,pw(j),m_1,m_{12} : m_1 = \oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))) \wedge \\
&\quad m_{12} = *(nc_2,*(nc_1,g_1)) \wedge nc_1,nc_2,pw(j),m_{12},*(nc_2,g_1) \notin DY(ik) \wedge \\
&\quad *(nc_1,g_1),*(pw(j),g_2) \notin DY(ik) \wedge \oplus(m_{12},*(nc_2,*(m_1,g_1))) \in ik \wedge \\
&\quad (m = \oplus(m_{12},*(nc_2,*(m_1,g_1))) \vee m = \oplus(\ominus(*(inv(m_1),m_{12})),\ominus(*(nc_2,g_1))) \vee \\
&\quad m = \oplus(*(inv(m_1),m_{12}),*(nc_2,g_1)) \vee m = \oplus(\ominus(m_{12}),\ominus(*(nc_2,*(m_1,g_1))))))
\end{aligned}
$$

Third, predicate $\text{TCAMP\_}mrg(m,ik)$ holds for all $\oplus$-objects m that result by merging other $\oplus$-objects originating from regular steps:

$$
\begin{aligned}
&\text{TCAMP\_}mrg(m,ik) \Leftrightarrow \\
&(\exists nc_1,nc_3,g_1,g_2,pw(j) : nc_1 \neq nc_3 \wedge \oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))) \in ik \wedge \\
&\quad \oplus(*(nc_3,g_1),\ominus(*(pw(j),g_2))) \in ik \wedge m = \oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))) \wedge \\
&\quad nc_1,nc_3,pw(j),*(nc_1,g_1),*(nc_3,g_1),*(pw(j),g_2) \notin DY(ik))
\end{aligned}
$$

Using these predicates, we obtain the following invariant:

Property$^{\text{VSE}}$ 94 (Derivable $\oplus$-Objects ; TC-AMP):

$$
\begin{aligned}
&(tr \in \text{TCAMP} \wedge isObj^{\oplus}(\hat{m}) \wedge DY(spies(tr)) \cap ccl_2^{\oplus}(\hat{m}) \neq \emptyset) \Rightarrow \\
&(\exists ms_b,ms_p,n : syn^{\overline{\oplus}}(\hat{m},ms_b \uplus ms_p) \wedge \\
&\quad (\forall m \in ms_b : \neg isObj^{\oplus}(m) \wedge m \in DYl(spies(tr),n)) \wedge \\
&\quad (\forall m \in ms_p : (\exists m',ms : m' \in DYl(spies(tr),n) \wedge ms \subseteq DYl(spies(tr),n) \wedge \\
&\qquad syn^{\overline{*}}(m,ms,m') \wedge (\text{TCAMP\_}msg_1(m',spies(tr)) \vee \\
&\qquad\quad \text{TCAMP\_}mrg(m',spies(tr)) \vee \text{TCAMP\_}oops(m',spies(tr))))))
\end{aligned}
$$

In the following, we explain how $\text{TCAMP\_}msg_1$, TCAMP_*oops* and TCAMP_*mrg* satisfy the requirements $\aleph_1$ and $\aleph_2$ necessary for the provided proof of the invariant (in Sec. 6.5.1.2).

Requirement $\aleph_1$ is about the non-confidential left $*$-parts occurring in the $\oplus$-objects that originate from regular messages according to the given predicates. The $\oplus$-objects m that satisfy $\text{TCAMP\_}msg_1(m,ik)$ or $\text{TCAMP\_}mrg(m,ik)$ include only confidential left $*$-parts. The $\oplus$-objects m that satisfy $\text{TCAMP\_}oops(m,ik)$ have just one non-confidential left $*$-part, which corresponds to a first TC-AMP message $m_1 = \oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2)))$ or to $inv(m_1)$. Here, the definition of $\text{TCAMP\_}oops(m,ik)$ satisfies obviously requirement $\aleph_1$. Take for instance $m = \oplus(m_{12},*(nc_2,*(m_1,g_1)))$, requirement $\aleph_1$ necessitates that $*(inv(m_1),m)$ is covered by one of the given predicates. This is clearly the case, as $\text{TCAMP\_}oops(*(inv(m_1),m),ik)$ ensues from $\text{TCAMP\_}oops(m,ik)$.

Requirement $\aleph_2$ ensures that all $\oplus$-objects (m_{lr}) resulting by merging other $\oplus$-objects (m_l,m_r) linked to regular protocol messages are covered by the protocol-specific predicates according to this principle: After the simplification of the *public common* left $*$-sub-messages of m_{lr} (if any), the resulting m'_{lr} must be covered by the protocol-specific predicates, too. According to the used predicates, we distinguish the following merge-sides:

- m_1 equals $\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2)))$ and m_3 equals $\oplus(\ominus(*(nc_3,g_1)),*(\pi,g_2))$, where TCAMP_$msg_1(m_1,ik)$ and TCAMP_$msg_1(m_3,ik)$ hold: Here, we obtain the $\oplus$-object $\oplus(m_1,m_3) = \oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1)))$ and this resulting message is covered by TCAMP_$mrg(\oplus(m_1,m_3),ik)$. It does not possess any common left $*$-sub-message.
- m_{12} equals $\oplus(*(nc_1,g_1),\ominus(*(nc_2,g_1)))$ and m_2 equals $\oplus(*(nc_2,g_1),\ominus(*(\pi,g_2)))$, where TCAMP_$mrg(m_{12},ik)$ and TCAMP_$msg_1(m_2,ik)$ hold: Here, we obtain the $\oplus$-object $\oplus(m_{12},m_2) = \oplus(*(nc_1,g_1),\ominus(*(\pi,g_2)))$ and this message is covered by TCAMP_$msg_1(\oplus(m_{12},m_2),ik)$. It does not possess any common left $*$-sub-message.
- m_{12} equals $\oplus(*(nc_1,g_1),\ominus(*(nc_2,g_1)))$ and m_{23} equals $\oplus(*(nc_2,g_1),\ominus(*(nc_3,g_1)))$, where TCAMP_$mrg(m_{12},ik)$ and TCAMP_$mrg(m_{23},ik)$ hold: Here, we obtain the $\oplus$-object $\oplus(m_{12},m_{23}) = \oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1)))$ and this message is covered by TCAMP_$mrg(\oplus(m_{12},m_{23}),ik)$. It does not possess any common left $*$-sub-message.

9.5.2 Proof Sketch

The authenticity of the message in step 2 of TC-AMP relies on the central argument that the peer is not able to generate the third h_1-part without employing the message $\widehat{m}_B = \oplus(*(nc_1,g_1),*(\ominus(\pi_A),g_2))$ (sent by the terminal B in step 1) and the password π_A (used in the generation of this message).

In step 2, B receives $pair(M_2,H_1)$ and checks its authenticity by the comparison of H_1 with the own h_1-message $h_1(\widehat{m}_B,M_2,\oplus(*(inv(\pi_A),*(nc_1,M_2)),*(inv(\pi_A),*(\widehat{m}_B,M_2))))$.

In the following we use $\widehat{k}_B$ to denote the third h_1-part.

The nonce nc_1 generated by B in step 1 is a local secret of B and was sent *bound with* the generator g_1 in $\widehat{m}_B$. Thus, the only way to *embed* nc_1 in $\widehat{k}_B$ is to use $\widehat{m}_B$ in its computation. This implies that $*(inv(\pi_A),*(nc_1,M_2)) = *(x_1,\ldots,*(x_n,*(nc_1,g_1))\ldots)$, i.e. $*(inv(\pi_A),M_2) = *(x_1,\ldots,*(x_n,g_1)\ldots)$, and that the peer has used $x_1,\ldots,x_n$ and has extracted $*(nc_1,g_1)$ from $\widehat{m}_B$ to compute $\widehat{k}_B$. The latter necessitates to make use of the crypt-key $*(\pi_A,g_2)$, which cannot be computed by the peer without employing π_A.

The above mentioned binding and regularity arguments are in particular required when handling the fake case in the authenticity proof. It yields to a proof state where

1. the *pair*-message $\widetilde{m}_2 = pair(M_2,h_1(\widehat{m}_B,M_2,\widehat{k}_B))$ does not belong to the observable messages ($\widetilde{m}_2 \notin ik$),
2. and $\widetilde{m}_2$ is *assumed* to be forged by the attacker ($\widetilde{m}_2 \in DY(ik)$).

The proof goal is handled then by contradiction, i.e. tracing back the assumption (in 2) to proof states where basic confidentiality properties are violated. This is done in two major steps where we need to apply our proof technique from Chap. 6.

Step 1: The occurrence of the *pair*-message can be immediately reduced to the occurrence of the parts M_2 and $h_1(\widehat{m}_B,M_2,\widehat{k}_B)$. Afterwards, we reduce the occurrence of the h_1-object to the occurrence of the h_1-parts, in particular the third one. Following our proof technique, we use ccl_2. The applied correctness theorem for h_1-objects is similar to theorem 54. It is used in combination with the following *regularity lemma*. It holds, when the protocol messages do not allow for the *extraction* of a fixed h_1-object, i.e. when they do not include a fixed h_1-object neither in clear-text, nor as a select-part, nor as a crypt-part.

Property$^{\text{VSE}}$ 95 (Protected h_1-Objects):

$$(tr \in \text{TCAMP} \wedge pair(m_1,h_1(m_0,m_1,m_2)) \notin spies(tr) \wedge \neg(m_0,m_1,m_2 \in DY(spies(tr))))$$
$$\Rightarrow spies(tr) \cap ccl_2(h_1(m_0,m_1,m_2)) = \emptyset$$

It is easy to check that definition 52 (of ccl_2), the corresponding correctness theorem for h_1-objects (the counterpart of theorem 54) and the protocol property 95 allow us to conclude from the assumption in (2) and the hypothesis in (1) that the h_1-part $\widehat{k}_B$ belongs to $DY(ik)$.

Step 2: The proof task in this step is to reduce the occurrence of the h_1-part $\widehat{k}_B$ together with M_2 in the intruder knowledge to proof states where basic confidentiality properties are violated. This is done in the following successive case distinctions:

First Case Distinction: We consider whether $inv(\pi_A)$ in $\widehat{k}_B$ can be simplified.

(I) $inv(\pi_A)$ persists in $\widehat{k}_B$, i.e. $\widehat{k}_B$ or a basic $\oplus$-part of $\widehat{k}_B$ has $inv(\pi_A)$ as a left $*$-part: Here, we obtain $\widehat{k}_B \in ccl_2(*(inv(\pi_A),m'))$ with $obj^*(*(inv(\pi_A),m'),inv(\pi_A),m')$ and $\neg isObj^*(m')$. Since this $*$-object $*(inv(\pi_A),m')$ does not match any message part of the protocol, its occurrence in $\widehat{k}_B$ can be simply reduced to the availability of $inv(\pi_A)$ and thus of π_A in the intruder knowledge.

The reduction is carried out with the regularity lemma

$$\begin{aligned}&(tr \in \text{TCAMP} \wedge *(inv(pwd(ag)),m_0) \notin spies(tr) \wedge \neg isObj^*(m_0)\\ &\wedge obj^*(*(inv(pwd(ag)),m_0),inv(pwd(ag)),m_0)\\ &\wedge inv(pwd(ag)) \notin DY(spies(tr)))\\ &\Rightarrow spies(tr) \cap ccl_2(*(inv(pwd(ag)),m_0)) = \emptyset.\end{aligned}$$

It is proved similar to property 95 using the correctness theorem 56 (of ccl_2 applied to $*$-objects).

(II) $inv(\pi_A)$ does not persist in $\widehat{k}_B$, i.e. M_2 or every basic $\oplus$-part of M_2 has π_A as a left $*$-part, which means that $syn^*(M_2,\pi_A,\widehat{M}_2)$ holds for some $\widehat{M}_2$: This allows us to simplify $\widehat{k}_B$ to $\oplus(*(nc_1,\widehat{M}_2),*(\widehat{m}_B,\widehat{M}_2))$, where $\widehat{M}_2$ (and every basic $\oplus$-part of $\widehat{M}_2$) does not have $inv(\pi_A)$ as a left $*$-part.

Second Case Distinction: It is carried out to continue the proof in case (II). We consider whether nc_1 is (partly) simplified in $*(nc_1,\widehat{M}_2)$.

(1) nc_1 is (partly) simplified in $*(nc_1,\widehat{M}_2)$, i.e. $\widehat{M}_2$ or some basic $\oplus$-part of $\widehat{M}_2$ has $inv(nc_1)$ as a left $*$-part: Here, M_2, which equals $*(\pi_A,\widehat{M}_2)$, belongs to $ccl_2(*(inv(nc_1),m))$ with $obj^*(*(inv(nc_1),m),*inv(nc_1),m)$ and $\neg isObj^*(m)$. This allows us to proceed as in case (I) and deduce the availability of $inv(nc_1)$ and thus of nc_1 in the intruder knowledge. The employed regularity lemma will be about $ccl_2(*(inv(nc_1),m))$.

(2) nc_1 is not (partly) simplified in $*(nc_1,\widehat{M}_2)$, i.e. nc_1 is a left $*$-part of $*(nc_1,\widehat{M}_2)$ (and of all $\oplus$-parts of $*(nc_1,\widehat{M}_2)$): This means, $\widehat{M}_2$ in $\widehat{k}_B = \oplus(*(nc_1,\widehat{M}_2),*(\widehat{m}_B,\widehat{M}_2))$ (and every basic $\oplus$-part of $\widehat{M}_2$) does not have neither $inv(\pi_A)$ nor $inv(nc_1)$ as a left $*$-part.

Third Case Distinction: It is carried out to continue the proof in case (2), where we have $\widehat{k}_B = \oplus(*(nc_1,\widehat{M}_2),*(\widehat{m}_B,\widehat{M}_2))$, $M_2 = *(\pi_A,\widehat{M}_2)$ and $\widehat{k}_B, M_2 \in DY(ik)$. Here, we apply first the invariant 94 to reduce $\widehat{k}_B \in DY(ik)$ to the following three complementary cases:

(a) There is a basic $\oplus$-part $*(nc_1,m_1)$ of $\widehat{k}_B$ in $DY(ik)$.

(b) The $\oplus$-object $\widehat{k}_B$ is composed from two basic $\oplus$-parts and there is a $ms \subset DY(ik)$ with $syn^{\overline{*}}(\widehat{k}_B,ms,\oplus(*(nc_1,m_1),*(\widehat{m}_B,m_1)))$ and

(i) $\text{TCAMP\_}msg_1(\oplus(*(nc_1,m_1),*(\widehat{m}_B,m_1)),ik)$,

(ii) $\text{TCAMP\_}mrg(\oplus(*(nc_1,m_1),*(\widehat{m}_B,m_1)),ik)$ or

(iii) $\text{TCAMP\_}oops(\oplus(*(nc_1,m_1),*(\widehat{m}_B,m_1)),ik)$.

(c) The $\oplus$-object $\widehat{k}_B$ possesses more than two basic $\oplus$-parts, i.e. we have $obj^{\oplus}(\widehat{M}_2,m_1,m_2)$ for a basic $\oplus$-part m_1 and a complementary m_2. Furthermore, there is a basic $\oplus$-part m_3 of $\oplus(*(nc_1,m_2),\oplus(*(\widehat{m}_B,m_1),*(\widehat{m}_B,m_2)))$ such that $\oplus(*(nc_1,m_1),m_3) \in DY(ik)$ and there is $ms \subset DY(ik)$ with $syn^{\overline{*}}(\oplus(*(nc_1,m_1),m_3),ms,\oplus(*(nc_1,m_4),m_5))$ and

(i) $\text{TCAMP\_}msg_1(\oplus(*(nc_1,m_4),m_5),ik)$,

(ii) $\text{TCAMP\_}mrg(\oplus(*(nc_1,m_4),m_5),ik)$ or

(iii) $\text{TCAMP\_}oops(\oplus(*(nc_1,m_4),m_5),ik)$.

9.5.3 Unicity Theorem

In Sec. 9.5.4, we describe how cases (a)–(c) are closed by refutation based on basic confidentiality properties or mismatched structures. Many proof situations require to apply the following key property on the occurrence of arbitrary large $*$-objects that have nc_1 as a left $*$-part. It is due to the binding of the nonce nc_1 with the right $*$-part g_1. The property is a binding theorem that reminds of theorem 82 in the PACE proof. We consider arbitrary $*$-objects m that are composed (as specified below with $syn^{\overline{*}}(m,nc_1{\bullet}ms,m')$ introduced in Sec. 6.5.1.1) from $nc_1{\bullet}ms$ and the right $*$-part m', where nc_1 is the nonce generated by a regular participant in step 1. The following binding theorem provides us with all possible alternatives for the attacker to embed nc_1 with the arbitrary many $*$-parts in ms and the right $*$-part m'. Like in the proof of theorem 82, we need to reason on the derivation of m by induction on its $*$-structure. For this purpose, we formulate the occurrence of m with the help of $ccl_2(m)$.

Property[VSE] 96 (Binding the first nonce with the first base):

$$
\begin{aligned}
&(tr \in \text{TCAMP} \wedge send(B,A,nc_1,(nc_1 * g_1) \oplus (\ominus(\pi_A * g_2))) \in tr \\
&\quad \wedge nc_1 \notin DY(spies(tr)) \wedge isObj^*(m) \wedge syn^{\overline{*}}(m,nc_1{\bullet}ms,m') \\
&\quad \wedge \neg isObj^*(m') \wedge DY(spies(tr)) \cap ccl_2(m) \neq \emptyset) \\
&\Rightarrow \\
&(m' = g_1 \wedge \\
&\quad ((\exists nc,ms' : \\
&\qquad note(spy,\oplus(*(nc,*(nc_1,g_1)),*(nc,*(\oplus(*(nc_1,g_1),*(\pi_A,\ominus(g_2))),g_1)))) \in tr \wedge \\
&\qquad ms = nc{\bullet}ms' \wedge ms' \subset DY(spies(tr))) \\
&\quad \vee ms \subset DY(spies(tr))))
\end{aligned}
$$

This property states that each $*$-object m having $nc_1{\bullet}ms$ as the left $*$-parts and m' as the right $*$-part originates

1. either from an oops event, where $*(nc_1,g_1)$ is extended with another $*$-part nc and possibly with further $*$-parts from the intruder knowledge,
2. or from the first message possibly using further $*$-parts from the intruder knowledge.

The binding property 96 is proven similar to theorem 82 by nested induction: The first induction is wrt. the length of ms.

In the base case, where ms is empty and $m = *(nc_1, m')$, we employ property 97 (see below).

In the step case, where ms is not empty, we consider $*$-objects m with at least *two* left $*$-parts. Such a message part matches only the message parts that can be gained by the attacker in oops events. In particular, it does not occur in a second TC-AMP message, as the used nonce is *new* and thus differs from nc_1. These facts are proven by induction on TCAMP traces. In the obtained step case, the conjecture hypothesis includes one proposition of the form

$$DY(\{m_{Tcamp}\} \cup spies(tr)) \cap ccl_2(\overline{*}(nc_1 \bullet ms, m')) \neq \emptyset,$$

where m_{Tcamp} originates from the last event in the extended trace, ms is not empty and where $syn^{\overline{*}}(\overline{*}(nc_1 \bullet ms, m'), nc_1 \bullet ms, m')$ holds.

Applying the correctness theorem 56 of ccl_2 to this part of the conjecture hypothesis, we obtain three cases:

1. There is an observable message that allows us to derive the $*$-object $\overline{*}(nc_1 \bullet ms, m')$ by extraction, i.e. $\{m_{Tcamp}\} \cup spies(tr) \cap ccl_2(\overline{*}(nc_1 \bullet ms, m')) \neq \emptyset$.

2. The intruder knowledge includes a message $m_{(nc_1, ms, m')}$ that allows for the derivation of $\overline{*}(nc_1 \bullet ms, m')$ by extraction; This message must be composed out of nc_1 and another message $m_{(ms,m')}$ from $DY(\{m_{Tcamp}\} \cup spies(tr))$ such that we have $m_{(ms,m')} \in ccl_2(\overline{*}(ms, m'))$.

3. The list ms equals $m_0 \bullet ms_0$ such that there is a message $m_{(m_0, nc_1 \bullet ms_0, m')}$ in the intruder knowledge that allows for the derivation of $\overline{*}(nc_1 \bullet ms, m')$ by extraction; This message must be composed out of m_0 and $m_{(nc_1 \bullet ms_0, m')}$ from $DY(\{m_{Tcamp}\} \cup spies(tr))$ such that $m_{(nc_1 \bullet ms_0, m')} \in ccl_2(\overline{*}(nc_1 \bullet ms_0, m'))$ holds. That is, we get that m_0 belongs to $DY(\{m_{Tcamp}\} \cup spies(tr))$ and $DY(\{m_{Tcamp}\} \cup spies(tr)) \cap ccl_2(\overline{*}(nc_1 \bullet ms_0, m')) \neq \emptyset$ holds.

The first proof state is handled with a further case distinction:

- There is some message in $spies(tr)$ that allows for the derivation of $\overline{*}(nc_1 \bullet ms, m')$ by extraction, i.e. $spies(tr) \cap ccl_2(\overline{*}(nc_1 \bullet ms, m')) \neq \emptyset$. This proof goal is closed with the help of the corresponding induction hypothesis.

- The message m_{Tcamp}, which originates from the last event in the extended trace, allows us to derive $\overline{*}(nc_1 \bullet ms, m')$ by extraction, i.e. $m_{Tcamp} \in ccl_2(\overline{*}(nc_1 \bullet ms, m'))$. Since $\overline{*}(nc_1 \bullet ms, m')$ has at least two left $*$-parts and nc_1 differs from the nonce in the second TC-AMP step, only a message m_{Tcamp} that originates from an oops event fulfills this property. This provides us with the first alternative in the conclusion of theorem 96. Here, we have the specific case where the list ms' is empty.

The second proof state above is closed by contradiction with the assumption that nc_1 is confidential, i.e. $nc_1 \notin DY(\{m_{Tcamp}\} \cup spies(tr))$.

In the third proof state, $DY(\{m_{Tcamp}\} \cup spies(tr)) \cap ccl_2(\overline{*}(nc_1 \bullet ms_0, m')) \neq \emptyset$ where ms_0 is smaller than ms allows us to apply the original induction hypothesis (wrt. the list length). The message lists in the resulting consequences are then extended with m_0 from $DY(\{m_{Tcamp}\} \cup spies(tr))$.

We continue this section with an overview on the base case in the proof of our theorem 96. Since the considered $*$-object $*(nc_1, m')$ can be composed only out of nc_1 and m' and nc_1 is at the same time confidential, the only remaining option for the attacker to derive $*(nc_1, m')$ is to reuse $*(nc_1, g_1)$ from the first message. This means, $m' = g_1$, which we prove in the following unicity theorem by induction on TCAMP traces.

Property[VSE] 97 (Unicity of the right $*$-part associated with a nonce):

$$
\begin{array}{l}
(tr \in \text{TCAMP} \wedge nc \notin DY(spies(tr))) \\
\Rightarrow \\
(\exists \widehat{g} : (\forall m : obj^*(*(nc,m),nc,m) \Rightarrow (spies(tr) \cap ccl_2(*(nc,m)) \neq \emptyset \Rightarrow m = \widehat{g})))
\end{array}
$$

This unicity property of TCAMP holds not only for the nonce nc_1, which is generated in step 1, but also for the second nonce that is generated in step 2.

9.5.4 Refutation by Confidentiality and Constrained Structures

We continue in the following the proof sketched in Sec. 9.5.2. We describe how cases (a)–(c) are closed by refutation based on basic confidentiality properties or mismatched structures.

9.5.4.1 Handling of Case (a):

In the corresponding proof state, the basic $\oplus$-part $*(nc,m_1)$ in $DY(ik)$ permits to apply theorem 96, as $*(nc,m_1)$ is of the form $\overline{*}(nc_1 \bullet ms,m')$ or $\ominus(\overline{*}(nc_1 \bullet ms,m'))$. This yields a first case that refutes the absence of oops-events (in an assumption of property 92) and a second case where the considered ms is a subset of $DY(ik)$. Since the right $*$-part of m, i.e. m', equals g_1, we deduce that $*(nc_1,g_1)$ belongs to $DY(ik)$. This means, we obtain a contradiction with the basic confidentiality property.

9.5.4.2 Handling of Case (b):

In the corresponding proof state, we have the $\oplus$-object $\oplus(*(nc_1,m_1),*(\widehat{m}_B,m_1))$ in $DY(ik)$ satisfying $\text{TCAMP\_}msg_1$, TCAMP_mrg or TCAMP_$oops$, which yields the following cases:

(i) $\text{TCAMP\_}msg_1(\oplus(*(nc_1,m_1),*(\widehat{m}_B,m_1)),ik)$: According to the definition of predicate $\text{TCAMP\_}msg_1$, we obtain $obj^{\oplus}(\oplus(*(nc_1,m_1),*(\widehat{m}_B,m_1)),*(nc'_1,g'_1),\ominus(*(pw(j),g'_2)))$ or $obj^{\oplus}(\oplus(*(nc_1,m_1),*(\widehat{m}_B,m_1)),\ominus(*(nc'_1,g'_1)),*(pw(j),g'_2))$. In both cases, we have mismatched structures, as $pw(j)$ does not match neither nc_1 nor $\widehat{m}_B$.

(ii) TCAMP_$mrg(\oplus(*(nc_1,m_1),*(\widehat{m}_B,m_1)),ik)$: According to the above definition of predicate TCAMP_mrg, we obtain $obj^{\oplus}(\oplus(*(nc_1,m_1),*(\widehat{m}_B,m_1)),*(nc'_1,g'_1),\ominus(*(nc'_3,g'_1)))$. Similarly, we have mismatched structures, as $\widehat{m}_B$ does not match neither nc'_1 nor nc'_3.

(iii) TCAMP_$oops(\oplus(*(nc_1,m_1),*(\widehat{m}_B,m_1)),ik)$: According to TCAMP_$oops(m_1,ik)$, we may focus on two cases:

- In case the messages $*(nc'_2,*(nc'_1,g'_1))$ and $*(nc'_2,*(\widehat{m}_1,g'_1))$ (or respectively the messages $\ominus(*(nc'_2,*(nc'_1,g'_1)))$ and $\ominus(*(nc'_2,*(\widehat{m}_1,g'_1))))$ are complementary $\oplus$-parts of $\oplus(*(nc_1,m_1),*(\widehat{m}_B,m_1))$, we obtain $nc_1 = nc'_1$, $\widehat{m}_B = \widehat{m}_1$ and $m_1 = *(nc'_2,g'_1)$ or $m_1 = \ominus(*(nc'_2,g'_1))$ for $\widehat{m}_1 = \oplus(*(nc'_1,g'_1),\ominus(*(pw(j),g'_2)))$. This refutes (an assumption of property 92 about) the absence of oops-events.
- In case the messages $*(inv(\widehat{m}_1),*(nc'_2,*(nc'_1,g'_1)))$ and $*(nc'_2,g'_1)$ (or respectively $\ominus(*(inv(\widehat{m}_1),*(nc'_2,*(nc'_1,g'_1))))$ and $\ominus(*(nc'_2,g'_1))$ are complementary $\oplus$-parts of $\oplus(*(nc_1,m_1),*(\widehat{m}_B,m_1))$, we have mismatched structures, as $\widehat{m}_B$ does not match neither a nonce, i.e. nc'_1 or nc'_2, nor $inv(\widehat{m}_1)$.

9.5.4.3 Handling of Case (c):

In the corresponding proof state, we have the $\oplus$-object $\oplus(*(nc_1,m_1),m_3)$ in $DY(ik)$, where m_3 is a basic $\oplus$-part of $\oplus(*(nc_1,m_2),\oplus(*(\widehat{m}_B,m_1),*(\widehat{m}_B,m_2)))$. Furthermore, we have $syn^{\overline{*}}(\oplus(*(nc_1,m_1),m_3),ms,\oplus(*(nc_1,m_4),m_5))$ for $ms \subset DY(ik)$ and for $\oplus(*(nc_1,m_4),m_5)$ satisfying TCAMP_msg_1, TCAMP_mrg or TCAMP_$oops$, which yields the following cases:

(i) TCAMP_$msg_1(\oplus(*(nc_1,m_4),m_5),ik)$: According to the above definition of this predicate TCAMP_msg_1, we obtain $obj^{\oplus}(\oplus(*(nc_1,m_4),m_5),*(nc'_1,g'_1),\ominus(*(pw(j),g'_2)))$ or $obj^{\oplus}(\oplus(*(nc_1,m_4),m_5),\ominus(*(nc'_1,g'_1)),*(pw(j),g'_2))$, for generators $g'_1 \neq g'_2$. W.l.o.g., we focus on the latter case, which yields the equalities $m_4 = \ominus(g'_1)$, $nc_1 = nc'_1$ and $m_5 = *(pw(j),g'_2)$. This implies the equalities $m_1 = \ominus(\overline{*}(ms,g'_1))$ and $m_3 = \overline{*}(ms,*(pw(j),g'_2))$. That is, the message part $\overline{*}(ms,*(pw(j),g'_2))$ is a basic $\oplus$-part of $\oplus(*(nc_1,m_2),\oplus(*(\widehat{m}_B,\ominus(\overline{*}(ms,g'_1))),*(\widehat{m}_B,m_2)))$. Since we have the disequality $\overline{*}(ms,*(pw(j),g'_2)) \neq \ominus(*(\widehat{m}_B,\overline{*}(ms,g'_1)))$, it follows that $\overline{*}(ms,*(pw(j),g'_2))$ equals or is a basic $\oplus$-part of

1. $*(nc_1,m_2)$
2. or $*(\widehat{m}_B,m_2)$.

In case (1), we deduce $nc_1 \in ms$ and thus $nc_1 \in DY(ik)$, which refutes the confidentiality of nc_1.

In case (2), we deduce that m_2 equals or has $\overline{*}(ms_2,*(pw(j),g'_2))$ as a basic $\oplus$-part, for $ms = \widehat{m}_B \uplus ms_2$. W.l.o.g., we focus on the case where $m_2 = \overline{*}(ms_2,*(pw(j),g'_2))$. Here, we get $\oplus(*(nc_1,\overline{*}(ms_2,*(pw(j),g'_2))),*(\widehat{m}_B,\ominus(\overline{*}(ms,g'_1)))) \in DY(ik)$ and this implies $DY(ik) \cap ccl_2(*(nc_1,\overline{*}(ms_2,*(pw(j),g'_2)))) \neq \emptyset$, which permits to apply theorem 96 and get $g'_2 = g'_1$, which refutes $g'_1 \neq g'_2$.

(ii) TCAMP_$mrg(\oplus(*(nc_1,m_4),m_5),ik)$: According to the definition of TCAMP_mrg, we obtain $obj^{\oplus}(\oplus(*(nc_1,m_4),m_5),*(nc'_1,g'_1),\ominus(*(nc'_3,g'_1)))$, for nonces $nc'_1 \neq nc'_3$. This yields the equalities $*(nc_1,m_4) = *(nc'_1,g'_1)$ and $m_5 = \ominus(*(nc'_3,g'_1))$ or the equalities $m_5 = *(nc'_1,g'_1)$ and $*(nc_1,m_4) = \ominus(*(nc'_3,g'_1))$.

W.l.o.g., we focus on the former case, which implies $nc_1 = nc'_1$, $m_4 = g'_1$, $m_1 = \overline{*}(ms,g'_1)$ and $m_3 = \ominus(\overline{*}(ms,*(nc'_3,g'_1)))$. That is, $\ominus(\overline{*}(ms,*(nc'_3,g'_1)))$ is a basic $\oplus$-part of $\oplus(*(nc'_1,m_2),\oplus(*(\widehat{m}_B,\overline{*}(ms,g'_1)),*(\widehat{m}_B,m_2)))$. Since, $\ominus(\overline{*}(ms,*(nc'_3,g'_1))) \neq *(\widehat{m}_B,\overline{*}(ms,g'_1))$, it follows that $\ominus(\overline{*}(ms,*(nc'_3,g'_1)))$ equals or is a basic $\oplus$-part of

1. $*(nc'_1,m_2)$
2. or $*(\widehat{m}_B,m_2)$.

In case (1), we deduce based on $nc'_3 \neq nc'_1$ that nc'_1 must be in ms. This refutes the confidentiality of nc'_1, as ms is a subset of $DY(ik)$.

In case (2), we deduce that message m_2 equals or has $\ominus(\overline{*}(ms_2,*(nc'_3,g'_1)))$ as a basic $\oplus$-part, for $ms = \widehat{m}_B \uplus ms_2$. This implies that message $*(nc'_1,m_2)$ equals or has $\ominus(\overline{*}(ms_3,*(nc'_1,*(nc'_3,g'_1))))$ as a basic $\oplus$-part, for $ms_2 = nc'_1 \uplus ms_3$. Hence, the confidentiality of nc'_1 is refuted, because ms_2 is a subset of $DY(ik)$.

(iii) TCAMP_$oops(\oplus(*(nc_1,m_4),m_5),ik)$: According to TCAMP_$oops$, we have for some $\widehat{m}'_B = \oplus(*(nc'_1,g'_1),\ominus(*(pw(j),g'_2)))$ two couples of cases:

1. $obj^{\oplus}(\oplus(*(nc_1,m_4),m_5),*(nc'_2,*(nc'_1,g'_1)),*(nc'_2,*(\widehat{m}'_B,g'_1)))$
   or $obj^{\oplus}(\oplus(*(nc_1,m_4),m_5),\ominus(*(nc'_2,*(nc'_1,g'_1))),\ominus(*(nc'_2,*(\widehat{m}'_B,g'_1))))$
2. and $obj^{\oplus}(\oplus(*(nc_1,m_4),m_5),*(inv(\widehat{m}'_B),*(nc'_2,*(nc'_1,g'_1))),*(nc'_2,g'_1))$
   or $obj^{\oplus}(\oplus(*(nc_1,m_4),m_5),\ominus(*(inv(\widehat{m}'_B),*(nc'_2,*(nc'_1,g'_1)))),\ominus(*(nc'_2,g'_1)))$.

W.l.o.g., we focus in case (1) on the second alternative, which yields two cases:

(1-a) $*(nc_1,m_4) = \ominus(*(nc'_2,*(nc'_1,g'_1)))$ and $m_5 = \ominus(*(nc'_2,*(\widehat{m}'_B,g'_1)))$: Here, nc_1 matches nc'_2 or nc'_1.
When $nc_1 = nc'_2$, the oops message $\oplus(*(nc_1,*(nc'_1,g'_1)),*(nc_1,*(\widehat{m}'_B,g'_1)))$ in ik permits to apply theorem 96 where $ms = \{nc'_1\}$. This yields a first case that refutes the absence of oops-events (in an assumption of property 92) and a second case where the considered ms is a subset of $DY(ik)$. Hence, we get $nc'_1 \in DY(ik)$, which refutes the confidentiality of nc'_1 guaranteed by TCAMP_*oops*.
When $nc_1 = nc'_1$, we proceed similarly. Here, we get $nc'_2 \in DY(ik)$, which refutes the confidentiality of nc'_2 guaranteed by TCAMP_*oops*.

(1-b) $*(nc_1,m_4) = \ominus(*(nc'_2,*(\widehat{m}'_B,g'_1)))$ and $m_5 = \ominus(*(nc'_2,*(nc'_1,g'_1)))$: Here, nc_1 matches nc'_2 as in the first alternative of case (1-a). The proof is similar.

W.l.o.g., we focus in case (2) on the first alternative, which yields two cases:

(2-a) $*(nc_1,m_4) = *(inv(\widehat{m}'_B),(*(nc'_2,*(nc'_1,g'_1))))$ and $m_5 = *(nc'_2,g'_1))$: Here, nc_1 matches nc'_2 or nc'_1, as in case (1-a). The proof is similar.

(2-b) $*(nc_1,m_4) = *(nc'_2,g'_1)$ and $m_5 = *(inv(\widehat{m}'_B),*(nc'_2,*(nc'_1,g'_1)))$: Here, the nonce nc_1 matches nc'_2 as in case (1-b). The proof is similar.

9.6 Protection of the Third h_2-Part

In this section we describe the main proof arguments to exclude that the h_2-message received by the card in the third TC-AMP step is faked by the attacker.

9.6.1 Proof Sketch

The authenticity of the message in step 3 relies on the central argument, that the peer is not able to generate the third h_2-part without employing the message $\widehat{m}_A = *(nc_2,*(\pi_A,g_1))$ (sent by the card A as the first message part in step 2) and the password π_A (used in the generation of this message).

In step 3, A receives an arbitrary message $\widehat{m_3}$ and checks its authenticity by the comparison with the own h_2-message $h_2(\widehat{m_1},\widehat{m}_A,*(nc_2,\oplus(\oplus(*(\pi_A,g_2),\widehat{m_1}),*(\widehat{m_1},g_1))))$, where $\widehat{m_1}$ is received by A in the first TC-AMP step.

In the following we use $\widehat{k}_A$ to denote the third h_2-part.

The nonce nc_2 generated by A in step 2 is a local secret of A and was sent *bound with* the generator g_1 in $\widehat{m}_A$. Thus, the only way to *embed* nc_2 in $\widehat{k}_A$ is to use $\widehat{m}_A$ in its computation. In particular, $\widehat{k}_A$ contains the message part $*(nc_2,*(\widehat{m_1},g_1))$ and this implies that the peer has used $\widehat{m_1}$ and has extracted $*(nc_2,g_1)$ from $\widehat{m}_A$ to compute $\widehat{k}_A$. The latter necessitates to make use of the crypt-key $inv(\pi_A)$, which cannot be computed by the peer without employing π_A.

The above mentioned binding and regularity arguments are in particular required when handling the fake case in the authenticity proof. It yields to a proof situation where

1. $\widehat{m_3} = h_2(\widehat{m_1},\widehat{m}_A,\widehat{k}_A)$ does not belong to the observable messages ($\widehat{m}_3 \notin ik$),
2. and $\widehat{m}_3$ is *assumed* to be forged by the attacker ($\widehat{m}_3 \in DY(ik)$).

The proof goal is handled then by contradiction, i.e. tracing back the assumption (in 2) to proof states where basic confidentiality properties are violated. This is done in two major steps where we need to apply our proof technique from Chap. 6.

Step 1: We reduce the occurrence of the h_2-object to the occurrence of the h_2-parts, in particular the third one. Following our proof technique, we use ccl_2. The applied correctness theorem for h_2-objects is similar to theorem 54. It is used in combination with the following *regularity lemma*. It holds, when the protocol messages do not allow for the extraction of a fixed h_2-object.

Property$^{\text{VSE}}$ 98 (Absence of protected h_2-Objects):

$$(tr \in \text{TCAMP} \wedge h_2(m_0,m_1,m_2) \notin spies(tr) \wedge \neg(m_0,m_1,m_2 \in DY(spies(tr))))$$
$$\Rightarrow spies(tr) \cap ccl_2(h_2(m_0,m_1,m_2)) = \emptyset$$

It is easy to check that definition 52 (of ccl_2), the corresponding correctness theorem for h_2-objects (the counterpart of theorem 54) and the protocol property 98 allow us to conclude from the assumption in (2) and the hypothesis in (1) that the h_2-part $\widehat{k}_A$ belongs to $DY(ik)$.

Step 2: The proof task in this step consists in reducing the occurrence of the h_2-part $\widehat{k}_A = \oplus(*(nc_2,*(\pi_A,g_2)),\oplus(*(nc_2,\widehat{m_1}),*(nc_2,*(\widehat{m_1},g_1))))$ together with $\widehat{m_1}$ in the intruder knowledge to proof states where basic confidentiality properties are violated. This is done in the following successive case distinctions:

First Case Distinction: We consider whether $*(\pi_A,g_2)$ in $\widehat{k}_A$ can be simplified.

(I) The message part $*(nc_2,*(\pi_A,g_2))$ persists as a $\oplus$-part in $\widehat{k}_A$, which equally means that $obj^{\oplus}(\widehat{k}_A,*(nc_2,*(\pi_A,g_2)),\oplus(*(nc_2,\widehat{m_1}),*(nc_2,*(\widehat{m_1},g_1))))$ holds: Here, $\widehat{k}_A$ belongs to $ccl_2(*(nc_2,*(\pi_A,g_2)))$. Since there is no message part of the protocol that matches this $*$-object, we proceed similar to case (I) in Sec. 9.5.2 and deduce the availability of nc_2 or π_A in the intruder knowledge. In both cases, we obtain a contradiction with the basic confidentiality property.

The employed regularity lemma will be clearly about $ccl_2(*(nc_2,*(\pi_A,g_2)))$.

(II) The message part $*(nc_2,*(\pi_A,g_2))$ does not persist as a $\oplus$-part in $\widehat{k}_A$, that is, $obj^{\oplus}(\widehat{k}_A,*(nc_2,*(\pi_A,g_2)),\oplus(*(nc_2,\widehat{m_1}),*(nc_2,*(\widehat{m_1},g_1))))$ does not hold. In this setting, $\widehat{m_1}$ either equals $\ominus(*(\pi_A,g_2))$ or must be a $\oplus$-object composed out of the basic $\oplus$-part $\ominus(*(\pi_A,g_2))$ and a complementary $\oplus$-part $\widetilde{m_1}$. The former case yields that $\widehat{k}_A$ equals $*(nc_2,*(\widehat{m_1},g_1))$ with $\widehat{m_1},\widehat{k}_A \in DY(ik)$ and this refutes the confidentiality of $*(nc_2,g_1)$. In the latter case, we have $obj^{\oplus}(\widehat{m_1},\ominus(*(\pi_A,g_2)),\widetilde{m_1})$, where the message $*(\pi_A,g_2)$ is not a $\oplus$-part of $\widetilde{m_1}$. The third h_2-part $\widehat{k}_A$ equals $\oplus(*(nc_2,\widetilde{m_1}),*(nc_2,*(\oplus(\ominus(*(\pi_A,g_2)),\widetilde{m_1}),g_1)))$.

Second Case Distinction: It is carried out to continue the proof in case (II). We consider whether nc_2 persists in $*(nc_2,\widetilde{m_1})$.

(1) nc_2 does not persist in $*(nc_2,\widetilde{m_1})$, i.e. $inv(nc_2)$ is a left $*$-part of $\widetilde{m_1}$ or $\widetilde{m_1}$ possesses a basic $\oplus$-part having $inv(nc_2)$ as a left $*$-part: Here, $\widehat{m_1}$, which equals $\oplus(\ominus(*(\pi_A,g_2)),\widetilde{m_1})$, belongs to $ccl_2(*(inv(nc_2),m))$ for m as the right $*$-part of $*(inv(nc_2),m)$. This allows us to proceed as in case (I) and deduce the availability of $inv(nc_2)$ and thus of nc_2 in the intruder knowledge. The employed regularity lemma will be about $ccl_2(*(inv(nc_2),m))$.

(2) nc_2 persists in $*(nc_2,\widetilde{m_1})$, i.e. nc_2 is a left $*$-part of $*(nc_2,\widetilde{m_1})$ or $*(nc_2,\widetilde{m_1})$ corresponds to a $\oplus$-object where all basic $\oplus$-parts possess nc_2 as a left $*$-part: This means, all basic $\oplus$-parts of $\widehat{k}_A = \oplus(*(nc_2,\widetilde{m_1}),*(nc_2,*(\widehat{m_1},g_1)))$ possess nc_2 as a left $*$-part.

Since $\widehat{m_1}$ is in the knowledge set $DY(ik)$, we focus in the rest of the proof on $\widehat{k}'_A = \oplus(*(nc_2,*(inv(\widehat{m_1}),\widetilde{m_1})),*(nc_2,g_1))$.

Third Case Distinction: It is carried out to continue the proof in case (2). Here, we apply first the invariant 94 about the derivation of $\oplus$-objects and then reduce $\widehat{k}'_A \in DY(ik)$ to the following three complementary cases:

(a) The basic $\oplus$-part $*(nc_2,g_1)$ of $\widehat{k}'_A$ is in $DY(ik)$. This case is refuted immediately by the basic confidentiality property.

(b) $\widehat{k}'_A$ is composed from two basic $\oplus$-parts with $syn^{\overline{*}}(\widehat{k}'_A,\emptyset,\widehat{k}'_A)$ and

  (i) TCAMP_$msg_1(\widehat{k}'_A,ik)$,

  (ii) TCAMP_$mrg(\widehat{k}'_A,ik)$ or

  (iii) TCAMP_$oops(\widehat{k}'_A,ik)$.

(c) $\widehat{k}'_A$ possesses more than two basic $\oplus$-parts, i.e. we have $obj^{\oplus}(*(inv(\widehat{m_1}),\widetilde{m_1}),m_1,m_2)$ for a basic $\oplus$-part m_1. Furthermore, we have $\oplus(*(nc_2,g_1),*(nc_2,m_1)) \in DY(ik)$, $syn^{\overline{*}}(\oplus(*(nc_2,g_1),*(nc_2,m_1)),\emptyset,\oplus(*(nc_2,g_1),*(nc_2,m_1)))$ and

  (i) TCAMP_$msg_1(\oplus(*(nc_2,g_1),*(nc_2,m_1)),ik)$,

  (ii) TCAMP_$mrg(\oplus(*(nc_2,g_1),*(nc_2,m_1)),ik)$ or

  (iii) TCAMP_$oops(\oplus(*(nc_2,g_1),*(nc_2,m_1)),ik)$.

9.6.2 Unicity Theorem

In Sec. 9.6.3, we describe how cases (b) and (c) are closed by refutation based on basic confidentiality properties and structural constraints. Many proof situations require to apply the following key property on the occurrence of arbitrary large $*$-objects that have nc_2 as a left $*$-part. It is a binding theorem that reminds of theorem 82 in the PACE proof and of theorem 96. For a $*$-object $\overline{*}(ms,*(m',*(nc_2,g_1)))$, which contains a left $*$-part nc_2 (generated by a regular participant in step 2) together with a second confidential left $*$-part m', our theorem provides us with all possible alternatives for the attacker to embed nc_2 and m' with the arbitrary many left $*$-parts in ms and the right $*$-part g_1. Like in the proofs of theorems 82 and 96, we need to reason on the derivation of $\overline{*}(ms,*(m',*(nc_2,g_1)))$ by induction on its $*$-structure. For this purpose, we formulate the occurrence of $m = \overline{*}(ms,*(m',*(nc_2,g_1)))$ with the help of $ccl_2(m)$.

Property VSE 99 (Binding the second nonce with another secret $*$-part):

$$
\begin{aligned}
&(tr \in \text{TCAMP} \wedge send(A,B,nc_2,pair(*(nc_2,*(\pi_A,g_1)),\\
&\quad h_1(m_1,*(nc_2,*(\pi_A,g_1)),*(nc_2,\oplus(\oplus(*(\pi_A,g_2),m_1),*(m_1,g_1)))))) \in tr\\
&\wedge nc_2,m' \notin DY(spies(tr)) \wedge isObj^*(m) \wedge DY(spies(tr)) \cap ccl_2(m) \neq \emptyset\\
&\wedge (\exists m'' : syn^{\overline{*}}(m,ms \uplus \{m',nc_2\},m'') \wedge \neg isObj^*(m')))\\
&\Rightarrow\\
&(ms \subset DY(spies(tr)) \wedge\\
&((\exists nc :\\
&\quad note(spy,\oplus(*(nc_2,*(nc,g_1)),*(nc_2,*(\oplus(*(nc,g_1),*(\pi_A,\ominus(g_2))),g_1)))) \in tr \wedge\\
&\quad m' = nc) \vee\\
&m' = \pi_A))
\end{aligned}
$$

This property states that $*(m',*(nc_2,g_1))$ in the $*$-object $\overline{*}(ms,*(m',*(nc_2,g_1)))$ originates

1. from an oops event,
2. or from the first part of a second message.

In both cases, the set ms includes only left $*$-parts from the intruder knowledge.

The proof of theorem 99 is by nested induction, similar to the proofs of theorems 96 and 82. The first induction is wrt. the length of ms. The base case as well as the step case are proven by induction on TC-AMP traces.

9.6.3 Refutation by Confidentiality and Structural Constraints

We continue in the following the proof sketched in Sec. 9.6.1. We describe how cases (b) and (c) are closed by refutation based on basic confidentiality properties or mismatched structures.

9.6.3.1 Handling of Case (b):

In the corresponding proof state, we have the $\oplus$-object $\oplus(*(nc_2,g_1),*(nc_2,*(inv(\widehat{m_1}),\widetilde{m_1})))$ in $DY(ik)$ satisfying TCAMP_msg_1, TCAMP_mrg or TCAMP_$oops$, which yields the following cases:

(i) TCAMP_$msg_1(\oplus(*(nc_2,g_1),*(nc_2,*(inv(\widehat{m_1}),\widetilde{m_1}))),ik)$: According to TCAMP_$msg_1$, we obtain $obj^{\oplus}(\oplus(*(nc_2,g_1),*(nc_2,*(inv(\widehat{m_1}),\widetilde{m_1}))),*(nc'_1,g'_1),\ominus(*(pw(j),g'_2)))$ or $obj^{\oplus}(\oplus(*(nc_2,g_1),*(nc_2,*(inv(\widehat{m_1}),\widetilde{m_1}))),\ominus(*(nc'_1,g'_1)),*(pw(j),g'_2))$. In both cases, we have mismatched structures, as $pw(j)$ does not match nc_2.

(ii) TCAMP_$mrg(\oplus(*(nc_2,g_1),*(nc_2,*(inv(\widehat{m_1}),\widetilde{m_1}))),ik)$: According to TCAMP_$mrg$, we obtain $obj^{\oplus}(\oplus(*(nc_2,g_1),*(nc_2,*(inv(\widehat{m_1}),\widetilde{m_1}))),*(nc'_1,g'_1),\ominus(*(nc'_3,g'_1)))$ for $nc'_1 \neq nc'_3$. The structural constraints imply $nc'_1 = nc_2$ and $nc'_3 = nc_2$ and this refutes $nc'_1 \neq nc'_3$.

(iii) TCAMP_$oops(\widehat{k}'_A,ik)$ for $\widehat{k}'_A = \oplus(*(nc_2,g_1),*(nc_2,*(inv(\widehat{m_1}),\widetilde{m_1})))$: According to the definition of TCAMP_$oops(m_1,ik)$, we may focus on two cases:

- In case the messages $*(nc'_2,*(nc'_1,g'_1))$ and $*(nc'_2,*(\widehat{m}'_1,g'_1))$ (or respectively $\ominus(*(nc'_2,*(nc'_1,g'_1)))$ and $\ominus(*(nc'_2,*(\widehat{m}'_1,g'_1)))$) are complementary $\oplus$-parts of $\widehat{k}'_A$, we have mismatched structures, as $*(nc_2,g_1)$ does not match any candidate basic $\oplus$-part.
- In case the messages $*(inv(\widehat{m}_1),*(nc'_2,*(nc'_1,g'_1)))$ and $*(nc'_2,g'_1)$ (or respectively $\ominus(*(inv(\widehat{m}_1),*(nc'_2,*(nc'_1,g'_1))))$ and $\ominus(*(nc'_2,g'_1))$) are complementary $\oplus$-parts of $\widehat{k}'_A$, we obtain the equality $nc_2 = nc'_2$. That is, the oops message $\oplus(*(nc_2,*(nc'_1,g'_1)),*(nc_2,*(\widehat{m}'_B,g'_1)))$ in ik permits to apply theorem 99 where $m' = nc'_1$ and $ms = \emptyset$. This yields a first case that refutes the absence of oops-events (in an assumption of property 93) and a second case where $m' = \pi_A$, i.e. $nc'_1 = \pi_A$, refutes $nc'_1 \neq \pi_A$.

9.6.3.2 Handling of Case (c):

In the corresponding proof state, we have the $\oplus$-object $\oplus(*(nc_2,g_1),*(nc_2,m_1))$ in $DY(ik)$ satisfying TCAMP_msg_1, TCAMP_mrg or TCAMP_$oops$. Here, $*(nc_2,m_1)$ is handled similar to $*(nc_2,*(inv(\widehat{m_1}),\widetilde{m_1}))$, as nc_2 cannot be simplified in both cases. For that reason, case (c) is handled the same way as case (b).

9.7 Forward Secrecy of Session Keys

The authentication guarantees of TC-AMP imply that the participants A and B share (already in step 2) the third h_1- and h_2-part, which we denote by $\widehat{k_{AB}}$. Due to its confidentiality, $\widehat{k_{AB}}$ can be used as key material to generate the session key used afterwards for the secure transmission of the application data between A and B. Hence, the confidentiality of the session key follows immediately from the confidentiality of $\widehat{k_{AB}}$, which belongs in turn to the central arguments of the authenticity proofs. In this section, we are interested in a stronger confidentiality property, i.e. the forward secrecy of $\widehat{k_{AB}}$, which implies as well that of the session key.

According to the authentication properties, $\widehat{k_{AB}}$ is uniquely determined through the first message by B and the response of A. Furthermore, it is necessary to assume additionally that the key is not disclosed accidentally. In this way, we obtain the following formalization for the intended forward secrecy property:

Property$^{\text{VSE}}$ 100 (Forward Secrecy):

$$
\begin{aligned}
&(tr \in \text{TCAMP} \wedge A,B \notin bad \wedge \\
&\ send(B,A,nc_1,\oplus(*(nc_1,g_1),*(pwd(A),\ominus(g_2)))) \in tr \wedge \\
&\ send(A,B,nc_2,\langle *(nc_2,*(pwd(A),g_1)), \\
&\qquad h_1(\oplus(*(nc_1,g_1),*(pwd(A),\ominus(g_2))),*(nc_2,*(pwd(A),g_1)), \\
&\qquad \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(\oplus(*(nc_1,g_1),*(pwd(A),\ominus(g_2))),g_1))))\rangle) \in tr \\
&\ \wedge note(spy,\oplus(*(nc_2,*(nc_1,g_1)), \\
&\qquad\qquad *(nc_2,*(\oplus(*(nc_1,g_1),*(pwd(A),\ominus(g_2))),g_1)))) \notin tr) \\
&\Rightarrow \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(\oplus(*(nc_1,g_1),*(pwd(A),\ominus(g_2))),g_1)) \\
&\quad \notin DY(\{pwd(A)\} \cup spies(tr))
\end{aligned}
$$

The exclusion of the *note* event implies that we are dealing with a secret $\oplus$-object having a basic $\oplus$-part $*(nc_2,*(nc_1,g_1))$ that does not occur in any protocol message. According to our proof technique from Chap. 6, we may then prove the property 100 with the help of the *ccl*-function ccl_2 and the correctness theorem 56: Abbreviating $pwd(A)$ with π_A, we assume $\widehat{k_{AB}} \in DY(\{\pi_A\} \cup spies(tr))$. This permits to obtain $DY(\{\pi_A\} \cup spies(tr)) \cap ccl_2(*(nc_2,*(nc_1,g_1))) \neq \emptyset$ and to apply the correctness theorem 56, which yields two cases:

1. $(\{\pi_A\} \cup spies(tr)) \cap ccl_2(*(nc_2,*(nc_1,g_1))) \neq \emptyset$
2. $nc_2 \in DY(\{\pi_A\} \cup spies(tr))$ and $DY(\{\pi_A\} \cup spies(tr)) \cap ccl_2(*(nc_1,g_1)) \neq \emptyset$ or $nc_1 \in DY(\{\pi_A\} \cup spies(tr))$ and $DY(\{\pi_A\} \cup spies(tr)) \cap ccl_2(*(nc_2,g_1)) \neq \emptyset$.

Both cases are refuted based on the (forward) secrecy of nc_1 and nc_2, as formulated in properties 90-(3) and (4). In contrast to the second case, the first case necessitates to employ a corresponding regularity property of TC-AMP, where the absence of the oops-event and $nc_1, nc_2 \notin DY(\{\pi_A\} \cup spies(tr))$ imply $(\{\pi_A\} \cup spies(tr)) \cap ccl_2(*(nc_2,*(nc_1,g_1))) = \emptyset$.

Part III

Handling Indistinguishability Properties

Chapter 10

Dealing with Indistinguishability Properties

In general, cryptographic protocols based on (non-trivial) message algebras aim at security goals that can be only defined in terms of indistinguishability properties. For instance, the resistance against offline testing attacks of PACE and TC-AMP is a typical indistinguishability property. It means that the attacker is not able to distinguish (through offline computations) between the extension of the observable messages with a *correct* password (π_A belonging to a participant A) and that with a *false* password (π' non-used by any participant). That is, the knowledge bases $\pi_A.\overline{ik}$ and $\pi'.\overline{ik}$ *look equivalent to (are indistinguishable by) the attacker*, for any *list* $\overline{ik}$ of observable messages.[1]

Basically, all indistinguishability properties, including those introduced in Sec. 1.1.2.4, are defined in terms of *associated* knowledge bases kb (with the *genuine* information) and kb' (with the *false* or *changed* information). They require that kb and kb' may not be distinguishable through offline computations. For the verification of these properties we thus need to reason on all *relevant* offline computations and to link the use of kb and kb'. In this chapter, we first motivate the restriction of the relevant offline computations to DY derivations and equality checks (Sec. 10.1). Then, we describe how the use of kb and kb' can be linked based on the notion of *generic* DY derivations. This allows us to prove the indistinguishability of kb and kb' through a tailored bijective relation (*a simulation relation*) from $DY(kb)$ to $DY(kb')$ (Sec. 10.2).

In order for the tailored relation to imply the indistinguishability, it has to map the result of *each* generic derivation using kb to the result of the *same* generic derivation using kb'. For an inductive proof of this property, we come up with an appropriate enumeration of the generic derivations (Sec. 10.3). This yields to an *extensional* definition of the DY knowledge, which enriches derivable messages with corresponding derivation trees. Like the intentional definition (cp. Def. 13 in Sec. 3.3), the extensional definition is specific to the algebra and can be defined according to the same schema (Sec. 10.4). The definition of the extensional DY knowledge forms the algebra-specific formal framework for the inductive proof of the mentioned necessary property of the used simulation relation. This framework allows us to explore (by case distinctions on the possible operations) *how* this property is (can be) enforced. In doing so, we acquire as described in Chap. 11 and 13 general, necessary and sufficient conditions. These are then proven employing regularity properties of the protocol, which are *separately* verified by induction on protocol traces, as described in Part II.

[1] We use the notation $\overline{ik}$ and $\pi'.\overline{ik}$ to emphasize that $\overline{ik}$ and $\pi'.\overline{ik}$ are the list representations of the finite message sets ik and $\{\pi'\} \uplus ik$. Furthermore, we denote (message lists handled as) general knowledge bases simply by kb, kb' and the like.

10.1 Offline Computations

Indistinguishability properties allow us to express the security of some *hidden* information (within a protocol trace) in situations where partial knowledge (despite the observable messages) about this information is (supposed to be) public. They mean that a DY attacker is not able to learn the hidden information through the use of the additional knowledge and the observable messages.

In case of resistance against offline password testing, the protocol is required to hide *every password π used by any honest participant ag*. Here, it is assumed that the used passwords are chosen from a *relatively* small set of values. The property aims at the protection of the passwords in situations where the DY attacker *knows the set of the password candidates*. Offline testing in such a situation consists in trying the password candidates using the observable messages $\overline{ik}$. It fails if trying a *genuine* password π and a *false* password candidate π' (in combination with $\overline{ik}$) yields the same result. That is, the knowledge bases $kb = \pi.\overline{ik}$ and $kb' = \pi'.\overline{ik}$ are indistinguishable.

For anonymous authentication, the protocol is required to hide that *some participant ag has been successfully authenticated*. This should hold, although the DY attacker trivially *knows both possible outcomes of authentication attempts*. For that purpose, the protocol messages that are generated after the authentication check may not allow for an attacker to learn the result. This means, we need to contrast observable messages $kb = \overline{ik}$ with a succeeded authentication of ag and the associated knowledge base kb' where the same authentication attempt is enforced to fail. We have anonymity if kb and kb' are indistinguishable.

The privacy property in voting protocols means that the protocol is required to hide that *some participant ag has committed to a vote v*. This holds, even if the DY attacker knows *the voting result*. This property makes sense, only if at least two voters ag and ag' has participated and they have committed to different votes v and v'. Here, we contrast observable messages $kb = \overline{ik}$ (with the genuine votes) and the associated knowledge base kb' where the votes of ag and ag' are switched. We have the privacy if kb and kb' are indistinguishable.

In all three properties, the attacker tries to distinguish the considered knowledge bases kb and kb' through offline computations. Actually "distinction" means in this context rather *classification*, as the actual goal is to know that one of the knowledge bases (*either kb or kb'*) contains the genuine information. Thus, the offline computations serve to decide some predicate P, which holds *either* for kb *or* for kb'. The predicate $P(\overline{ms})$ in the resistance property expresses that "the first item in $\overline{ms}$ is a password used in the rest of $\overline{ms}$ by some participant ag". In case of anonymous authentication, $P(\overline{ms})$ expresses that "the authentication attempt in $\overline{ms}$ of some participant ag succeeded". Considering the privacy of voting, $P(\overline{ms})$ expresses that "the participants ag and ag' committed in $\overline{ms}$ to the different votes v and v', respectively".

In the rest of this section, we motivate our restriction of the relevant offline computations to the application of the available function symbols in Op (cp. Def. 13) and to equality checks.

Assume that there is a program $prog_P$ that allows the attacker to decide $P(\overline{ms})$. Recall that we are interested only in deterministic programs that do not make use of any precomputed constant, as these would be called by a DY attacker. We thus expect the program $prog_P$ to return upon inputs $\overline{ms}$, "yes" if $P(\overline{ms})$ holds, and "no" otherwise.

In the initial state of $prog_P$, the only alternative to proceed the computation is to call a first subroutine $proc_1$ and compute $m_1 = proc_1(\overline{ms})$. Since no precomputed constant is used in $prog_P$, the next relevant step is also the call of a second subroutine $proc_2$ to compute a second value m_2. Here, we can use $\overline{ms}$ and/or m_1 as arguments for $proc_2$. Knowing that m_1 can be recomputed out of $\overline{ms}$, we restrict the arguments of $proc_2$ (and similarly of all subroutines called in successor states) to the input $\overline{ms}$ of $prog_P$. That is, we have $m_2 = proc_2(\overline{ms})$.

After the computation of m_1 and m_2, there are principally two relevant alternatives on how to continue the program: (i) compute the next value $m_3 = proc_3(\overline{ms})$ independent of the previous results (ii) or carry out an equality check between m_1 and m_2 to decide what to do next. The equality check permits us then either to output a result or to choose the next action between computing the next value and carrying out the next equality check.

Basically, any state of $prog_P$ after the third computation step includes values $m_1 = proc_1(\overline{ms}), \ldots, m_n = proc_n(\overline{ms})$ and the results of 0 up to $(n-1)!$ equality checks between m_i and m_j with $i \neq j$. In the final states where "yes" or "no" is returned, we may assume that the number of equality checks differs from 0.

Our analysis shows that the $prog_P$ does not terminate or returns the same output upon inputs kb and kb', when all pairs $proc_i$ and $proc_j$ of the possible subroutines satisfy

$$proc_i(kb) = proc_j(kb) \Leftrightarrow proc_i(kb') = proc_j(kb').$$

Recall that the distinction between kb and kb' should be carried out through offline computations by a DY attacker. This means, the program $prog_P$ and the possible subroutines $proc_i$ are confined through the corresponding computation capabilities. We may thus restrict the subroutines to arbitrary algorithms that apply available function symbols from Op to *selected* items of the given knowledge base and to intermediate results of these function applications. We call these algorithms *generic* DY *derivations*.

For kb and kb' to be indistinguishable we need to exclude any program for the computation of P that does not satisfy the above equivalence. For that purpose, we require that the equivalence holds for all generic DY *derivations* $proc_i$ and $proc_j$. In the next section, we describe how this can be proved.

10.2 Proof Technique

To verify that two *different* knowledge bases kb and kb' are indistinguishable, we need to prove that the above equivalence holds for all pairs $proc_i$ and $proc_j$ of generic DY derivations. Because of the difference between kb and kb', the equality $proc_l(kb) = proc_l(kb')$ does *not* hold for all generic DY derivations. For that reason, we prove the required equivalence through a tailored simulation relation $\rightsquigarrow$ from $DY(kb)$ to $DY(kb')$, which *maps* $proc_l(kb)$ to $proc_l(kb')$ for all generic DY derivations $proc_l$. This relation is bijective and provides us with two functions $lr^{\rightsquigarrow} : DY(kb) \to DY(kb')$ and (the inverse function) $rl^{\rightsquigarrow} : DY(kb') \to DY(kb)$ that fulfill

1. $lr^{\rightsquigarrow}(proc_l(kb)) = proc_l(kb')$ and
2. $rl^{\rightsquigarrow}(proc_l(kb')) = proc_l(kb)$

for all generic DY derivations $proc_l$. This permits us to prove the above equivalence as follows:

$\Rightarrow$: Assume $proc_i(kb) = proc_j(kb)$, combining this equality and property (1) allows us to obtain $proc_i(kb') = lr^{\rightsquigarrow}(proc_i(kb)) = lr^{\rightsquigarrow}(proc_j(kb)) = proc_j(kb')$.

$\Leftarrow$: Assume $proc_i(kb') = proc_j(kb')$, combining this equality and property (2) allows us to obtain similarly $proc_i(kb) = rl^{\rightsquigarrow}(proc_i(kb')) = rl^{\rightsquigarrow}(proc_j(kb')) = proc_j(kb)$.

Although the appropriate relation $\rightsquigarrow$ is specific both to the protocol and to the considered indistinguishability property, we managed to provide a common schema for its definition with the help of a recursive function (using a finite set of message pairs as a basis relation, see Chap. 11). Furthermore, we systematically prove that the defined relation $\rightsquigarrow$

is bijective and that the induced functions $lr^{\rightsquigarrow}$ and $rl^{\rightsquigarrow}$ satisfy the above properties (1) and (2). In the latter proof task, we cover all possible generic DY derivations $proc_l$ by induction (on the indexes l). For that purpose, we come up with an appropriate enumeration of the generic DY derivations. Before we describe the corresponding definitions in Sec. 10.4, we introduce the underlying principles in the following section.

10.3 Generic DY Derivations

Properties (1) and (2) of the functions $lr^{\rightsquigarrow} : DY(kb) \to DY(kb')$ and $rl^{\rightsquigarrow} : DY(kb') \to DY(kb)$ are not just about arbitrary elements in $DY(kb)$ and $DY(kb')$ but also about the way how they are derived using kb and kb', respectively. They mean that the results of the l-th generic DY derivations using kb are mapped (through $lr^{\rightsquigarrow}$ and $rl^{\rightsquigarrow}$) to their results using kb'. For their proof by induction we come up with an enumeration of the generic DY derivations including their results relative to an arbitrary finite list $\overline{ms}$ of messages. This corresponds to an *extensional* definition of the DY knowledge $DY(ms)$, which enriches derivable messages with corresponding derivation trees.

We use the function $\overline{DY} : MsgList, Nat \to Msg$ to denote the result of the i-th generic DY derivation using an arbitrary message list $\overline{ms}$, i.e. $proc_i(\overline{ms})$, by $\overline{DY}(\overline{ms}, i)$.

Starting from a list representation $\overline{ms}$ of an arbitrary finite message set ms the sets $DYl_{Op}(ms, i)$ are sequentially generated by an arbitrary but fixed strategy. See Fig. 10.1. This strategy induces a *derivation tree* for each element (index) of the enumeration. Typical examples can be found in Fig. 10.2 and 10.3.

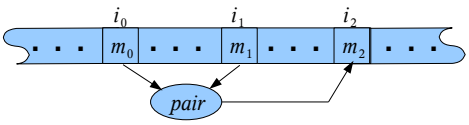

Figure 10.1: Enumeration of $DY(ms)$ by $\overline{DY}$

The second argument of $\overline{DY}$ is the index in the enumeration. This means that $\overline{DY}(\overline{ms}, i)$ is the message at position (index) i in the given enumeration.

The additional function $orig : MsgList, Nat \to NatList$ determines the tree structure on the enumeration in terms of indexes. It uses a fixed encoding of the available function symbols ($f \in Op$) as natural numbers. These symbols determine the semantics of derivation steps given in the definition of $orig$.

$orig(\overline{ms}, i)$ provides us with information about how the message at position i has been derived (in the derivation tree). For example in Fig. 10.3 $orig(\overline{ms}, i_3) = (2, n_{pair}, i_4, i_5)$, where $\overline{ms}$ is a fixed enumeration of the elements in ms, 2 is the arity of "$pair$", n_{pair} is a representative of this function symbol, and i_4, i_5 are the indexes of the ancestor messages of the message at i_3.

In general, there are two kinds of (alternating) nodes in such derivation trees: They are either messages that are labeled with a unique identifier (like $i_1 : enc(c_0, c_1)$ in Fig. 10.2) or a grey circle labeled with function names taken from the message algebra (like dec in Fig. 10.2). Root and leaf nodes are (labeled) messages and the function symbols in the gray circles tell us which function application led to the result message given the ancestor messages as arguments.

For instance, Fig. 10.2 shows a successful and a failing decryption step. Both correspond to the same function application, but with different arguments (messages in the ancestor

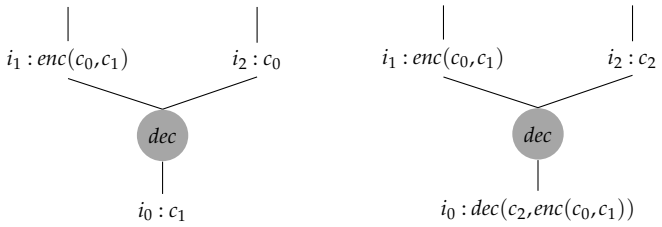

Figure 10.2: Successful versus failed decryption step

nodes). The derivation step on the left-hand side allows us to *extract* the crypt-part of the first argument, whereas the derivation step on the right-hand side corresponds to the *constructor-type* application of "*dec*" yielding to a *dec*-object (at index i_0).

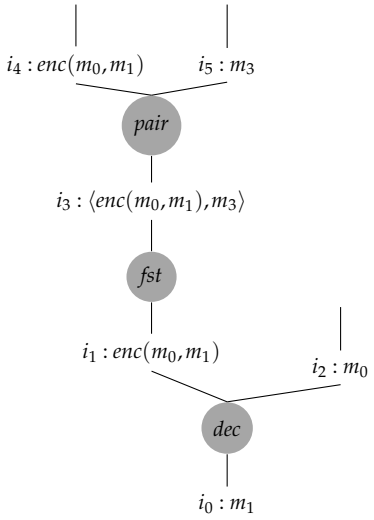

Figure 10.3: A reversing derivation step

Obviously, a message composition (by a constructor-type operation) may reverse a substructure extraction (for instance, by a decrypt-type operation) and vice versa, as illustrated in Fig. 10.3. We thus get trees that are different although *not essentially different*.

The functions $\overline{DY}$ and *orig* are defined non-constructively, embedding arbitrary enumerations that respect the above requirements about the numbers and the order of the derivation steps: Except of the starting enumeration defined by the used message list $\overline{ms}$ (DY level 0), the enumeration of the subsequent messages (in DY levels 1, 2, a.s.f. resulting by effective derivation steps) is formalized by axioms on $\overline{DY}$ and *orig* that impose the inherent order of the DY levels. The obtained formalization guarantees the soundness and the completeness of the enumeration. That is, every result of $\overline{DY}$ belongs to the intruder's knowledge, and every intruder's message occurs in the enumeration.

10.4 Formalization of Generic DY Derivations

We want to enumerate the elements of $DY(ms)$ in an arbitrary order and fix at the same time how every message in the sequence is generated from afore given messages. For this

we use the function $\overline{DY}$ which maps a message list $\overline{ms}$ and a natural number i to the i-th message in the enumeration.

The enumeration by the function $\overline{DY}$ is not exactly defined, e.g., constructively by a (deterministic) algorithm. It is any message sequence that lists the elements of the sets $DYl_{Op}(ms,0), DYl_{Op}(ms,1), DYl_{Op}(ms,2), \ldots$ from definition 13 in succession. If m_2 from $DYl_{Op}(ms,lv+1)$ is generated, for instance, by the application of "*pair*" to m_0 and m_1 from $DYl_{Op}(ms,lv)$, then m_2 is listed *after* m_0 and m_1, as it is shown in Fig. 10.1. This requirement is expressed with the help of a function LV that maps the given message list $\overline{ms}$ and a natural number lv to a natural number, i.e., the limit for the enumeration of $DYl_{Op}(ms,lv)$.

By $\overline{DY}$ we obtain only the enumeration without the information on how the messages are derived. For this purpose we want to express how the i-th message is generated, i.e. which $f \in Op$ is applied and what are the (indexes of the) used arguments. The application of a function symbol $f \in (Op \cap \Sigma\langle n\rangle)$ to messages with indexes $i_0,\ldots,i_{n-1}$ can be represented by the list $(f,i_0,\ldots,i_{n-1})$ (see Fig. 10.1, 10.2 and 10.3). For technical reasons, we use instead of the function symbol f the arity n followed by a number n_f that identifies f in an enumeration of $Op \cap \Sigma\langle n\rangle$. Then the derivation step by the application of f is represented by $(n,n_f,i_0,\ldots,i_{n-1})$. For every message $\overline{DY}(\overline{ms},i)$ in the enumeration we have a list of natural numbers that represents the last derivation step yielding this message. In correlation with $\overline{DY}$ we use the function *orig* which maps a message list $\overline{ms}$ and a natural number i to the list of natural numbers that represent the last derivation step of $\overline{DY}(\overline{ms},i)$.

In order to enumerate all the messages in $DY(ms)$ by $\overline{DY}$, we have to cover all possible derivations by *orig*. We start with the enumeration of $DYl_{Op}(ms,0)$.

Axiom$^{\mathbf{VSE}}$ 101 ($LV,\overline{DY},orig$, Level 0):
Let $len(l)$ denote the length of a list l and $sel(i,l)$ the $i+1$-th element of l, if any. Then we require the following properties on the functions LV, $\overline{DY}$ and *orig*, for arbitrary message lists $\overline{ms}$ and arbitrary natural numbers i:

$$LV(\overline{ms},0) = len(\overline{ms})$$

$$i < LV(\overline{ms},0) \Rightarrow (\overline{DY}(\overline{ms},i) = sel(i,\overline{ms}) \wedge orig(\overline{ms},i) = \epsilon)$$

The enumeration of $DYl_{Op}(ms,0) = ms$ by $\overline{DY}$ is given by the list representation $\overline{ms}$. Since the elements of ms are obtained without derivations, *orig* maps the indexes of these elements to the empty list ϵ.

On the other hand, for the elements of $DY(ms) \setminus ms$ we require *orig* to map their indexes to non-empty lists of natural numbers. These lists encode all possible derivations that result in the elements of $DYl_{Op}(ms,1) \setminus ms$, then of $DYl_{Op}(ms,2) \setminus DYl_{Op}(ms,1)$ and so forth.

Axiom$^{\mathbf{VSE}}$ 102 (*orig*, Next Levels):
Let $l_1\#l_2$ denote the concatenation of two lists l_1 and l_2 and let $\mathbb{N}^n = \overbrace{\mathbb{N} \times \ldots \times \mathbb{N}}^{n \text{ times}}$. Let every function symbol $f \in Op$ be associated with a pair of natural numbers (n,n_f) where n is the arity of f and n_f is the position of f in $\overline{(Op \cap \Sigma\langle n\rangle)}$, a fixed list representation of $(Op \cap \Sigma\langle n\rangle)$. Then, for all these pairs (n,n_f), we require the following properties on the functions LV and *orig*, for arbitrary message lists $\overline{ms}$:

$$\forall il \in \mathbb{N}^n : ((\forall j \in il : j < LV(\overline{ms},0)) \Rightarrow (\exists i < LV(\overline{ms},1) : orig(\overline{ms},i) = (n,n_f)\#il))$$

$$\forall lv \in \mathbb{N}, il \in \mathbb{N}^n : (((\forall j \in il : j < LV(\overline{ms}, lv+1)) \wedge (\exists j \in il : LV(\overline{ms}, lv) \leq j)) \\ \Rightarrow (\exists i < LV(\overline{ms}, lv+2) : orig(\overline{ms}, i) = (n, n_f)\#il))$$

The next enumerated messages in the DY knowledge are obtained by the application of functions $f \in Op$ to messages resulting from previous derivations. By *orig* we first associate every derivation step by a function f using arguments from ms with an index on level 1. Every subsequent derivation step, i.e. the application of a function f using arguments from level $lv+1$ is associated by *orig* with an index on level $lv+2$, when some of the used arguments do not already occur on level lv.

So far it is guaranteed by *orig* that all derivation steps (yielding level $lv+1$) are *completely* associated with indexes (smaller than $LV(\overline{ms}, lv+1)$). It remains to associate these derivation steps, represented by $orig(\overline{ms}, i)$, with the corresponding results, given by $\overline{DY}(\overline{ms}, i)$.

Axiom$^{\mathbf{VSE}}$ 103 ($\overline{DY}$ and *orig*):
We require the following property on the functions $\overline{DY}$, LV and $orig$, for arbitrary message lists $\overline{ms}$, and arbitrary numbers $lv, i \in \mathbb{N}$:

$$\begin{aligned} &LV(\overline{ms}, lv) \leq i < LV(\overline{ms}, lv+1) \Rightarrow \\ &\exists n > 0, f \in Op \cap \Sigma\langle n \rangle, n_f, i_0, \ldots, i_{n-1} \in \mathbb{N} : \\ &\quad (orig(\overline{ms}, i) = (n, n_f, i_0, \ldots, i_{n-1}) \wedge \\ &\quad\ (\forall j \in \{i_0, \ldots, i_{n-1}\} : j < LV(\overline{ms}, lv)) \wedge \\ &\quad\ sel(n_f, \overline{Op \cap \Sigma\langle n \rangle}) = f \wedge \\ &\quad\ \overline{DY}(\overline{ms}, i) = f(\overline{DY}(\overline{ms}, i_0), \ldots, \overline{DY}(\overline{ms}, i_{n-1}))) \end{aligned}$$

The enumeration of level 0, given by $\overline{ms}$ through axiom 101, is extended by $\overline{DY}$ according to the derivation steps that are provided by *orig*. The elements on level $lv+1$ that are not already present on level lv are enumerated as given by the derivation information encoded by *orig*.

The above axioms are specified in VSE with minor technical differences. For instance, we use $ap(n, n_f, (\overline{DY}(\overline{ms}, i_0), \ldots, \overline{DY}(\overline{ms}, i_{n-1})))$ instead of $f(\overline{DY}(\overline{ms}, i_0), \ldots, \overline{DY}(\overline{ms}, i_{n-1}))$ in axiom 103. The function $ap : Nat, Nat, MsgList \to Msg_\perp$ determines the function symbol $f \in Op$ from n and n_f and applies this to the given message list. Here, the type $Msg_\perp = Msg \cup \{\perp\}$ extends Msg with $\perp$ for the undefined results of ap. Such an undefined result is obtained, for instance, when the length of the given message list differs from the first argument. The definition of ap handles all the undefined, as well as the defined cases. The defined cases are given by the association of the function symbols $f \in (Op \cap \Sigma\langle n \rangle)$ to n and n_f.

Our formalization of $\overline{DY}$ allows us to obtain the following property on $\overline{DY}$ and DY:

Theorem$^{\mathbf{VSE}}$ 104 ($\overline{DY}$ and DY):
For all finite message sets ms and all list representations $\overline{ms}$ of ms, we have

$$DY(ms) = \{\overline{DY}(\overline{ms}, i) \mid i \in \mathbb{N}\}.$$

This theorem is immediately proven using the lemma

$$\begin{aligned} &DYl_{Op}(ms, n) \subseteq \{\overline{DY}(\overline{ms}, i) \mid i \in \{0, \ldots, LV(\overline{ms}, n) - 1\}\} \\ &\wedge \{\overline{DY}(\overline{ms}, i) \mid i \in \{0, \ldots, LV(\overline{ms}, n) - 1\}\} \subseteq DYl_{Op}(ms, n). \end{aligned}$$

The lemma is proved by induction on n.

This formalization of the generic DY derivations is carried out in VSE for the PACE algebra. The corresponding VSE-Theory `tenumdy` is shown in Fig. 7.2, together with the VSE-theory `tindy`, where *DY* and *DYl* are defined in a slightly but equivalent way as in Def. 13. The union of both VSE-theories provides the VSE-theory `tenumdy_dy` shown in Fig. 7.2, which defines the local context for theorem 104. This theorem is included in the VSE-theory `tenumdy_thms`, shown in Fig. 7.3 with a satisfies-link to the mentioned VSE-theory `tenumdy_dy`.

The formalization of the generic DY derivations in all these mentioned VSE-theories can be straightforwardly adapted to any message algebra.

Chapter 11

Proving Indistinguishability Properties (in PACE)

In this chapter we describe our proof technique for indistinguishability properties. Using generic DY derivations defined in Sec. 10.4, the indistinguishability of knowledge bases kb, kb' is formalized by

$$\forall i, j : \overline{DY}(kb, i) = \overline{DY}(kb, j) \Leftrightarrow \overline{DY}(kb', i) = \overline{DY}(kb', j). \tag{11.1}$$

This property is simply shown according to the proof in Sec. 10.2, if we provide a bijective (simulation) relation $\rightsquigarrow$ that satisfies

$$\forall l : \overline{DY}(kb, l) \rightsquigarrow \overline{DY}(kb', l). \tag{11.2}$$

Since the knowledge bases kb, kb' vary with the considered protocol traces, the definition of $\rightsquigarrow$ and the proof of its properties are specific to the protocol. Nevertheless, our proof technique described in this chapter *permits to handle the majority of the proof work only once for each message algebra*: We prove the existence of the appropriate simulation relations by providing basis relations in form of finite sets xy of message pairs. They are extended uniformly through composition to infinite relations $\rightsquigarrow$. The needed properties of $\rightsquigarrow$ follow by *a generic central theorem* that fixes corresponding necessary and sufficient conditions on xy, $DY(kb)$ and $DY(kb')$. This way, indistinguishability properties are reduced to *protocol-independent proof obligations* that are verified using typical trace properties (regularity properties) of the protocol.

Before we describe the proof of the central theorem, we give more details to the introduced proof method using the resistance of PACE against password testing as a running example.

11.1 The Building Blocks of Indistinguishability Proofs

In indistinguishability properties, the knowledge bases kb, kb' for which property (11.1) must be shown are fixed relative to protocol traces and according to the corresponding kind of security objectives. For instance, resistance against offline password testing is defined for all protocol traces tr, passwords π, π' and knowledge bases[1] kb, kb' that satisfy the following assumptions:

1. π is confidential, i.e. $\pi \notin DY(spies(tr))$,
2. π' does not occur in tr, i.e. $\forall m \in spies(tr) : \neg uses(m, \pi')$,

[1] Knowledge bases are lists of messages, although they are used in certain contexts as finite sets.

3. kb and kb' are obtained by adding π and respectively π' to $spies(tr)$, i.e. $kb = \pi.spies(tr)$ and $kb' = \pi'.spies(tr)$,

4. and tr includes a *regular* occurrence of π.

The instantiation in case of PACE is as follows:

Property$^{\mathbf{VSE}}$ 105 (Resistance against Offline Password Guessing; PACE):

$$
\begin{aligned}
&(tr \in \text{PACE} \wedge \pi \notin DY(spies(tr)) \wedge (\forall m \in spies(tr) : \neg uses(m, \pi')) \wedge \\
&\ ((\exists nc : enc(\pi, nc) \in spies(tr)) \vee \\
&\ \ (\exists g, m_1, m_2, nc : obj^{dec}(dec(\pi, m_1), \pi, m_1) \wedge \\
&\ \ \ dh(gen(dh(g, dec(\pi, m_1)), m_2), nc) \in spies(tr)))) \\
&\Rightarrow \\
&\forall i, j : \overline{DY}(\pi.\overline{spies(tr)}, i) = \overline{DY}(\pi.\overline{spies(tr)}, j) \Leftrightarrow \\
&\qquad \overline{DY}(\pi'.\overline{spies(tr)}, i) = \overline{DY}(\pi'.\overline{spies(tr)}, j)
\end{aligned}
$$

Except of the fourth assumption, which is specified using the protocol messages with occurrences of the protected password π, the remaining part (assumptions 1–3 and the conclusion) are property-specific. The formalization of resistance against offline password guessing of other protocols requires just to adapt assumption 4. See property 138 of TC-AMP in Sec. 14.1.

In case of PACE, we formalize the use of a protected password π in some protocol run through the occurrence in a first or in a fourth message. The latter situation covers the runs where both participants act as terminals (cp. the authentication property 8.4).

Indistinguishability property 105 illustrates how the knowledge bases kb, kb' are fixed by relevant protocol traces tr and by a tuple $\overline{x}$ of parameters that are specific to the corresponding kind of security objectives ($\overline{x} = (\pi, \pi')$ in case of resistance against offline password guessing). In our discussion, we abbreviate the conjunction of assumptions that fix kb, kb' by $\Omega(tr, \overline{x}, kb, kb')$.

Basically, such a conjunction of assumptions $\Omega(tr, \overline{x}, kb, kb')$ exists in all kinds of indistinguishability properties. In contrast to static equivalence properties, e.g., resistance against offline guessing, where kb and kb' are given by tr and $\overline{x}$, trace equivalence properties require to provide (as part of the proof), based on tr and $\overline{x}$, an equivalent protocol trace tr' used as a source for kb' like tr for kb. In both kinds of indistinguishability properties, the main proof work consists in showing the proof goal in (11.1) for all knowledge bases kb and kb' in the proof situations given by the assumptions in $\Omega(tr, \overline{x}, kb, kb')$.

Referring to the proof of indistinguishability properties with the help of a bijective (simulation) relation $\rightsquigarrow$ satisfying property (11.2), we need (i) to provide such a relation for each pair of knowledge bases kb, kb' (see Sec. 11.1.1) and (ii) to prove property (11.2) for this relation (see Sec. 11.1.2).

11.1.1 Defining $\rightsquigarrow$ Relations

The proof of indistinguishability using simulation relations requires to ensure for all protocol traces tr, corresponding parameters $\overline{x}$ and knowledge bases kb, kb' fulfilling $\Omega(tr, \overline{x}, kb, kb')$ *the existence* of a bijective relation $\rightsquigarrow$ that satisfies property (11.2). This means, we are dealing with a (second order) conjecture where the relation $\rightsquigarrow$ is existentially quantified. Instead of using an existentially quantified variable for the relation $\rightsquigarrow$,

we use that for a finite set xy of message pairs, employed as a basis relation to define the relation $\rightsquigarrow$ as described in the following.

11.1.1.1 Use of the $\widetilde{rec}$ Function

For given finite sets xy of message pairs, we define the relations $\overset{xy}{\rightsquigarrow}$ (abbreviated by $\rightsquigarrow$) with the help of a (recursive) function $\widetilde{rec}$:

$$m \overset{xy}{\rightsquigarrow} m' :\Leftrightarrow m' \in \widetilde{rec}\ (xy, m)$$

The range of $\widetilde{rec}$ consists of finite sets of messages, permitting to define relations $\overset{xy}{\rightsquigarrow}$ that extend given basis relations in form of finite sets xy of message pairs. The basis relation xy is used in the definition of $\widetilde{rec}$ like a look-up table: If m belongs to the domain of xy, $\widetilde{rec}$ returns the set of all messages associated to m by xy. Otherwise, m is expected to be a composed message, i.e. some f-object. In this case, each possible derivation of m by *composition*, i.e. by a constructor-type operation or by a synthesis operation, provides an alternative tuple of direct substructures or synth-parts of m, which we call *the (composition) sub-messages* of m. For each alternative sub-messages $m_0, \ldots, m_{n-1}$, $\widetilde{rec}$ identifies *recursively* the *Cartesian product* of the sets $\widetilde{rec}\ (xy, m_0), \ldots, \widetilde{rec}\ (xy, m_{n-1})$ and then uses the resulting tuples *if any* to derive messages *applying the same function symbol* like in the derivation of m from $m_0, \ldots, m_{n-1}$. The message sets resulting from *all* alternative sub-messages are finally combined by *union* to define (the result of) $\widetilde{rec}\ (xy, m)$ (for an f-object m). Note that the application of function symbols in the recursive case is not restricted to constructor-type and respectively synthesis operations.

In the PACE algebra, we obtain the following definition of $\widetilde{rec}$.

Definition$^{\mathbf{VSE}}$ 106 ($\widetilde{rec}$; PACE):
Let xy be a finite set of message pairs. Then, we have for two arbitrary messages m and m':

$$\begin{aligned}
&m' \in \widetilde{rec}\ (xy, m) \Leftrightarrow \\
&((m, m') \in xy \vee \\
&\quad ((\forall z : (m, z) \notin xy) \wedge \\
&\qquad ((\exists m_0, m_0' : m_0' \in \widetilde{rec}\ (xy, m_0) \wedge \\
&((obj^{fst}(m, m_0) \wedge m' = fst(m_0')) \vee (obj^{snd}(m, m_0) \wedge m' = snd(m_0')))) \vee \\
&\qquad\quad (\exists m_0, m_1, m_0', m_1' : m_0' \in \widetilde{rec}\ (xy, m_0) \wedge m_1' \in \widetilde{rec}\ (xy, m_1) \wedge \\
&((obj^{pair}(m, m_0, m_1) \wedge m' = pair(m_0', m_1')) \vee (obj^{enc}(m, m_0, m_1) \wedge m' = enc(m_0', m_1')) \vee \\
&(obj^{dec}(m, m_0, m_1) \wedge m' = dec(m_0', m_1')) \vee (obj^{dh}(m, m_0, m_1) \wedge m' = dh(m_0', m_1')) \vee \\
&(obj^{gen}(m, m_0, m_1) \wedge m' = gen(m_0', m_1')) \vee (obj^{mac}(m, m_0, m_1) \wedge m' = mac(m_0', m_1'))))))).
\end{aligned}$$

Note that the recursive case of $\widetilde{rec}$ considers for dh-objects m implicitly *several* alternatives of sub-messages, as they are composed by constructor-type applications of the *permutative* function symbol dh. Contrarily to the other composed messages, the handling of dh-objects will necessitate for that reason (a kind of an) additional induction (dimension) on the number of sub-message alternatives (see Sec. 11.3).

Having the function $\widetilde{rec}$, we describe in the following section how to define suitable sets xy of message pairs (as basis relations) to use $\overset{xy}{\rightsquigarrow}$ (defined above) as simulation relations. In

particular, the bijection property for a relation $\overset{xy}{\rightsquigarrow}$ is shown below by considering in parallel the (inverse) relation $\overset{xy}{\rightsquigarrow}$, which we define by

$$m \overset{xy}{\rightsquigarrow} m' :\Leftrightarrow m \in \overset{\rightsquigarrow}{rec}\,(rev(xy), m'), \text{where}$$

$m \overset{xy}{\rightsquigarrow} m'$ denotes $(m', m) \in \overset{xy}{\rightsquigarrow}$ and where $rev(xy)$ equals the set $\{(m_2, m_1) \mid (m_1, m_2) \in xy\}$.

We denote the bijection property of $\overset{xy}{\rightsquigarrow}$ with the help of the following abbreviations:

$$m \overset{xy}{\mapsto} m' :\Leftrightarrow \overset{\rightsquigarrow}{rec}\,(xy, m) = \{m'\}$$
$$m \overset{xy}{\leftarrow} m' :\Leftrightarrow \overset{\rightsquigarrow}{rec}\,(rev(xy), m') = \{m\}$$
$$m \overset{xy}{\leftrightarrow} m' :\Leftrightarrow (m \overset{xy}{\mapsto} m' \wedge m \overset{xy}{\leftarrow} m')$$

11.1.1.2 Defining the Basis Relations xy

A crucial part of the proof work consists in identifying the contents of the sets xy that are used as basis relations. This is done taking into consideration the items in kb, kb', which are themselves defined relative to ik (the immediately observable messages of the protocol) and the property-specific parameters $\overline{x}$ (π, π' in resistance against offline password guessing).

The contents of xy must be carefully identified, considering two issues:

1. In order for $\overset{xy}{\rightsquigarrow}$ to be *total*, i.e. to have $DY(kb)$ as domain and $DY(kb')$ as codomain, the basis relations xy must cover all derivable atomic messages (Ψ^a) and all derivable composed messages that *cannot be derived by composition* because of missed submessages (Ψ^c):

$$\Psi^a: \quad ((m \in At \wedge m \in DY(kb)) \Rightarrow (\exists m' : (m, m') \in xy)) \wedge$$
$$((m \in At \wedge m \in DY(kb')) \Rightarrow (\exists m' : (m', m) \in xy))$$

$$\begin{aligned}\Psi^c: \quad & ((m \notin At \wedge m \in DY(kb) \wedge \neg isObj^{pair}(m) \\ & \quad \wedge ((obj^{fst}(m, m_0) \vee obj^{snd}(m, m_0)) \Rightarrow m_0 \notin DY(kb)) \\ & \quad \wedge ((obj^{enc}(m, m_0, m_1) \vee obj^{dec}(m, m_0, m_1) \vee obj^{dh}(m, m_0, m_1) \vee \\ & \qquad obj^{gen}(m, m_0, m_1) \vee obj^{mac}(m, m_0, m_1)) \Rightarrow \neg(m_0, m_1 \in DY(kb)))) \\ & \quad \Rightarrow (\exists m' : (m, m') \in xy)) \wedge \\ & ((m \notin At \wedge m \in DY(kb') \wedge \neg isObj^{pair}(m) \\ & \quad \wedge ((obj^{fst}(m, m_0) \vee obj^{snd}(m, m_0)) \Rightarrow m_0 \notin DY(kb')) \\ & \quad \wedge ((obj^{enc}(m, m_0, m_1) \vee obj^{dec}(m, m_0, m_1) \vee obj^{dh}(m, m_0, m_1) \vee \\ & \qquad obj^{gen}(m, m_0, m_1) \vee obj^{mac}(m, m_0, m_1)) \Rightarrow \neg(m_0, m_1 \in DY(kb')))) \\ & \quad \Rightarrow (\exists m' : (m', m) \in xy))\end{aligned}$$

2. Besides Ψ^a and Ψ^c, we identify below further conditions on xy, $DY(kb)$ and $DY(kb')$ *necessary* to prove that $\overset{xy}{\rightsquigarrow}$ correspond to appropriate simulation relations (see for instance Ψ^2_{mac} in Sec. 11.1.2.1). All these conditions form the proof obligations, which shall be shown with the help of regularity properties of the protocol on the different types of the derivable messages and the protection of their substructures (see for instance the regularity property 118 in Sec. 12.2.4). To simplify matters, we use sets xy that include *as few pairs as possible*. This permits to get rid of many proof obligations (by revoking their assumptions) due to the absence of redundant pairs in the basis relations xy.

To reach both purposes (coverage and minimality), we define the basis relations xy as the *smallest* sets of message pairs that satisfy tailored inclusion rules. With these rules, we identify the pairs that shall be included in xy at the protocol steps and under certain conditions. For instance, the inclusion of the pair (π,π') in xy for the resistance proof of PACE is defined by a rule that requires the occurrence of a $enc(\pi,nc)$ or $dh(gen(dh(g,dec(\pi,m_1)),m_2),nc)$ in the observable messages ik, for some nc,g,m_1,m_2 (see rule Φ_1 in Sec. 12.1).

Note that certain inclusion rules make use of negative conditions to exclude pairs (m,m') where m and m' are derivable by composition from other items in $DY(kb)$ and respectively $DY(kb')$. For instance, the inclusion of pairs $(mac(m_1,m_2),mac(m_1,m_2))$ in xy for the resistance proof of PACE is defined by a rule that requires $mac(m_1,m_2) \in ik$ and the (negative) condition $m_1 \notin DY(ik)$ (see rule Φ_5 in Sec. 12.1).

Like the contents of kb and kb', the content of xy is also defined (by the inclusion rules) relative to the observable messages ik and the parameters $\overline{x}$ (π,π' in resistance against offline password guessing). So, we use $\Phi(xy,ik,\overline{x})$ to denote in our discussion the formula defining basis relations xy.

Technically, we use the fixed definition $\Phi(xy,ik,\overline{x})$ of basis relations xy in two lemmata, which provide us with a bijective (simulation) relation to use in the proof of indistinguishability as described in Sec. 10.2.

1. The first lemma $\Omega(tr,\overline{x},kb,kb') \Rightarrow \exists xy : \Phi(xy,ik,\overline{x})$, which we call *basis simulation relation lemma*, provides the sets xy for the definition of $\overset{xy}{\sim}$. It is verified by induction on protocol traces (see Sec. 11.1.1.3).

2. The second lemma $(\Omega(tr,\overline{x},kb,kb') \wedge \Phi(xy,ik,\overline{x})) \Rightarrow (\forall l : \overline{DY}(kb,l) \overset{xy}{\leftrightarrow} \overline{DY}(kb',l))$ ensures the required properties of $\overset{xy}{\sim}$. It is handled by the so-called *central indistinguishability theorem* (see Sec. 11.1.2) to obtain the proof obligations shown using (besides $\Omega(tr,\overline{x},kb,kb')$ and $\Phi(xy,ik,\overline{x})$) regularity properties verified by induction on protocol traces.

11.1.1.3 Basis Simulation Relation Lemma

In the basis simulation relation lemma, we provide for every considered pair of knowledge bases kb and kb' a set xy of message pairs that corresponds to the required basis relation as stated in the conjectured definition Φ. Afterwards, we show that the corresponding relations $\overset{xy}{\sim}$ are bijective and satisfy property (11.2).

Using a definition $\Phi(xy,spies(tr),\pi,\pi')$ for the basis relation sets in the resistance proof of PACE, we obtain the following basis simulation relation lemma:

Property$^{\text{VSE}}$ 107 (Basis Simulation Relation Lemma; PACE):

$$
\begin{aligned}
&(tr \in \text{PACE} \wedge \pi \notin DY(spies(tr)) \wedge (\forall m \in spies(tr) : \neg uses(m,\pi')) \wedge \\
&\quad ((\exists nc : enc(\pi,nc) \in spies(tr)) \vee \\
&\qquad (\exists g,m_1,m_2,nc : obj^{dec}(dec(\pi,m_1),\pi,m_1) \wedge \\
&\qquad\quad dh(gen(dh(g,dec(\pi,m_1)),m_2),nc) \in spies(tr)))) \\
&\Rightarrow (\exists xy : \Phi(xy,spies(tr),\pi,\pi'))
\end{aligned}
$$

The proof of this property is by induction on all PACE traces. The base case is trivial.

In the step case, the induction hypothesis provides xy_p that satisfies $\Phi(xy_p,ik_p,\pi,\pi')$. For the extended intruder knowledge $ik = ik_p \uplus ik_e$, we need to provide xy relative to xy_p,

ik_p and ik_e and show that $\Phi(xy, ik_p \uplus ik_e, \pi, \pi')$ holds. For that purpose, we analyze how the new items in ik_e contribute to the derivation of elements in $DY(kb) \setminus DY(kb_p)$ and $DY(kb') \setminus DY(kb'_p)$ (for $kb = kb_p \uplus ik_e$ and $kb' = kb'_p \uplus ik_e$). In most cases of the regular protocol steps, the new items permit the derivation of new atomic messages or new composed messages with protected sub-messages. Here, xy is obtained by extending xy_p with pairs for these new atomic or composed messages. For instance, in the sixth and seventh steps of PACE, we obtain $ik_e = \{mac(m_1, m_2)\}$ with $m_2 \in ik_p$ and distinguish the case where m_1 is protected, where xy is then set to $xy_p \cup \{(mac(m_1, m_2), mac(m_1, m_2))\}$ (see Sec. 12.3).

In few cases of the regular protocol steps, the new items permit the derivation of sub-messages of composed messages m_c occurring in pairs of xy_p, such that m_c becomes derivable by composition. For xy to be minimal, xy must be obtained from xy_p by removing the m_c-pairs and adding pairs for corresponding sub-messages and possibly for other new items (as above, in the simple extension case). For instance, in the oops case of PACE, the lost key in ik_e can be the MAC-key in some pair $(mac(m_1, m_2), mac(m_1, m_2))$ in xy_p, which necessitates to set xy to the result of replacing this pair in xy_p with (m_1, m_1) (see Sec. 12.3). Note that m_1 possesses protected sub-messages ((m_1, m_1) cannot be substituted with pairs for sub-messages of m_1) and that m_2 occurs in ik_p (no need to include (m_2, m_2) or pairs for sub-messages of m_2).

In the fake case, we need to set $xy = xy_p$ as the attacker should not be able to derive new *sensitive* messages, which can be used to violate the indistinguishability property. This is verified (foremost at this phase) with the help of the above mentioned regularity properties of the protocol. Whenever m in ik_e (and from $DY(ik_p)$) matches a message that fires an inclusion rule, the regularity properties shall imply that this specific message must be already considered when building xy_p. In case for instance $m = mac(m_1, m_2)$ with a protected m_1, the regularity property about derivable *mac*-objects implies $mac(m_1, m_2) \in ik_p$ and $m_1 \notin DY(ik_p)$, which yielded to the inclusion of $(mac(m_1, m_2), mac(m_1, m_2))$ in xy_p (see property 118 in Sec. 12.2.4).

In all (above mentioned) cases, property $\Phi(xy, ik_p \uplus ik_e, \pi, \pi')$ shall be simply shown from $\Phi(xy_p, ik_p, \pi, \pi')$ and the assumptions yielding to the corresponding definition of xy relative to xy_p (see Sec. 12.3).

11.1.2 The Central Indistinguishability Theorem

The central indistinguishability theorem is used to prove the required properties of the relations $\overset{xy}{\rightsquigarrow}$ for given basis relations xy (cp. step 2 in the proof plan for indistinguishability properties, described in Sec. 11.1.1.2). Its premise provides the necessary and sufficient conditions, which form the proof obligations that we show with the help of regularity properties of the protocol verified by trace induction (see Sec. 11.1.2.5).

The necessary and sufficient conditions can be distinguished as follows:

- Ψ^a and Ψ^c in Sec. 11.1.1.2 ensure that $\overset{xy}{\rightsquigarrow}$ is total.
- Two further *generic* conditions (Ψ^b and Γ) used to ensure that bijective basis relations $xy \subset DY(kb) \times DY(kb')$ are used and that $\overset{xy}{\rightsquigarrow}$ correctly maps the first DY-level kb to kb' (see Sec. 11.1.2.2).
- Additional *algebra-specific* conditions on xy ensure that derivations by (single) applications of function symbols do not violate indistinguishability (see Sec. 11.1.2.1).

11.1.2.1 Algebra-specific Conditions

In this section, we present the algebra-specific conditions for indistinguishability in the PACE algebra, emphasizing their different types to support adaptations to other message algebras.

The generation of the conditions follows two general principles:

- If the f-parts $m_0,\ldots,m_{n-1}$ of m that is mapped by xy (resp. $rev(xy)$) are public, then these f-parts must be mapped to $m'_0,\ldots,m'_{n-1}$ such that the image of m equals $f(m'_0,\ldots,m'_{n-1})$.
- When f has algebraic effects, additional conditions are used to capture the allowed effects and exclude the critical ones.

We start with the most simple conditions, where algebraic effects of f (if any) are not (explicitly) handled.

$$\Psi^1_{mac}: \quad ((mac(m_{x1},m_{x2}),m_y) \in xy \wedge m_{x1},m_{x2} \in DY(kb)) \Rightarrow (\exists m_{y1},m_{y2} \in DY(kb') : (m_{x1},m_{y1}),(m_{x2},m_{y2}) \in xy \wedge m_y = mac(m_{y1},m_{y2}))$$

$$\Psi^2_{mac}: \quad ((m_x,mac(m_{y1},m_{y2})) \in xy \wedge m_{y1},m_{y2} \in DY(kb')) \Rightarrow (\exists m_{x1},m_{x2} \in DY(kb) : (m_{x1},m_{y1}),(m_{x2},m_{y2}) \in xy \wedge m_x = mac(m_{x1},m_{x2}))$$

$$\Psi^1_{gen}: \quad ((gen(m_{x1},m_{x2}),m_y) \in xy \wedge m_{x1},m_{x2} \in DY(kb)) \Rightarrow (\exists m_{y1},m_{y2} \in DY(kb') : (m_{x1},m_{y1}),(m_{x2},m_{y2}) \in xy \wedge m_y = gen(m_{y1},m_{y2}))$$

$$\Psi^2_{gen}: \quad ((m_x,gen(m_{y1},m_{y2})) \in xy \wedge m_{y1},m_{y2} \in DY(kb')) \Rightarrow (\exists m_{x1},m_{x2} \in DY(kb) : (m_{x1},m_{y1}),(m_{x2},m_{y2}) \in xy \wedge m_x = gen(m_{x1},m_{x2}))$$

$$\Psi^1_{dh}: \quad ((dh(m_{x1},m_{x2}),m_y) \in xy \wedge m_{x1},m_{x2} \in DY(kb)) \Rightarrow (\exists m_{y1},m_{y2} \in DY(kb') : (m_{x1},m_{y1}),(m_{x2},m_{y2}) \in xy \wedge m_y = dh(m_{y1},m_{y2}))$$

$$\Psi^2_{dh}: \quad ((m_x,dh(m_{y1},m_{y2})) \in xy \wedge m_{y1},m_{y2} \in DY(kb')) \Rightarrow (\exists m_{x1},m_{x2} \in DY(kb) : (m_{x1},m_{y1}),(m_{x2},m_{y2}) \in xy \wedge m_x = dh(m_{x1},m_{x2}))$$

$$\Psi^1_{pair}: \quad (pair(m_{x1},m_{x2}),m_y) \in xy \Rightarrow (\exists m_{y1},m_{y2} \in DY(kb') : (m_{x1},m_{y1}),(m_{x2},m_{y2}) \in xy \wedge m_y = pair(m_{y1},m_{y2}))$$

$$\Psi^2_{pair}: \quad (m_x,pair(m_{y1},m_{y2})) \in xy \Rightarrow (\exists m_{x1},m_{x2} \in DY(kb) : (m_{x1},m_{y1}),(m_{x2},m_{y2}) \in xy \wedge m_x = pair(m_{x1},m_{x2}))$$

The following conditions integrate the allowed reversing effects of *enc* and *dec*.

$$\Psi^1_{enc}: \quad (obj^{enc}(m_x,m_{x1},m_{x2}) \wedge (m_x,m_y) \in xy \wedge m_{x1} \in DY(kb)) \Rightarrow (\exists m_{y1} \in DY(kb') : (m_{x1},m_{y1}),(m_{x2},dec(m_{y1},m_y)) \in xy)$$

$$\Psi^2_{enc}: \quad (obj^{enc}(m_y,m_{y1},m_{y2}) \wedge (m_x,m_y) \in xy \wedge m_{y1} \in DY(kb')) \Rightarrow (\exists m_{x1} \in DY(kb) : (m_{x1},m_{y1}),(dec(m_{x1},m_x),m_{y2}) \in xy)$$

$$\Psi^1_{dec}: \quad (obj^{dec}(m_x,m_{x1},m_{x2}) \wedge (m_x,m_y) \in xy \wedge m_{x1} \in DY(kb)) \Rightarrow (\exists m_{y1} \in DY(kb') : (m_{x1},m_{y1}),(m_{x2},enc(m_{y1},m_y)) \in xy)$$

$$\Psi^2_{dec}: \quad (obj^{dec}(m_y,m_{y1},m_{y2}) \wedge (m_x,m_y) \in xy \wedge m_{y1} \in DY(kb')) \Rightarrow \\ (\exists m_{x1} \in DY(kb) : (m_{x1},m_{y1}),(enc(m_{x1},m_x),m_{y2}) \in xy)$$

A composition by *enc* (resp. *dec*) on one side may have a decrypt-type effect on the other side, only if the left part in the composition is mapped to the crypt-key.

The following conditions integrate the excluded effects of fst and *dec* on *pair*-objects.

$$\Psi^1_{fst}: \quad (obj^{fst}(m_x,m_{x1}) \wedge (m_x,m_y) \in xy \wedge m_{x1} \in DY(kb)) \Rightarrow \\ (\exists m_{y1} \in DY(kb') : (m_{x1},m_{y1}) \in xy \wedge obj^{fst}(m_y,m_{y1}))$$

$$\Psi^2_{fst}: \quad (obj^{fst}(m_y,m_{y1}) \wedge (m_x,m_y) \in xy \wedge m_{y1} \in DY(kb')) \Rightarrow \\ (\exists m_{x1} \in DY(kb) : (m_{x1},m_{y1}) \in xy \wedge obj^{fst}(m_x,m_{x1}))$$

$$\Psi^1_{snd}: \quad (obj^{snd}(m_x,m_{x1}) \wedge (m_x,m_y) \in xy \wedge m_{x1} \in DY(kb)) \Rightarrow \\ (\exists m_{y1} \in DY(kb') : (m_{x1},m_{y1}) \in xy \wedge obj^{snd}(m_y,m_{y1}))$$

$$\Psi^2_{snd}: \quad (obj^{snd}(m_y,m_{y1}) \wedge (m_x,m_y) \in xy \wedge m_{y1} \in DY(kb')) \Rightarrow \\ (\exists m_{x1} \in DY(kb) : (m_{x1},m_{y1}) \in xy \wedge obj^{snd}(m_x,m_{x1}))$$

An application of fst (resp. *snd*) is not allowed to extract the select-part of a *pair*-object that is mapped to a fst-part (resp. *snd*-part). For instance, $obj^{fst}(m_y,m_{y1})$ in Ψ^1_{fst}, where the mapped message m_x is a fst-object, excludes that m_{y1}, mapped to the fst-part of m_x, is a *pair*-object.

The following conditions would not be identified without a thorough proof of the central indistinguishability theorem (see Sec. 11.2–11.4). They are specific to the algebraic effects of *dh*. Except that multiple right *dh*-parts can be switched, there are no other algebraic effects of *dh* and *dh*-objects. These features necessitate the use of $\Psi^{1,2}_{dh}$ and $\Psi^{2,2}_{dh}$ to exclude critical effects of *dh*, as illustrated by this example: The set xy can include $(dh(a,b),y_b)$ and $(dh(a,c),y_c)$ with $b,c \in DY(kb)$ and $a \notin DY(kb)$. Here, $dh(dh(a,b),c)$ is derivable by the application of *dh* using $(dh(a,b),c)$ as well as $(dh(a,c),b)$. For that reason, the same derivations on the other side, i.e. $dh(y_b,m_c)$ for m_c mapped to c and $dh(y_c,m_b)$ for m_b mapped to b, must provide the same result, i.e. $dh(y_b,m_c) = dh(y_c,m_b)$. This justifies the necessary condition $\Psi^{1,2}_{dh}$.

$$\Psi^{1,2}_{dh}: \quad ((\overline{dh}(m_z,ms_{x1}),m_{y1}),(\overline{dh}(m_z,ms_{x2}),m_{y2}) \in xy \wedge m_z \notin DY(kb) \wedge \\ ms_{x1} \neq ms_{x2} \wedge ms_{x1},ms_{x2} \subset DY(kb)) \Rightarrow \\ (\exists ms_{y1},ms_{y2} \subset DY(kb'),m_u \notin DY(kb') : \wp(ms_{x1},ms_{y1}),\wp(ms_{x2},ms_{y2}) \subseteq xy \wedge \\ m_{y1} = \overline{dh}(m_u,ms_{y1}) \wedge m_{y2} = \overline{dh}(m_u,ms_{y2}))$$

$$\Psi^{2,2}_{dh}: \quad ((m_{x1},\overline{dh}(m_z,ms_{y1})),(m_{x2},\overline{dh}(m_z,ms_{y2})) \in xy \wedge m_z \notin DY(kb') \wedge \\ ms_{y1} \neq ms_{y2} \wedge ms_{y1},ms_{y2} \subset DY(kb')) \Rightarrow \\ (\exists ms_{x1},ms_{x2} \subset DY(kb),m_u \notin DY(kb) : \wp(ms_{x1},ms_{y1}),\wp(ms_{x2},ms_{y2}) \subseteq xy \wedge \\ m_{x1} = \overline{dh}(m_u,ms_{x1}) \wedge m_{x2} = \overline{dh}(m_u,ms_{x2}))$$

In $\Psi^{1,2}_{dh}$, $\wp(ms_{x1},ms_{y1})$ denotes a finite set of pairs mapping each element in ms_{x1} to a unique element in ms_{y1}. According to $\Psi^{1,2}_{dh}$, if xy maps two *dh*-objects having a same protected left

dh-part m_z, their images must be also *dh*-objects with a same protected left *dh*-part m_u. These images must result by successive applications of *dh* to m_u and to the images of ms_{x1} and ms_{x2}, respectively.

11.1.2.2 Generic Conditions and Theorem

Besides Ψ^a and Ψ^c in Sec. 11.1.1.2, two further generic conditions are used.

First, Ψ^b ensures that a finite set xy of message pairs corresponds to a bijective relation between a subset of $DY(kb)$ and a subset of $DY(kb')$.

$$\begin{aligned}\Psi^b: \quad & (m,m') \in xy \Rightarrow \\ & (m \in DY(kb) \wedge m' \in DY(kb') \wedge \\ & ((m,m'') \in xy \Rightarrow m'' = m') \wedge ((m'',m') \in xy \Rightarrow m'' = m))\end{aligned}$$

In the following, we use $\Psi(xy,kb,kb')$ to denote the conjunction of Ψ^a, Ψ^c, Ψ^b and the algebra-specific conditions (in Sec. 11.1.2.1).

Besides $\Psi(xy,kb,kb')$, the following condition

$$\Gamma: \quad 0 \leq l < len(kb) \Rightarrow sel(l,kb) \overset{xy}{\rightsquigarrow} sel(l,kb')$$

ensures that $\overset{xy}{\rightsquigarrow}$ maps the first DY-level relative to kb to the first DY-level relative to kb', as required for the base case of the theorem's proof (see Sec. 11.4). Using the mentioned conditions, the central indistinguishability theorem is formalized by:

Theorem$^{\mathbf{VSE}}$ 108 ($\overline{DY}(kb,l) \overset{xy}{\leftrightarrow} \overline{DY}(kb',l)$):
Let xy be a finite set of message pairs and let kb,kb' be two equally-long lists of message items. Then, we have

$$(\Psi(xy,kb,kb') \wedge \Gamma(xy,kb,kb')) \Rightarrow (\forall l : \overline{DY}(kb,l) \overset{xy}{\leftrightarrow} \overline{DY}(kb',l)).$$

The theorem permits to prove that the relations $\overset{xy}{\rightsquigarrow}$ correspond to specific bijective functions. They map each generic derivation applied to kb to its result when applied to kb'. The proof of this property is by induction on the successive enumeration of the generic derivations, which is compatible with a natural order of the DY-levels as described in Sec. 10.3 and 10.4. We describe this proof in Sec. 11.4, where we make use of two lemmata (Structural Mapping Lemma introduced in Sec. 11.1.2.3 and Domain Restriction Lemma in Sec. 11.1.2.4).

11.1.2.3 Structural Mapping Lemma

According to the properties of $\overset{xy}{\rightsquigarrow}$ in theorem 108, the binary relation $\overset{xy}{\rightsquigarrow}$ corresponds to a (one-to-one) mapping between $\overline{DY}(kb,l)$ and $\overline{DY}(kb',l)$. For the proof of this mapping (in Sec. 11.4), we need an *appropriate reduction* of the mapping between composed messages in $DY(kb)$ and those in $DY(kb')$ *to* the mapping between their sub-messages. This is provided by the so-called *structural mapping lemma* introduced in this section. It does not only conjectures that $\overset{xy}{\rightsquigarrow}$ is a bijective relation between $DY(kb)$ and $DY(kb')$, but it also provides the so-called *structural condition* on the mapping of composed messages relative to the mappings of their sub-messages. Mapped messages m,m' for which no pair (m,m') exists in xy satisfy the structural condition abbreviated by $\Xi(m,m',xy,kb,kb')$.

In $\Xi(m,m',xy,kb,kb')$, we define how m' are obtained from messages $m'_0,\ldots,m'_{n-1}$ mapped by $\overset{xy}{\leftrightarrow}$ to composition sub-messages $m_0,\ldots,m_{n-1}$ of m. The definition of Ξ must be a complete case distinction, which covers all composition alternatives of f-objects m. It

includes canonical cases for f-objects that can be composed only by a unique constructor-type application of f using a definite tuple of f-parts. We abbreviate such a case with a predicate Ξ_f, defined according to the following schema:

$$\begin{aligned}&\Xi_f(m,m',xy,kb,kb') \Leftrightarrow\\ &(\exists m_0,\ldots,m_{n-1} \in DY(kb), m'_0,\ldots,m'_{n-1} \in DY(kb') : obj^f(m,m_0,\ldots,m_{n-1}) \wedge\\ &\quad m_0 \overset{xy}{\leftrightarrow} m'_0 \wedge \ldots \wedge m_{n-1} \overset{xy}{\leftrightarrow} m'_{n-1} \wedge obj^f(m',m'_0,\ldots,m'_{n-1}))\end{aligned}$$

For instance, the case for *enc*-objects is defined by

$$\begin{aligned}&\Xi_{enc}(m,m',xy,kb,kb') \Leftrightarrow\\ &(\exists m_0,m_1 \in DY(kb), m'_0,m'_1 \in DY(kb') : obj^{enc}(m,m_0,m_1) \wedge\\ &\quad m_0 \overset{xy}{\leftrightarrow} m'_0 \wedge m_1 \overset{xy}{\leftrightarrow} m'_1 \wedge obj^{enc}(m',m'_0,m'_1)).\end{aligned}$$

For f-objects m that do not fit in with the canonical cases, the definition of Ξ will include additional non-canonical predicates covering the (multiple) composition alternatives of m. In PACE, we use a predicate Ξ_{dh} for the case of *dh*-objects m (see Sec. 11.3), to specify derivations by composition through successive applications of *dh*. This supports the handling of multiple decomposition alternatives of *dh*-objects (in $\overset{\sim}{rec}$).

Using Ξ_{dh} and the canonical predicates $\Xi_{enc},\Xi_{dec},\Xi_{fst},\Xi_{snd},\Xi_{pair},\Xi_{mac},\Xi_{gen}$, we obtain the following structural mapping lemma:

Lemma$^{\mathbf{VSE}}$ 109 ($DY(kb) \overset{xy}{\leftrightarrow} DY(kb')$; PACE):

$$\begin{aligned}&(\Psi(xy,kb,kb') \wedge m \in DY(kb)) \Rightarrow\\ &(\exists m' \in DY(kb') : m \overset{xy}{\leftrightarrow} m' \wedge ((m,m') \in xy \vee\\ &\quad \Xi_{enc}(m,m',xy,kb,kb') \vee \Xi_{dec}(m,m',xy,kb,kb') \vee\\ &\quad \Xi_{fst}(m,m',xy,kb,kb') \vee \Xi_{snd}(m,m',xy,kb,kb') \vee\\ &\quad \Xi_{pair}(m,m',xy,kb,kb') \vee \Xi_{mac}(m,m',xy,kb,kb') \vee\\ &\quad \Xi_{gen}(m,m',xy,kb,kb') \vee \Xi_{dh}(m,m',xy,kb,kb')))\end{aligned}$$

The proof of this lemma (in Sec. 11.2) is by induction on the *(decomposition) structure* of the messages m in $DY(kb)$, i.e. on $|m|$. The handling of the non-canonical decomposition case by Ξ_{dh} is described in Sec. 11.3.

11.1.2.4 Domain Restriction Lemma

The central indistinguishability theorem and the structural mapping lemma rely on the property $\overset{xy}{\rightsquigarrow} \subset DY(kb) \times DY(kb')$, i.e. the domain of $\overset{xy}{\rightsquigarrow}$ is a subset of $DY(kb)$ and its codomain a subset of $DY(kb')$. This property permits to get rid of mapping alternatives by recursive calls of $\overset{\sim}{rec}$ where non-derivable sub-messages are used: If a sub-message required for a composition of $m \in DY(kb)$ is protected, i.e. does not belong to $DY(kb)$, all composition alternatives using this sub-message do not lead (in the recursive call of $\overset{\sim}{rec}$) to images of m in $DY(kb')$. The property $\overset{xy}{\rightsquigarrow} \subset DY(kb) \times DY(kb')$ is conjectured in lemma 110, which we call *the domain restriction lemma*. It makes use only of the generic condition Ψ^b (see Sec. 11.1.2.2)

Lemma$^{\mathbf{VSE}}$ 110 ($\overset{xy}{\rightsquigarrow} \subseteq DY(kb) \times DY(kb')$):

$$\Psi^b(xy,kb,kb') \Rightarrow ((m \notin DY(kb) \Rightarrow \overset{\rightsquigarrow}{rec}(xy,m) = \{\}) \wedge (m \notin DY(kb') \Rightarrow \overset{\rightsquigarrow}{rec}(rev(xy),m) = \{\}))$$

Proof: The proof is by induction on $|m|$ and it is identical in all message algebras, provided the recursive case of $\overset{\rightsquigarrow}{rec}$ covers all composition alternatives of f-objects, i.e. by constructor-type applications of f and by corresponding synthesis operations if any.

Base Case: We consider an arbitrary $m \in At$ ($m \in At \cup \Sigma\langle 0\rangle$ in message algebras with function symbols in $\Sigma\langle 0\rangle$).

We first assume $\overset{\rightsquigarrow}{rec}(xy,m) \neq \{\}$ and want to show $m \in DY(kb)$. By the definition of $\overset{\rightsquigarrow}{rec}$, we obtain $(m,m') \in xy$ from $\overset{\rightsquigarrow}{rec}(xy,m) \neq \{\}$. This permits to employ Ψ^b and conclude with $m \in DY(kb)$.

Similarly, we assume $\overset{\rightsquigarrow}{rec}(rev(xy),m) \neq \{\}$ and want to show $m \in DY(kb')$. By the definition of $\overset{\rightsquigarrow}{rec}$, we obtain $(m',m) \in xy$ from $\overset{\rightsquigarrow}{rec}(rev(xy),m) \neq \{\}$. This permits to employ Ψ^b and conclude with $m \in DY(kb')$.

Step Case: We consider an arbitrary composed message m.

We first assume $m \notin DY(kb)$ and want to show $\overset{\rightsquigarrow}{rec}(xy,m) = \{\}$ by contradiction. By the definition of $\overset{\rightsquigarrow}{rec}$, we have $\overset{\rightsquigarrow}{rec}(xy,m) \neq \{\}$ only if either xy includes at least one pair (m,m') or m can be composed from sub-messages $m_1,\ldots,m_n$ such that $\overset{\rightsquigarrow}{rec}(xy,m_i) \neq \{\}$ for $1 \leq i \leq n$. In the former case, Ψ^b yields to $m \in DY(kb)$, which refutes the assumption $m \notin DY(kb)$. In the latter case, the induction hypothesis yields to $m_i \in DY(kb)$ for $1 \leq i \leq n$. Using $m_1,\ldots,m_n$ from $DY(kb)$, we are then able to compose m in $DY(kb)$, which refutes the assumption $m \notin DY(kb)$.

The second proof task, i.e. proving $\overset{\rightsquigarrow}{rec}(rev(xy),m) = \{\}$ from $m \notin DY(kb')$, is also by contradiction, replaying mainly the same proof plan as in the first proof task. □

11.1.2.5 Handling of Proof Obligations

The application of the central indistinguishability theorem yields four categories of proof obligations:

- The proof obligations resulting from Ψ^a and Ψ^c, which ensure that $\overset{xy}{\rightsquigarrow}$ are total, are shown by the application of regularity properties followed by inclusion rules in Φ.

  For instance, Ψ^a requires that each password $pw(j)$ in $DY(\pi'.ik)$ must occur in the codomain of xy. This is shown with the help of a regularity property that identifies (besides the case $pw(j) = \pi'$) the regular messages where $pw(j)$ is derived from (see property 115-(3) in Sec. 12.2.1). For each case, an inclusion rule in Φ applies to deduce a pair $(\pi, pw(j))$ or $(pw(j), pw(j))$ in xy (see Sec. 12.4.1).

  Similarly, Ψ^c requires for instance that each $mac(m_1,m_2)$ in $DY(\pi.ik)$ must occur in the domain of xy if m_1 or m_2 is not in $DY(\pi.ik)$. This is shown with the help of two regularity properties that imply $mac(m_1,m_2) \in ik$ and $m_1 \notin DY(ik)$ (see properties 118-(1) and -(2) in Sec. 12.2.4). After that an inclusion rule in Φ applies to deduce a pair $(mac(m_1,m_2), mac(m_1,m_2))$ in xy (see Sec. 12.4.2).

  For types of composed messages without protected sub-messages, the corresponding regularity properties permit to conclude by refutation (cp. the proof of Ψ^c for *gen*-messages (in Sec. 12.4.2) using regularity property 119-(4) in Sec. 12.2.5).

- The proof obligation resulting from Ψ^b, which ensures that the basis relations xy are subsets of $DY(\pi.ik) \times DY(\pi'.ik)$ and are bijective, is shown based on the definition of Φ. The bijection property additionally requires to consider the context of the indistinguishability property, i.e. the Ω assumption, together with confidentiality and unicity properties of the protocol (see Sec. 12.4.3).

- The proof obligations resulting from the rest of Ψ, e.g. Ψ^2_{dec} and Ψ^1_{mac}, which ensures that the basis relations xy are complete, are shown starting with the application of Φ. Being complete means, if a composed message m in a pair of xy possesses derivable sub-messages that could be involved with m in certain derivations, then xy must include corresponding pairs to cover these derivations.

  For instance, Ψ^2_{dec} (in Sec. 11.1.2.1) requires in case $(m_x, dec(m_{y1}, m_{y2}))$ in xy and m_{y1} in $DY(\pi'.ik)$ that xy includes pairs $(m_{x1}, m_{y1}), (enc(m_{x1}, m_x), m_{y2})$. All three pairs cover the derivation of $dec(m_{y1}, m_{y2})$ by composition and that of m_{y2} by extraction. By the minimality of Φ, the pair $(m_x, dec(m_{y1}, m_{y2}))$ matches (as sketched in Sec. 12.4.4) just the case where $m_{y1} = \pi', m_{y2} = enc(\pi, nc(j)), m_x = nc(j)$ and $enc(\pi, nc(j)) \in ik$ hold. This permits to set $m_{x1} = \pi$ and then to obtain the required pairs $(\pi, \pi'), (enc(\pi, nc(j)), enc(\pi, nc(j)))$ by corresponding inclusion rules of Φ.

  In general, most proof obligations are shown by refutation due to the minimality principle of Φ. For example Ψ^1_{mac} (in Sec. 11.1.2.1) requires in case $(mac(m_{x1}, m_{x2}), m_y)$ in xy and m_{x1}, m_{x2} in $DY(\pi.ik)$ that xy includes pairs $(m_{x1}, m_{y1}), (m_{x2}, m_{y2})$ such that $m_y = mac(m_{y1}, m_{y2})$ holds. All these conditions cover a derivation by composition on both sides, i.e. from $\pi.ik$ and from $\pi'.ik$. By the minimality of Φ, the pair $(mac(m_{x1}, m_{x2}), m_y)$ matches (as sketched in Sec. 12.4.5) just the case where $m_y = mac(m_1, m_2), mac(m_1, m_2) \in ik$ and $m_1 \notin DY(ik)$ hold. This permits to apply a regularity property (see property 118-(2) in Sec. 12.2.4) to obtain $m_1 \notin DY(\pi.ik)$ and conclude by refutation using $m_{x1} \in DY(\pi.ik)$ and $m_{x1} = m_1$.

- The proof obligation resulting from Γ, which ensures that every element in $\pi.ik$ is mapped by $\overset{xy}{\rightsquigarrow}$ to the element in $\pi'.ik$ at the same position, is shown using $(\pi, \pi') \in xy$ and the lemma that $m \overset{xy}{\rightsquigarrow} m$ holds for all $m \in DY(ik)$ (see property 120 in Sec. 12.4.6). Note that the proof obligation is handled the same way in all resistance against offline password testing properties, i.e. independent of the considered password protocol.

  The proof of the mentioned lemma is by structural induction on m using in particular the structural mapping lemma 109. The proof steps can be translated to the following properties regarding the mapping of the items in $DY(ik)$ (in case of resistance against password testing, in general):

  1. For atomic messages $c_i \in DY(ik)$, the pairs (c_i, c_i) must be included in the basis relations xy.
  2. For composed messages $m \in DY(ik)$, if m can be derived by composition using sub-messages in $DY(ik)$ then $\overset{xy}{\rightsquigarrow}$ must map these sub-messages to themselves, otherwise the pairs (m, m) must be included in the basis relations xy.

11.2 Proof of the Structural Mapping Lemma

In this section, we describe the proof of lemma 109. It is by induction on the structure of m, i.e. on $|m|$.

11.2.1 Base Case:

The base case is shown employing Ψ^a, Ψ^b and the definition of $\overset{\rightsquigarrow}{rec}$.

Proof of the Base Case: Let m be an atomic message in $DY(kb)$. Then, condition Ψ^a provides (at least) one pair (m,m') in xy. Using Ψ^b and the definition of $\widetilde{rec}$, we obtain $m' \in DY(kb')$ and $m \overset{xy}{\mapsto} m'$. For the proof of $m \overset{xy}{\leftrightarrow} m'$, it remains to show $m \overset{xy}{\leftarrow\!\!\shortmid} m'$.

Since $(m,m') \in xy$, we have $(m',m) \in rev(xy)$, which implies $m \overset{xy}{\curvearrowleft} m'$. To get $m \overset{xy}{\leftarrow\!\!\shortmid} m'$, we assume $m'' \overset{xy}{\curvearrowleft} m'$ and then show $m'' = m$: According to the definition of $\widetilde{rec}$, $m'' \overset{xy}{\curvearrowleft} m'$ and $m' \in dom(rev(xy))$ imply $(m'',m') \in xy$. This permits to use $(m,m'),(m'',m') \in xy$ and apply Ψ^b to obtain $m'' = m$, as required. □

11.2.2 Step Case:

To emphasize the general proof idea, we refer sometimes to synthesis operations to cover message algebras with this kind of composition operations.

In the step case of lemma 109, we consider an arbitrary f-object, i.e. composed message, m in $DY(kb)$. The proof is by case distinction:

- $m \in dom(xy)$: The proof is similar to the Base Case.
- $m \notin dom(xy)$: According to Ψ^c, there is a possible composition of m by a constructor-type application of f (or by a synthesis operation of some g) using sub-messages $m_0,\ldots,m_{n-1} \in DY(kb)$, where n is the arity of f (or respectively g). Applying the definition of $\widetilde{rec}$ and lemma 110, we obtain $m'_0,\ldots,m'_{n-1},m' \in DY(kb')$, satisfying $m_0 \overset{xy}{\rightsquigarrow} m'_0,\ldots,m_{n-1} \overset{xy}{\rightsquigarrow} m'_{n-1}, m \overset{xy}{\rightsquigarrow} m'$ and the equality $m' = f(m'_0,\ldots,m'_{n-1})$ (or respectively $m' = g(m'_0,\ldots,m'_{n-1})$). Due to $|m_0|,\ldots,|m_{n-1}| < |m|$, the induction hypothesis applies to $m_0,\ldots,m_{n-1}$. This implies $m_0 \overset{xy}{\leftrightarrow} m'_0,\ldots,m_{n-1} \overset{xy}{\leftrightarrow} m'_{n-1}$, in particular.

  So, if m can be composed by a *unique* constructor-type application of its head-symbol f, as for example in the case of *mac*-objects (in PACE and *inv*-objects in TC-AMP), then we have $obj^f(m,m_0,\ldots,m_{n-1})$ and the definition of $\widetilde{rec}$ provides us with the mapping $m \overset{xy}{\mapsto} m'$, for $m' = f(m'_0,\ldots,m'_{n-1})$. It remains to show $m \overset{xy}{\leftarrow\!\!\shortmid} m'$ and the other part of the structural condition on the mapped messages m and m'. We want to describe this proof for the case of $f = enc$ (in Sec. 11.2.2.1). This proof plan can be adapted straightforwardly for all other canonical cases, i.e. all other f-objects that can be composed only by a unique constructor-type application of f (see Sec. 11.2.2.2).

  For the non-canonical cases, which necessitate to treat arbitrary many decompositions of m, substantial adaptations are necessary. The proof of these cases is quasi conducted by the definition of the corresponding used predicates: For the non-canonical cases in PACE, we use Ξ_{dh} as described in Sec. 11.3. The non-canonical cases in the TC-AMP algebra are handled with the help of three predicates $\Xi_{\ominus}$, Ξ_{*} and $\Xi_{\oplus}$, as described in Chap. 13.

11.2.2.1 Handling of *enc*-Objects:

In this section, we describe the proof steps to handle *enc*-objects, which can be straightforwardly adapted (see below) for the handling of f-objects that have single decomposition alternatives.

The proof task consists in showing $\Xi_{enc}(m,m',xy,kb,kb')$ in the proof situation where $m,m_0,m_1 \in DY(kb)$, $m \notin dom(xy)$, $obj^{enc}(m,m_0,m_1)$, $m',m'_0,m'_1 \in DY(kb')$, $m_0 \overset{xy}{\leftrightarrow} m'_0$, $m_1 \overset{xy}{\leftrightarrow} m'_1$, $m \overset{xy}{\mapsto} m'$ and $m' = enc(m'_0,m'_1)$ hold. The remaining conjectures (i) $obj^{enc}(m',m'_0,m'_1)$ and (ii) $m \overset{xy}{\leftarrow\!\!\shortmid} m'$ are shown as described below.

11.2.2.1.1 Proof Task (i): We prove $obj^{enc}(enc(m'_0,m'_1),m'_0,m'_1)$ by contradiction, where its negation yields to the assumption $obj^{dec}(m'_1,m'_0,m'_2)$.

We have $(m_1,m'_1) \in xy$ or $\Xi(m_1,m'_1,xy,kb,kb')$ according to the induction hypothesis. This permits to proceed by the following case distinction:

1. $(m_1,m'_1) \in xy$: We first apply Ψ^2_{dec} using $(m_1,dec(m'_0,m'_2)) \in xy$ and $m'_0 \in DY(kb')$ to obtain $(m_{x1},m'_0),(enc(m_{x1},m_1),m'_2) \in xy$. Then, we combine $m_0 \overset{xy}{\leftrightarrow} m'_0$ with $(m_{x1},m'_0) \in xy$ to get $m_{x1} = m_0$. This yields $(enc(m_0,m_1),m'_2) \in xy$, which refutes the assumption $enc(m_0,m_1) \notin dom(xy)$.

2. $(m_1,m'_1) \notin xy$: Here, we want to base the proof on the structure of m'_1, being a *dec*-object. For that purpose, we require the *definition of Ξ to respect a certain mutual-exclusion principle* so that $\Xi(m_1,m'_1,xy,kb,kb')$ for a *dec*-object m'_1 can be reduced to the canonical case Ξ_{dec}. That is, we obtain $m_3,m_4 \in DY(kb)$, $obj^{dec}(m_1,m_3,m_4)$, $m'_3,m'_4 \in DY(kb')$, $m_3 \overset{xy}{\leftrightarrow} m'_3$, $m_4 \overset{xy}{\leftrightarrow} m'_4$ and $obj^{dec}(m'_1,m'_3,m'_4)$. Using $obj^{dec}(m'_1,m'_3,m'_4)$ and $obj^{dec}(m'_1,m'_0,m'_2)$, we get $m'_3 = m'_0$ and then $m_3 \overset{xy}{\leftrightarrow} m'_0$ (from the mapping $m_3 \overset{xy}{\leftrightarrow} m'_3$). Then, we combine $m_3 \overset{xy}{\leftrightarrow} m'_0$ with $m_0 \overset{xy}{\leftrightarrow} m'_0$ to obtain the equality $m_3 = m_0$. Thus, $obj^{dec}(m_1,m_3,m_4)$ is equivalent to $obj^{dec}(m_1,m_0,m_4)$, which refutes $obj^{enc}(enc(m_0,m_1),m_0,m_1)$. □

11.2.2.1.2 Proof Task (ii): After having shown $obj^{enc}(enc(m'_0,m'_1),m'_0,m'_1)$, we want to show the mapping $enc(m_0,m_1) \overset{xy}{\leftarrow} enc(m'_0,m'_1)$. This follows by the induction hypothesis and the definition of $\overset{\rightsquigarrow}{rec}$, provided $enc(m'_0,m'_1) \in dom(rev(xy))$ does not hold. That is, we proceed by assuming $(m_x,enc(m'_0,m'_1)) \in xy$ and aim at a contradiction.

Due to $m_0 \overset{xy}{\leftrightarrow} m'_0$, we know that the crypt-key m'_0 of $enc(m'_0,m'_1)$ is in $DY(kb')$ and this permits to apply condition Ψ^2_{enc} for $(m_x,enc(m'_0,m'_1)) \in xy$. The application of Ψ^2_{enc} yields the pairs $(m_{x1},m'_0),(dec(m_{x1},m_x),m'_1) \in xy$. Combining $m_0 \overset{xy}{\leftrightarrow} m'_0$ with the pair (m_{x1},m'_0) in xy implies $m_{x_1} = m_0$. Similarly, combining $m_1 \overset{xy}{\leftrightarrow} m'_1$ with the pair $(dec(m_{x1},m_x),m'_1)$ in xy implies $m_1 = dec(m_{x1},m_x)$. The obtained equalities permit to deduce $enc(m_0,m_1) = enc(m_{x_1},dec(m_{x1},m_x)) = m_x$ and thus $(m_x,enc(m'_0,m'_1)) \in xy$ refutes the assumption $enc(m_0,m_1) \notin dom(xy)$. □

Note that this application of Ψ^2_{enc} permits to *instantiate* m_x according to the structure of (its mapped message) $enc(m'_0,m'_1)$ substituting m'_0 and m'_1 with their mapped messages. In the following, we often abbreviate such an application of a necessary condition focusing on the instantiation effect. Here, Ψ^2_{enc} is applied to instantiate m_x with $enc(m_0,m_1)$.

11.2.2.2 Handling Other Canonical Cases:

All other canonical cases are handled by replaying the proof steps in Sec. 11.2.2.1. For the handling of *dec*-objects, the proof steps are adapted by switching "*enc*" and "*dec*", also in the used necessary conditions.

11.2.2.2.1 Proof Task (i): This proof task is relevant only for the f-objects where applications of function symbols f are not always constructor-type. It is handled by assuming $\neg obj^f(f(m'_0,\ldots,m'_{n-1}),m'_0,\ldots,m'_{1-n})$ and using that to obtain the structure for some m'_i. After that, the structural condition for the mapping $m_i \overset{xy}{\leftrightarrow} m'_i$ permits to propagate the structure of m'_i to m_i and then to refute one of the assumptions on $f(m_0,\ldots,m_{n-1})$. This is done with the help of an appropriate inclusion condition in case $(m_i,m'_i) \in xy$ (cp. situation (1)) and with the help of the Ξ-definition in the complementary case (cp. situation (2)).

Proof Details: For the handling of fst-objects, the assumption $\neg obj^{fst}(fst(m'_0),m'_0)$ yields $obj^{pair}(m'_0,m'_1,m'_2)$ and this permits to apply Ψ^2_{pair} in situation (1). This yields $m_0 = pair(m_{x1},m_{x2})$, which refutes $obj^{fst}(fst(m_0),m_0)$.

The handling of snd-objects is similar, based also on Ψ^2_{pair}. □

11.2.2.2.2 Proof Task (ii): The handling of this task is mainly by assuming the pair $(m_x,f(m'_0,\ldots,m'_{n-1}))$ in xy and then using an appropriate necessary condition to instantiate m_x with $f(m_0,\ldots,m_{n-1})$, which permits to refute $f(m_0,\ldots,m_{n-1}) \notin dom(xy)$.

The handling of $pair$-objects is based on Ψ^2_{pair}. For the handling of fst-, snd-, gen- and respectively mac-objects, we use the necessary conditions Ψ^2_{fst}, Ψ^2_{snd}, Ψ^2_{gen} and respectively Ψ^2_{mac}.

11.3 Handling of the non-canonical Ξ_{dh} Case

The Ξ_{dh}-case in lemma 109 defines the mapping of dh-objects by decomposition. Since multiple decomposition alternatives are possible, we use a definition of Ξ_{dh} that covers also nested dh-parts obtained by successive decomposition. This definition employs the Θ^{dh}_1 predicate to qualify the sub-message where the successive decomposition halts.

$$\Theta^{dh}_1(m,ik) \Leftrightarrow \forall m_0,m_1 \in DY(ik) : m \neq dh(m_0,m_1)$$

Accordingly, $\Theta^{dh}_1(x,kb)$ means in the following definition of Ξ_{dh} that x does not possess available dh-parts in $DY(kb)$.

$$\begin{aligned}\Xi_{dh}: \quad & \exists ms \subset DY(kb), x \in DY(kb), ms' \subset DY(kb'), x' \in DY(kb') : \\ & m = \overline{dh}(x,ms) \wedge ms \neq \emptyset \wedge \Theta^{dh}_1(x,kb) \wedge \wp(ms,ms') \subset^{\overset{xy}{\leftrightarrow}} \wedge \\ & x \overset{xy}{\leftrightarrow} x' \wedge m' = \overline{dh}(x',ms')\end{aligned}$$

The predicate Ξ_{dh} defines the mapped message m' to a dh-object m (with available dh-parts) relative to *arbitrarily many* sub-messages (x and the elements of ms) of m obtained by successive decomposition into available also nested dh-parts: m' is the result of successive applications of dh using the messages mapped to x and to the elements of ms.

Before we use the predicate Ξ_{dh} in the proof of theorem 108 (see Sec. 11.4), we describe the handling of dh-objects, i.e. the Ξ_{dh}-case, in the step case of lemma 109: For $m \in DY(kb)$, $m \notin dom(xy)$ and $isObj^{dh}(m)$, we need to provide $m' \in DY(kb')$ satisfying $m \overset{xy}{\leftrightarrow} m'$ and $\Xi_{dh}(m,m',xy,kb,kb')$.

The proof starts by applying lemma 111 for *a successive decomposition of m into available dh-parts in* $DY(kb)$. It provides $ms \subset DY(kb)$ and $x \in DY(kb)$ with $m = \overline{dh}(x,ms)$, $ms \neq \emptyset$ and $\Theta^{dh}_1(x,kb)$. Since all elements in ms and x are smaller than m, the induction hypothesis provides $x' \in DY(kb')$ and $ms' \subset DY(kb')$ with $x \overset{xy}{\leftrightarrow} x'$ and $\wp(ms,ms') \subset^{\overset{xy}{\leftrightarrow}}$, where $\wp(ms,ms')$ denotes a finite set of pairs mapping each element in ms to a unique element in ms'.

We continue the proof by setting $m' = \overline{dh}(x',ms')$ and showing $m \overset{xy}{\leftrightarrow} m'$ as follows:

- First, we apply lemma 112 *to obtain the mapping of m by* $\overset{xy}{\mapsto}$ *using the available dh-parts*. It permits to show $\overline{dh}(x,ms) \overset{xy}{\mapsto} \overline{dh}(x',ms')$, i.e. $m \overset{xy}{\mapsto} \overline{dh}(x',ms')$.
- Then, we refute $\overline{dh}(x',ms') \in codom(xy)$, i.e. $(m_x,\overline{dh}(x',ms')) \in xy$, with the help of the necessary condition Ψ^2_{dh}: Using Ψ^2_{dh}, we instantiate m_x with $\overline{dh}(x,ms)$, i.e. m, to refute $m \notin dom(xy)$.

The instantiation of m_x with $\overline{dh}(x,ms)$ is by successive applications of Ψ^2_{dh} permitting to obtain $\wp(ms_{x2},ms') \subset xy$, $(m_{x1},x') \in xy$ and $m_x = \overline{dh}(m_{x1},ms_{x2})$. Then, we use $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$ and $x \overset{xy}{\leftrightarrow} x'$ to deduce $ms_{x2} = ms$ and $m_{x1} = x$ and thus $m_x = \overline{dh}(x,ms)$.

- Finally, we use $\overline{dh}(x',ms') \notin codom(xy)$ and apply lemma 113 *to obtain the mapping of* $\overline{dh}(x',ms')$ *by* $\overset{xy}{\leftarrow}$ *using the available dh-parts*. It permits to show $\overline{dh}(x,ms) \overset{xy}{\leftarrow} \overline{dh}(x',ms')$, i.e. $\overline{dh}(x,ms) \overset{xy}{\leftarrow} m'$.

11.3.1 Decomposition into dh-Parts in $DY(kb)$

For dh-objects, the following lemma permits to identify the result of a successive decomposition into available dh-parts.

Lemma[VSE] 111 (Decomposition into dh-Parts in $DY(kb)$):
Let m be a dh-object in $DY(kb)$ that can be decomposed in two dh-parts in $DY(kb)$. Then, it exists $ms \subset DY(kb)$ and $x \in DY(kb)$ satisfying

$$m = \overline{dh}(x,ms) \wedge ms \neq \emptyset \wedge \Theta^{dh}_1(x,kb).$$

Proof: The proof (by induction on $|m|$) is trivial. In the base case, m is of the form $dh(c_i,c_j)$ for atomic messages $c_i,c_j \in DY(kb)$. So, we have $\Theta^{dh}_1(c_i,kb)$ and this permits to set $ms = \{c_j\}$ and $x = c_i$.

In the step case, we consider arbitrary $m_0,m_1 \in DY(kb)$ satisfying $m = dh(m_0,m_1)$ and continue by a case distinction:

1. In case m_0 is not a dh-object or m_0 is a dh-object without a pair of dh-parts in $DY(kb)$, we have $\Theta^{dh}_1(m_0,kb)$ and this permits to set $ms = \{m_1\}$ and $x = m_0$.
2. In the complementary case, i.e. m_0 is a dh-object with a pair of dh-parts in $DY(kb)$, the induction hypothesis provides x_0 and ms_0 satisfying $m_0 = \overline{dh}(x_0,ms_0)$, $ms_0 \neq \emptyset$ and $\Theta^{dh}_1(x_0,kb)$. This permits to set $x = x_0$ and $ms = ms_0 \uplus m_1$, where $ms_0 \uplus m_1$ abbreviates $ms_0 \uplus \{m_1\}$, i.e. the extension of the multiset ms_0 with the message m_1. □

11.3.2 Mapping by $\overset{xy}{\mapsto}$ using the dh-Parts in $DY(kb)$

For dh-objects that are not mapped in xy, the following lemma provides the mapping by $\overset{xy}{\mapsto}$ relative to the mappings of the available dh-parts.

Lemma[VSE] 112 (Mapping of dh-Objects by $\overset{xy}{\mapsto}$):
Let $ms \subset DY(kb)$, $x \in DY(kb)$, $ms' \subset DY(kb)$ and $x' \in DY(kb)$ satisfy $ms \neq \emptyset$, $\Theta^{dh}_1(x,kb)$, $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$, $x \overset{xy}{\leftrightarrow} x'$ and $\overline{dh}(x,ms) \notin dom(xy)$. Let (the induction hypothesis of) lemma 109 hold for all $\hat{m}$ with $|\hat{m}| < |\overline{dh}(x,ms)|$. Then, we have

$$\overline{dh}(x,ms) \overset{xy}{\mapsto} \overline{dh}(x',ms').$$

Proof: Since $\overline{dh}(x,ms) \notin dom(xy)$ and $ms \neq \emptyset$ hold, we prove $\overline{dh}(x,ms) \overset{xy}{\mapsto} \overline{dh}(x',ms')$ by considering arbitrary $m_0,m_1 \in DY(kb)$ with $dh(m_0,m_1) = \overline{dh}(x,ms)$ and showing that m'_0 and m'_1 in the mappings $m_0 \overset{xy}{\leftrightarrow} m'_0$ and $m_1 \overset{xy}{\leftrightarrow} m'_1$, which are provided by (the induction hypothesis of) lemma 109, satisfy the equality $dh(m'_0,m'_1) = \overline{dh}(x',ms')$.

For $dh(m_0,m_1) = \overline{dh}(x,ms)$ and $ms \neq \emptyset$, we distinguish two cases:

1. In case $m_1 \in ms$, we have $ms = m_1 \uplus ms_1$ and $m_0 = \overline{dh}(x, ms_1)$. Furthermore, $\wp(ms, ms') \subset \overset{xy}{\leftrightarrow}$ means that there is ms'_1 with $ms' = m'_1 \uplus ms'_1$ and $\wp(ms_1, ms'_1) \subset \overset{xy}{\leftrightarrow}$. In the following, we want to show that $m'_0 = \overline{dh}(x', ms'_1)$ to prove the required equality by

$$\begin{aligned} dh(m'_0, m'_1) &= dh(\overline{dh}(x', ms'_1), m'_1) \\ &= \overline{dh}(x', ms'_1 \uplus m'_1) \\ &= \overline{dh}(x', ms'). \end{aligned}$$

If $ms_1 = \emptyset$, we obtain $m_0 = x$, $m'_0 = x'$, and $ms'_1 = \emptyset$. This permits to show the required equality by $\overline{dh}(x', ms'_1) = \overline{dh}(x', \emptyset) = x' = m'_0$.

In the complementary case, i.e. $ms_1 \neq \emptyset$, m_0 is a dh-object and this permits to show $m'_0 = \overline{dh}(x', ms'_1)$ based on the structural condition for $m_0 \overset{xy}{\leftrightarrow} m'_0$ by the following case distinction:

(a) When $(m_0, m'_0) \in xy$, i.e. $(\overline{dh}(x, ms_1), m'_0) \in xy$, we apply Ψ^1_{dh} to instantiate m'_0 with $\overline{dh}(x', ms'_1)$.

(b) When $\Xi(m_0, m'_0, xy, kb, kb')$, the structure of $m_0 = \overline{dh}(x, ms_1)$ permits to reduce this to the Ξ_{dh}-case. That is, we get $m_0 = \overline{dh}(x_0, ms_0)$, $ms_0 \neq \emptyset$, $\Theta^{dh}_1(x_0, kb)$, $\wp(ms_0, ms'_0) \subset \overset{xy}{\leftrightarrow}$, $x_0 \overset{xy}{\leftrightarrow} x'_0$ and $m'_0 = \overline{dh}(x'_0, ms'_0)$.
Based on $m_0 = \overline{dh}(x, ms_1) = \overline{dh}(x_0, ms_0)$, $x, x_0 \in DY(kb)$, $ms_1, ms_0 \subset DY(kb)$, $\Theta^{dh}_1(x, kb)$ and $\Theta^{dh}_1(x_0, kb)$, we obtain two cases:

- $x = x_0$ and $ms_1 = ms_0$: Here, $m'_0 = \overline{dh}(x'_0, ms'_0)$ rewrites immediately to the equality $m'_0 = \overline{dh}(x', ms'_1)$.
- $x = \overline{dh}(u, mx)$, $x_0 = \overline{dh}(u, mx_0)$, $ms_1 = ms_{01} \uplus mx_0$ and $ms_0 = ms_{01} \uplus mx$ for $mx, mx_0 \neq \emptyset$ satisfying $mx \neq mx_0$: Based on $\Theta^{dh}_1(x, kb)$, the structure of $x = \overline{dh}(u, mx)$ and on $mx \subset DY(kb)$, the mapping $x \overset{xy}{\leftrightarrow} x'$ implies $(\overline{dh}(u, mx), x') \in xy$ with $u \notin DY(kb)$. Similarly, $x_0 \overset{xy}{\leftrightarrow} x'_0$ implies $(\overline{dh}(u, mx_0), x'_0) \in xy$ with $u \notin DY(kb)$.
This proof situation is handled with the help of the necessary condition $\Psi^{1,2}_{dh}$: The application of $\Psi^{1,2}_{dh}$ for $(\overline{dh}(u, mx), x'), (\overline{dh}(u, mx_0), x'_0) \in xy$ yields $\wp(mx, mx'), \wp(mx_0, mx'_0) \subseteq xy$, $x' = \overline{dh}(z, mx')$ and $x'_0 = \overline{dh}(z, mx'_0)$ for some $z \notin DY(kb')$. Taking all equalities and mappings into consideration, we show $m'_0 = \overline{dh}(x', ms'_1)$ as follows:

$$\begin{aligned} m'_0 &= \overline{dh}(x'_0, ms'_0) \\ &= \overline{dh}(\overline{dh}(z, mx'_0), ms'_{01} \uplus mx') \\ &= \overline{dh}(\overline{dh}(z, mx'), ms'_{01} \uplus mx'_0) \\ &= \overline{dh}(x', ms'_1). \end{aligned}$$

2. In case $m_1 \notin ms$, we have $x = dh(x_1, m_1)$ and $m_0 = \overline{dh}(x_1, ms)$. Based on $\Theta^{dh}_1(x, kb)$, the structure of $x = dh(x_1, m_1)$ and on $m_1 \in DY(kb)$, the mapping $x \overset{xy}{\leftrightarrow} x'$ implies $(dh(x_1, m_1), x') \in xy$ with $x_1 \notin DY(kb)$. Then, we proceed by case distinction according to the structural condition for the mapping $m_0 \overset{xy}{\leftrightarrow} m'_0$:

(a) When $(m_0,m_0') \in xy$, i.e. $(\overline{dh}(x_1,ms),m_0') \in xy$, we employ $ms \neq \{m_1\}$ and $x_1 \notin DY(kb)$ to apply $\Psi_{dh}^{1,2}$ for $(\overline{dh}(x_1,ms),m_0') \in xy$ and $(dh(x_1,m_1),x') \in xy$ and obtain $m_0' = \overline{dh}(z,ms')$ and $x' = dh(z,m_1')$ for some $z \notin DY(kb')$. Hence, the required equality can be shown by

$$\begin{aligned} dh(m_0',m_1') &= dh(\overline{dh}(z,ms'),m_1') \\ &= \overline{dh}(dh(z,m_1'),ms') \\ &= \overline{dh}(x',ms_1'). \end{aligned}$$

(b) When $\Xi(m_0,m_0',xy,kb,kb')$, the structure of $m_0 = \overline{dh}(x_1,ms)$ permits to focus on the Ξ_{dh}-case, where we obtain $m_0 = \overline{dh}(x_0,ms_0)$, $ms_0 \neq \emptyset$, $\Theta_1^{dh}(x_0,kb)$, $\wp(ms_0,ms_0') \subset \overset{xy}{\leftrightarrow}$, $x_0 \overset{xy}{\leftrightarrow} x_0'$ and $m_0' = \overline{dh}(x_0',ms_0')$. Based on $m_0 = \overline{dh}(x_0,ms_0) = \overline{dh}(x_1,ms)$, $x_0 \in DY(kb)$, $x_1 \notin DY(kb)$, $ms_0 \neq \emptyset$, $ms \neq \emptyset$, and $ms_0 \subset DY(kb)$, we deduce the equalities $ms_0 = ms_{01} \uplus mx_1$, $ms = ms_{01} \uplus mx_0$, $x_0 = \overline{dh}(u,mx_0)$ and $x_1 = \overline{dh}(u,mx_1)$ for $mx_0 \neq \emptyset$. Then, the structure of $x_0 = \overline{dh}(u,mx_0)$ in $x_0 \overset{xy}{\leftrightarrow} x_0'$, $mx_0 \neq \emptyset$ and $mx_0 \subset DY(kb)$ permit to obtain $(\overline{dh}(u,mx_0),x_0') \in xy$ with $u \notin DY(kb)$. After that, we use $x_1 = \overline{dh}(u,mx_1)$ to rewrite $(dh(x_1,m_1),x') \in xy$ to $(\overline{dh}(u,mx_1 \uplus m_1),x') \in xy$. Since $m_1 \notin ms$ and $ms = ms_{01} \uplus mx_0$, the obtained pairs $(\overline{dh}(u,mx_0),x_0') \in xy$ and $(\overline{dh}(u,mx_1 \uplus m_1),x') \in xy$ satisfy $mx_0 \neq mx_1 \uplus m_1$. This permits, together with $u \notin DY(kb)$, to apply $\Psi_{dh}^{1,2}$ and obtain $x_0' = \overline{dh}(z,mx_0')$ and $x' = \overline{dh}(z,mx_1' \uplus m_1')$ for $z \notin DY(kb')$. Taking all equalities and mappings into consideration, we show $dh(m_0',m_1') = \overline{dh}(x',ms')$ as follows:

$$\begin{aligned} dh(m_0',m_1') &= dh(\overline{dh}(x_0',ms_0'),m_1') \\ &= dh(\overline{dh}(\overline{dh}(z,mx_0'),ms_{01}' \uplus mx_1'),m_1') \\ &= \overline{dh}(\overline{dh}(z,mx_1' \uplus m_1'),mx_0' \uplus ms_{01}') \\ &= \overline{dh}(x',ms') \end{aligned}$$

□

11.3.3 Mapping by $\overset{xy}{\hookleftarrow}$ using the dh-Parts in $DY(kb')$

For dh-objects that do not occur in $codom(xy)$, the following lemma provides the mapping by $\overset{xy}{\hookleftarrow}$ relative to the mappings of the available dh-parts.

Lemma[VSE] 113 (Mapping of dh-Objects by $\overset{xy}{\hookleftarrow}$):
Let $ms \subset DY(kb)$, $x \in DY(kb)$, $ms' \subset DY(kb)$ and $x' \in DY(kb)$ satisfy $ms \neq \emptyset$, $\Theta_1^{dh}(x,kb)$, $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$, $x \overset{xy}{\leftrightarrow} x'$ and $\overline{dh}(x',ms') \notin codom(xy)$. Let (the induction hypothesis of) lemma 109 hold for all $\hat{m}$ with $|\hat{m}| < |\overline{dh}(x,ms)|$. Then, we have

$$\overline{dh}(x,ms) \overset{xy}{\hookleftarrow} \overline{dh}(x',ms').$$

Proof: Since $\overline{dh}(x',ms') \notin codom(xy)$ and $ms' \neq \emptyset$ hold, we prove $\overline{dh}(x,ms) \overset{xy}{\hookleftarrow} \overline{dh}(x',ms')$ by considering arbitrary $m_0',m_1' \in DY(kb)$ with $dh(m_0',m_1') = \overline{dh}(x',ms')$, providing m_0 and m_1 *uniquely* mapped to m_0' and respectively m_1' and showing that m_0 and m_1 satisfy the equality $dh(m_0,m_1) = \overline{dh}(x,ms)$.

For $dh(m_0',m_1') = \overline{dh}(x',ms')$ and $ms' \neq \emptyset$, we distinguish two cases:

1. In case $m'_1 \in ms'$, the multiset ms' equals $m'_1 \uplus ms'_1$ and m'_0 equals $\overline{dh}(x', ms'_1)$. Furthermore, $\wp(ms, ms') \subset^{xy}_{\leftrightarrow}$ means that there is ms_1 with $ms = m_1 \uplus ms_1$, $m_1 \overset{xy}{\leftrightarrow} m'_1$ and $\wp(ms_1, ms'_1) \subset^{xy}_{\leftrightarrow}$. In the following, we want to show that $\overline{dh}(x, ms_1) \overset{xy}{\leftrightarrow} m'_0$ holds to set $m_0 = \overline{dh}(x, ms_1)$ and prove $dh(m_0, m_1) = \overline{dh}(x, ms)$ by

$$dh(m_0, m_1) = dh(\overline{dh}(x, ms_1), m_1) = \overline{dh}(x, ms_1 \uplus m_1) = \overline{dh}(x, ms).$$

If $ms_1 = \emptyset$, we obtain $\overline{dh}(x, ms_1) = x$, $ms'_1 = \emptyset$ and $m'_0 = \overline{dh}(x', ms'_1) = x'$. This permits to rewrite $x \overset{xy}{\leftrightarrow} x'$ to $\overline{dh}(x, ms_1) \overset{xy}{\leftrightarrow} m'_0$.

In the complementary case, i.e. $ms_1 \neq \emptyset$, $\overline{dh}(x, ms_1)$ is a dh-object that is smaller than $\overline{dh}(x, ms)$. Thus, (the induction hypothesis of) lemma 109 provides $\overline{dh}(x, ms_1) \overset{xy}{\leftrightarrow} m'$ and it remains to show $m' = m'_0 = \overline{dh}(x', ms'_1)$ based on the structural condition for $\overline{dh}(x, ms_1) \overset{xy}{\leftrightarrow} m'$ by the following case distinction:

   (a) When $(\overline{dh}(x, ms_1), m') \in xy$, we use Ψ^1_{dh} to instantiate m' with $\overline{dh}(x', ms'_1)$.

   (b) When $\Xi(\overline{dh}(x, ms_1), m', xy, kb, kb')$, the structure of $\overline{dh}(x, ms_1)$ permits to reduce this to the Ξ_{dh}-case. That is, we get $\overline{dh}(x, ms_1) = \overline{dh}(x_0, ms_0)$, $ms_0 \neq \emptyset$, $\Theta^{dh}_1(x_0, kb)$, $\wp(ms_0, ms'_0) \subset^{xy}_{\leftrightarrow}$, $x_0 \overset{xy}{\leftrightarrow} x'_0$ and $m' = \overline{dh}(x'_0, ms'_0)$.
   Based on $\overline{dh}(x, ms_1) = \overline{dh}(x_0, ms_0)$, $x, x_0 \in DY(kb)$, $ms_1, ms_0 \subset DY(kb)$, $\Theta^{dh}_1(x, kb)$ and $\Theta^{dh}_1(x_0, kb)$, we obtain two cases:

   - $x = x_0$ and $ms_1 = ms_0$: Here, $m' = \overline{dh}(x'_0, ms'_0)$ rewrites immediately to the equality $m' = \overline{dh}(x', ms'_1)$.
   - $x = \overline{dh}(u, mx)$, $x_0 = \overline{dh}(u, mx_0)$, $ms_1 = ms_{01} \uplus mx_0$ and $ms_0 = ms_{01} \uplus mx$ for $mx, mx_0 \neq \emptyset$ satisfying $mx \neq mx_0$: Based on $\Theta^{dh}_1(x, kb)$, the structure of $x = \overline{dh}(u, mx)$ and on $mx \subset DY(kb)$, the mapping $x \overset{xy}{\leftrightarrow} x'$ implies $(\overline{dh}(u, mx), x') \in xy$ with $u \notin DY(kb)$. Similarly, $x_0 \overset{xy}{\leftrightarrow} x'_0$ implies $(\overline{dh}(u, mx_0), x'_0) \in xy$ with $u \notin DY(kb)$. This permits to use $\Psi^{1,2}_{dh}$ and obtain the instantiations $x' = \overline{dh}(z, mx')$ and $x'_0 = \overline{dh}(z, mx'_0)$ for $z \notin DY(kb')$. Taking all equalities and mappings into consideration, we show $m' = \overline{dh}(x', ms'_1)$ by

$$\begin{aligned} m' &= \overline{dh}(x'_0, ms'_0) \\ &= \overline{dh}(\overline{dh}(z, mx'_0), ms'_{01} \uplus mx') \\ &= \overline{dh}(\overline{dh}(z, mx'), ms'_{01} \uplus mx'_0) \\ &= \overline{dh}(x', ms'_1). \end{aligned}$$

2. In case $m'_1 \notin ms'$, we have $x' = dh(x'_1, m'_1)$ and $m'_0 = \overline{dh}(x'_1, ms')$. Based on $\Theta^{dh}_1(x, kb)$, the structure of $x' = dh(x'_1, m'_1)$, the mapping $x \overset{xy}{\leftrightarrow} x'$ implies $(x, dh(x'_1, m'_1)) \in xy$ and $\Theta^{dh}_1(dh(x'_1, m'_1), kb')$. The latter holds, because $(x, dh(x'_1, m'_1)) \in xy$ permits to map any assumed available dh-parts of $dh(x'_1, m'_1)$ by Ψ^2_{dh} to available dh-parts of x, refuting $\Theta^{dh}_1(x, kb)$. Using $\Theta^{dh}_1(dh(x'_1, m'_1), kb')$, $dh(x'_1, m'_1) \in DY(kb')$ and $m'_1 \in DY(kb')$, we deduce $x'_1 \notin DY(kb')$. This, together with $\overline{dh}(x'_1, ms') \in DY(kb')$, i.e. $m'_0 \in DY(kb')$, permits to employ lemma 114 and obtain $x'_1 = \overline{dh}(z, xs')$, $ms' = zs' \uplus us'$, $zs' \neq \emptyset$, $\overline{dh}(z, zs') \in DY(kb')$, $z \notin DY(kb')$, $xs' \uplus us' \subset DY(kb')$, and $\Theta^{dh}_1(\overline{dh}(z, zs'), kb')$. That is, $\overline{dh}(z, zs') \in DY(kb')$, $zs' \neq \emptyset$ and $\Theta^{dh}_1(\overline{dh}(z, zs'), kb')$ imply $(x_z, \overline{dh}(z, zs')) \in xy$ for some $x_z \in DY(kb)$. Furthermore, the equality $x'_1 = \overline{dh}(z, xs')$ permits to rewrite

$(x,dh(x'_1,m'_1)) \in xy$ to $(x,\overline{dh}(z,xs' \uplus m'_1)) \in xy$. Here, $zs' \neq xs' \uplus m'_1$, because $m'_1 \notin zs'$ ensues from $m'_1 \notin ms'$. This, together with $zs', xs' \uplus m'_1 \subset DY(kb')$ and $z \notin DY(kb')$, permits to use the pairs $(x,\overline{dh}(z,xs' \uplus m'_1)),(x_z,\overline{dh}(z,zs')) \in xy$ and apply $\Psi^{2,2,}_{dh}$. The application of $\Psi^{2,2}_{dh}$ provides $\wp(xs \uplus m_1, xs' \uplus m'_1), \wp(zs,zs') \subset xy$, $x_z = \overline{dh}(u,zs)$ and $x = \overline{dh}(u, xs \uplus m_1)$, for $u \notin DY(kb)$: Due to $ms' = zs' \uplus us'$ and $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$, we have $\wp(us,us'), \wp(zs,zs') \subset \overset{xy}{\leftrightarrow}$ and $zs \uplus us = ms$, for some us.

All these allow us to set $m_0 = \overline{dh}(u, zs \uplus xs \uplus us) = \overline{dh}(x_z, ms_z)$ for ms_z abbreviating $xs \uplus us$, and then to show $m_0 \overset{xy}{\leftrightarrow} m'_0$, i.e. $m_0 \overset{xy}{\leftrightarrow} \overline{dh}(x'_1, ms')$, before we prove $dh(m_0,m_1) = \overline{dh}(x,ms)$ below.

First, $x = \overline{dh}(u, xs \uplus m_1)$ and $zs \uplus us = ms$ permit to show that $m_0 = \overline{dh}(u, zs \uplus xs \uplus us)$ is smaller than $\overline{dh}(x,ms)$. Thus, (the induction hypothesis of) lemma 109 provides a mapping $m_0 \overset{xy}{\leftrightarrow} m'$ whose structural condition permits to show $m' = \overline{dh}(x'_1, ms')$ in the following case distinction:

(a) When $(m_0,m') \in xy$, we apply condition Ψ^1_{dh} for $(\overline{dh}(x_z,ms_z),m') \in xy$ to obtain $\wp(ms_z, xs' \uplus us') \subset xy$, $(x_z,\overline{dh}(z,zs')) \in xy$ and $m' = \overline{dh}(\overline{dh}(z,zs'), xs' \uplus us')$. This permits to show $m' = \overline{dh}(x'_1,ms')$ by

$$m' = \overline{dh}(\overline{dh}(z,zs'), xs' \uplus us') = \overline{dh}(\overline{dh}(z,xs'), zs' \uplus us') = \overline{dh}(x'_1,ms').$$

(b) When $\Xi(\overline{dh}(x_z,ms_z), m', xy, kb, kb')$, the structure of $\overline{dh}(x_z,ms_z)$ permits to reduce this to the Ξ_{dh}-case. This means, $\overline{dh}(x_z,ms_z) = \overline{dh}(x_0,ms_0)$, $ms_0 \neq \emptyset$, $\Theta^{dh}_1(x_0,kb)$, $\wp(ms_0,ms'_0) \subset \overset{xy}{\leftrightarrow}$, $x_0 \overset{xy}{\leftrightarrow} x'_0$ and $m' = \overline{dh}(x'_0,ms'_0)$.

Recall, x_z is mapped to $\overline{dh}(z,zs')$ in $(x_z,\overline{dh}(z,zs')) \in xy$, where $\Theta^{dh}_1(\overline{dh}(z,zs'),kb')$ holds. This permits to deduce $\Theta^{dh}_1(x_z,kb)$ based on Ψ^2_{dh}.

Based on $x_z, x_0 \in DY(kb)$, $ms_z, ms_0 \subset DY(kb)$, $\Theta^{dh}_1(x_z,kb)$ and $\Theta^{dh}_1(x_0,kb)$, the equality $\overline{dh}(x_z,ms_z) = \overline{dh}(x_0,ms_0)$ yields two cases:

- $x_z = x_0$ and $ms_z = ms_0$: In this case, $m' = \overline{dh}(x'_0,ms'_0)$ rewrites to the equality $m' = \overline{dh}(\overline{dh}(z,zs'), xs' \uplus us')$, which permits to show $m' = \overline{dh}(x'_1,ms')$ as in (2-a), above.
- $x_z = \overline{dh}(x_{z0}, mx_z)$, $x_0 = \overline{dh}(x_{z0}, mx_0)$, $ms_z = ms_{z0} \uplus mx_0$ and $ms_0 = ms_{z0} \uplus mx_z$ for $mx_z, mx_0 \neq \emptyset$ satisfying $mx_z \neq mx_0$: Here, $x_{z0} \notin DY(kb)$ holds, otherwise $\Theta^{dh}_1(x_z,kb)$ would be refuted. Additionally, $\Theta^{dh}_1(x_0,kb)$ and the structure of $x_0 = \overline{dh}(x_{z0},mx_0)$ permit to deduce $(\overline{dh}(x_{z0},mx_0),x'_0) \in xy$ from $x_0 \overset{xy}{\leftrightarrow} x'_0$. That is, we have the required conditions to apply $\Psi^{1,2}_{dh}$ for $(\overline{dh}(x_{z0},mx_z),\overline{dh}(z,zs')),(\overline{dh}(x_{z0},mx_0),x'_0) \in xy$ and obtain the equalities $\overline{dh}(z,zs') = \overline{dh}(v,mx'_z)$ and $x'_0 = \overline{dh}(v,mx'_0)$, for $v \notin DY(kb')$.

  Taking all equalities and mappings into consideration, we show $m' = \overline{dh}(x'_1,ms')$ as follows:

$$\begin{aligned}
m' &= \overline{dh}(x'_0,ms'_0)\\
&= \overline{dh}(\overline{dh}(v,mx'_0), ms'_{z0} \uplus mx'_z)\\
&= \overline{dh}(\overline{dh}(v,mx'_z), ms'_{z0} \uplus mx'_0)\\
&= \overline{dh}(\overline{dh}(z,zs'), ms'_z)\\
&= \overline{dh}(\overline{dh}(z,zs'), xs' \uplus us')\\
&= \overline{dh}(\overline{dh}(z,xs'), zs' \uplus us')\\
&= \overline{dh}(x'_1,ms').
\end{aligned}$$

Having $m_0 \overset{xy}{\leftrightarrow} \overline{dh}(x_1', ms')$ for $m_0 = \overline{dh}(u, zs \uplus xs \uplus us)$ and the above assumed or shown equalities and mappings, we want to conclude with the proof of $dh(m_0, m_1) = \overline{dh}(x, ms)$ as follows:

$$\begin{aligned} dh(m_0, m_1) &= dh(\overline{dh}(u, zs \uplus xs \uplus us), m_1) \\ &= \overline{dh}(\overline{dh}(u, xs \uplus m_1), zs \uplus us) \\ &= \overline{dh}(x, ms). \end{aligned}$$

□

11.3.4 Auxiliary Lemma on *dh*-Parts

For a *dh*-message $\overline{dh}(x, ms)$ with a protected left *dh*-part x, the following lemma permits to identify a sub-message satisfying Θ_1^{dh}. It is composed from (a protected left *dh*-part of) x and right *dh*-parts from ms.

Lemma$^{\mathbf{VSE}}$ 114 (Lemma on Protected *dh*-Parts):
Let ik be an arbitrary finite message set, ms an arbitrary non-empty multiset of messages, and let x be an arbitrary message. Then, we have

$$\begin{aligned} &(\overline{dh}(x, ms) \in DY(ik) \wedge x \notin DY(ik)) \Rightarrow \\ &(\exists z, xs, zs : x = \overline{dh}(z, xs) \wedge z \notin DY(ik) \wedge xs \subset DY(ik) \wedge \\ &\qquad zs \neq \emptyset \wedge zs \subseteq ms \wedge \overline{dh}(z, zs) \in DY(ik) \wedge \Theta_1^{dh}(\overline{dh}(z, zs), ik)). \end{aligned}$$

Proof: The proof of this lemma is by induction on $|\overline{dh}(x, ms)|$:

- In the base case, we have $\overline{dh}(x, ms) = dh(c_i, c_j)$ for atomic messages c_i and c_j. This means, $x = c_i$ and $ms = \{c_j\}$. Since x is not a *dh*-object, $x \notin DY(kb)$ implies $\Theta_1^{dh}(\overline{dh}(x, ms), ik)$ and this permits to conclude by setting $z = x$, $xs = \emptyset$ and $zs = ms$.
- In the step case, the proof is similar to the base case when $\overline{dh}(x, ms)$ does not possess any tuple of available *dh*-parts.

  For the complementary case, let $\overline{dh}(x, ms) = dh(m_0, m_1)$ where (m_0, m_1) is an arbitrary tuple of available *dh*-parts in $DY(ik)$. We distinguish two cases:

  1. In case $m_1 \in ms$, we have $ms = ms_0 \uplus m_1$ and $m_0 = \overline{dh}(x, ms_0)$. Furthermore, ms_0 may not be empty, because otherwise m_0 and x would be equal and this refutes $x \notin DY(ik)$. All these permit to conclude immediately applying the induction hypothesis to $\overline{dh}(x, ms_0) \in DY(ik)$.
  2. In case $m_1 \notin ms$, we have $x = dh(x_1, m_1)$ and $m_0 = \overline{dh}(x_1, ms)$. Here, x_1 must not be in $DY(ik)$, because otherwise it can be used with the available m_1 to derive x (and this refutes the assumption $x \notin DY(ik)$). Due to $x_1 \notin DY(ik)$, the induction hypothesis applies to $\overline{dh}(x_1, ms) \in DY(ik)$ and yields $x_1 = \overline{dh}(z_1, xs_1)$, $z_1 \notin DY(ik)$, $xs_1 \subset DY(ik)$, $zs_1 \neq \emptyset$, $zs_1 \subseteq ms$, $\overline{dh}(z_1, zs_1) \in DY(ik)$ and $\Theta_1^{dh}(\overline{dh}(z_1, zs_1), ik)$. This, together with $m_1 \in DY(ik)$, allows us to conclude by setting $z = z_1$, $xs = xs_1 \uplus m_1$ and $zs = zs_1$. □

11.4 Proof of the Central Indistinguishability Theorem

In this section, we describe the proof of (the central indistinguishability) theorem 108. The proof is by induction on the indexes of the generic derivations provided by their enumeration.

11.4.1 Base Case

In the base case, we show $\overline{DY}(kb,l) \overset{xy}{\leftrightarrow} \overline{DY}(kb',l)$ for the indexes l that satisfy $0 \leq l < len(kb)$. Using $\overline{DY}(kb,l) = sel(l,kb)$ and $\overline{DY}(kb',l) = sel(l,kb')$, our proof goal can be proven straightforwardly with the help of lemma 109 and based on $\Gamma(xy,kb,kb')$: Lemma 109 allows us to transform $sel(l,kb) \overset{xy}{\rightsquigarrow} sel(l,kb')$ to $sel(l,kb) \overset{xy}{\leftrightarrow} sel(l,kb')$, as required.

11.4.2 Step Case

In the step case, we need to show $\overline{DY}(kb,l) \overset{xy}{\leftrightarrow} \overline{DY}(kb',l)$ for the l-th generic derivations $\overline{DY}(.,l)$ where $l \geq len(kb)$. Here, the function "*orig*" (formalized in Sec. 10.4) fixes the last applied function symbol $f \in \Sigma\langle n \rangle$ and the indexes $i_0,\ldots,i_{n-1} < l$ of the arguments used in the derivation of $\overline{DY}(kb,l)$ and $\overline{DY}(kb',l)$ by the application of f. These arguments correspond to $\overline{DY}(kb,i_0),\ldots,\overline{DY}(kb,i_{n-1})$ and $\overline{DY}(kb',i_0),\ldots,\overline{DY}(kb',i_{n-1})$, respectively. In the rest of the proof, we set

$$m_l = \overline{DY}(kb,l),\ m_j = \overline{DY}(kb,i_j),\ m'_l = \overline{DY}(kb',l) \text{ and } m'_j = \overline{DY}(kb',i_j) \text{ for } 0 \leq j < n.$$

We have thus $m_l = f(m_0,\ldots,m_{n-1})$ and $m'_l = f(m'_0,\ldots,m'_{n-1})$ and the induction hypothesis provides us with $m_j \overset{xy}{\leftrightarrow} m'_j$. The proof task consists then in showing $m_l \overset{xy}{\leftrightarrow} m'_l$ using the mappings $m_j \overset{xy}{\leftrightarrow} m'_j$ and considering the application effects of f. The effects of f permit to obtain a case distinction on the structures of m_l and/or $m_0,\ldots,m_{n-1}$. In each case, we apply lemma 109 focusing on the corresponding structured message (m_l or $m_i \in \{m_0,\ldots,m_{n-1}\}$) to use the obtained mapping and structural condition for providing the appropriate m'_l and showing the equality $m'_l = f(m'_0,\ldots,m'_{n-1})$.

In the following, we start with the canonical proof situations, where composed messages have unique decomposition alternatives and where the possible non-constructor type operations can be only decrypt-type or selector-type. We first describe (in Sec. 11.4.2.1) the proof plan for $m_l = enc(m_0,m_1)$ and then discuss how this proof plan can be straightforwardly adapted to other canonical cases (see Sec. 11.4.2.2). Finally, we describe (in Sec. 11.4.2.3) the handling of the sole non-canonical proof situation, i.e. case $f = dh$.

11.4.2.1 Handling the Case for $f = enc$:

In this section, we describe the proof steps for handling the case where $m_l = enc(m_0,m_1)$. They can be straightforwardly adapted (see below) for the handling of other canonical cases: The proof plan for the $\mathbf{obj^{enc}}$-case applies to other constructor-type cases and that for the $\mathbf{dec^{enc}_{dec}}$-case is adapted to other decrypt-type or selector-type cases.

In the rest of this section, we assume the proof situation where $m_0 \overset{xy}{\leftrightarrow} m'_0$, $m_1 \overset{xy}{\leftrightarrow} m'_1$ and $m'_l = enc(m'_0,m'_1)$ hold. We need to show $enc(m_0,m_1) \overset{xy}{\leftrightarrow} m'_l$.

11.4.2.1.1 $\mathbf{obj^{enc}}$-Case: The obj^f-case is a typical canonical proof situation that is shown with the help of lemma 109 as described below.

Proof Details: Since the message $enc(m_0,m_1)$ belongs to the set $DY(kb)$, lemma 109 provides us with $enc(m_0,m_1) \overset{xy}{\leftrightarrow} m'$ for some m' in $DY(kb')$ satisfying $(enc(m_0,m_1),m') \in xy$ or $\Xi(enc(m_0,m_1),m',xy,kb,kb')$. This yields the unique mapping of $enc(m_0,m_1)$ with m' and permits us to show $enc(m'_0,m'_1) = m'$ by case distinction:

1. Case $(enc(m_0,m_1),m') \in xy$ is handled by applying Ψ^1_{enc} to instantiate m' with $enc(m'_0,m'_1)$ (cp. the use of Ψ^2_{enc} in Sec. 11.2.2.1.2).

2. In case $(enc(m_0,m_1),m') \notin xy$ and $\Xi(enc(m_0,m_1),m',xy,kb,kb')$, the definition of Ξ and the assumption $obj^{enc}(enc(m_0,m_1),m_0,m_1)$ permit to focus on the canonical case for *enc*-objects. That is, we obtain $m_0 \overset{xy}{\leftrightarrow} m_0''$, $m_1 \overset{xy}{\leftrightarrow} m_1''$ and $obj^{enc}(m',m_0'',m_1'')$. These mappings combined with $m_0 \overset{xy}{\leftrightarrow} m_0'$ and $m_1 \overset{xy}{\leftrightarrow} m_1'$ yield the equalities $m_0'' = m_0'$ and $m_1'' = m_1'$, which permit to use $obj^{enc}(m',m_0'',m_1'')$ and deduce the required equality $enc(m_0',m_1') = m'$. □

11.4.2.1.2 $\mathbf{dec}^{\mathbf{enc}}_{\mathbf{dec}}$-Case: In case of the decrypt-type application of *enc* to m_0 and m_1, where $obj^{dec}(m_1,m_0,m_l)$ holds, we apply lemma 109 focusing on the structured message m_1. This yields together with $m_1 \overset{xy}{\leftrightarrow} m_1'$ to $(m_1,m_1') \in xy$ or $\Xi(m_1,m_1',xy,kb,kb')$. Accordingly, we proceed by the following case distinction:

1. Case $(m_1,m_1') \in xy$ is handled by applying condition Ψ^1_{dec} to obtain the pairs $(m_0,m_{y1}),(m_l,enc(m_{y1},m_1')) \in xy$. This permits to map m_l to $enc(m_{y1},m_1')$, as indicated in $(m_l,enc(m_{y1},m_1')) \in xy$. Then, we show $enc(m_0',m_1') = enc(m_{y1},m_1')$, using the equality $m_0' = m_{y1}$, which ensues from the combination of $(m_0,m_{y1}) \in xy$ and $m_0 \overset{xy}{\leftrightarrow} m_0'$.

2. In case $(m_1,m_1') \notin xy$ and $\Xi(m_1,m_1',xy,kb,kb')$, the definition of Ξ and the assumption $obj^{dec}(m_1,m_0,m_l)$ permit to focus on the canonical case for *dec*-objects. That is, we obtain $m_0 \overset{xy}{\leftrightarrow} m_0''$, $m_l \overset{xy}{\leftrightarrow} m_1''$ and $obj^{dec}(m_1',m_0'',m_1'')$.

   As indicated in $m_l \overset{xy}{\leftrightarrow} m_1''$, we map m_l to m_1'' and want to show $enc(m_0',m_1') = m_1''$. For that purpose, we combine $m_0 \overset{xy}{\leftrightarrow} m_0''$ and $m_0 \overset{xy}{\leftrightarrow} m_0'$ to obtain the equality $m_0' = m_0''$. Then, we employ this equality and $obj^{dec}(m_1',m_0'',m_1'')$ to deduce $enc(m_0',m_1') = enc(m_0'',dec(m_0'',m_1'')) = m_1''$, as required. □

11.4.2.2 Handling Other Canonical Cases:

All other canonical cases are handled by replaying the proof steps in Sec. 11.4.2.1. For case $f = dec$, the proof steps are adapted by switching "*enc*" and "*dec*", also in the used necessary conditions.

11.4.2.2.1 $\mathbf{obj}^{\mathbf{f}}$-Case: When $m_l = f(m_0,\ldots,m_{n-1})$ is an f-object, the proof steps in Sec. 11.4.2.1.1 apply: Lemma 109 provides the required mapping $f(m_0,\ldots,m_{n-1}) \overset{xy}{\leftrightarrow} m'$ with its structural condition that we use to show $f(m_0',\ldots,m_{n-1}') = m'$. This is done based on a corresponding necessary condition in case $(f(m_0,\ldots,m_{n-1}),m') \in xy$ (cp. situation (1)) and with the help of the Ξ-definition in the complementary case (cp. situation (2)).

The handling of the cases $f = fst$, $f = snd$, $f = pair$, $f = mac$ and $f = gen$, respectively, is based on the necessary conditions Ψ^1_{fst}, Ψ^1_{snd}, Ψ^1_{pair}, Ψ^1_{mac} and respectively Ψ^1_{gen}.

11.4.2.2.2 Complementary Case: When the application of f in $m_l = f(m_0,\ldots,m_{n-1})$ is decrypt-type or selector-type, the proof situation is handled by replaying the proof steps in Sec. 11.4.2.1.2. The decrypt-type or selector-type operation implies generally some structure for a m_i relative to m_l. This permits to expand the structural condition for the mapping $m_i \overset{xy}{\leftrightarrow} m_i'$ given by lemma 109. It is done with the help of a corresponding necessary condition in case $(m_i,m_i') \in xy$ (cp. situation (1)) and with the help of the Ξ-definition in the complementary case (cp. situation (2)). In both cases, we shall obtain a mapped message to m_l that can be shown equal to $f(m_0',\ldots,m_{n-1}')$.

Proof Details: In case $f = fst$, the non-constructor-type application of fst in $m_l = fst(m_0)$ is selector-type. It implies $obj^{pair}(m_0, m_l, m_2)$ and the structural condition for $m_0 \overset{xy}{\leftrightarrow} m'_0$ is expanded in case $(m_0, m'_0) \in xy$ with the help of Ψ^1_{pair}. This permits to obtain a *pair*-structure for m'_0 and a mapped message to m_l that is a pendant of $fst(m'_0)$.

The complementary case, i.e. $(m_0, m'_0) \notin xy$, is handled similarly based on the canonical case for *pair*-objects in Ξ.

Case $f = snd$ is handled similarly. □

11.4.2.3 Handling the Case for $f = dh$:

For case $f = dh$, we assume $m_l = dh(m_0, m_1)$, $m_0 \overset{xy}{\leftrightarrow} m'_0$, $m_1 \overset{xy}{\leftrightarrow} m'_1$ and $m'_l = dh(m'_0, m'_1)$, for $m_l, m_0, m_1 \in DY(kb)$ and $m'_l, m'_0, m'_1 \in DY(kb')$. The proof consists then in providing $m_l \overset{xy}{\leftrightarrow} m'$ and showing $m' = m'_l$, i.e. $m' = dh(m'_0, m'_1)$.

Since $m_l \in DY(kb)$, lemma 109 provides us with $m_l \overset{xy}{\leftrightarrow} m'$ for $m' \in DY(kb')$, where $(m_l, m') \in xy$ or $\Xi_{dh}(m_l, m', xy, kb, kb')$ holds.

1. In case $(m_l, m') \in xy$, i.e. $(dh(m_0, m_1), m') \in xy$, we show $m' = dh(m'_0, m'_1)$ with the help of Ψ^1_{dh}.

2. In case $\Xi_{dh}(m_l, m', xy, kb, kb')$ holds, we have $m_l = \overline{dh}(x, ms)$, $ms \neq \emptyset$, $\Theta^{dh}_1(x, kb)$, $\wp(ms, ms') \subset \overset{xy}{\leftrightarrow}$, $x \overset{xy}{\leftrightarrow} x'$ and $m' = \overline{dh}(x', ms')$. Here, $dh(m_0, m_1) = \overline{dh}(x, ms)$ yields two cases:

   (a) $m_1 \in ms$, $ms = m_1 \uplus ms_1$ and $m_0 = \overline{dh}(x, ms_1)$.

   (b) $m_1 \notin ms$, $x = dh(x_1, m_1)$ and $m_0 = \overline{dh}(x_1, ms)$.

   Both cases are handled as in the proof of lemma 112 (see Sec. 11.3.2) for the proof of $dh(m'_0, m'_1) = \overline{dh}(x', ms')$, permitting to obtain $m' = dh(m'_0, m'_1)$, as required.

Chapter 12

Resistance Proof of PACE

The resistance of PACE against offline password testing is formalized in property 105 (see Sec. 11.1). In this chapter, we provide the details to its proof applying the proof technique from Chap. 11: After the definition of the basis relations (in Sec. 12.1), we present the required regularity properties (in Sec. 12.2), then we describe the proof of the basis simulation relation lemma (in Sec. 12.3). Finally, we show (in Sec. 12.4) the proof obligations resulting from the conditions of the central indistinguishability theorem 108.

12.1 Definition of the Basis Relations

According to the formalization in 105, the relevant traces tr in the resistance of PACE against offline password guessing include a first or a fourth PACE message with an occurrence of a confidential password π. Furthermore, the indistinguishability is needed to be shown for knowledge bases kb, kb' defined by adding π respectively π' (non-occurring in tr) to $ik = spies(tr)$. The mentioned assumptions for this proof goal are abbreviated by $\Omega(tr, \pi, \pi', kb, kb')$ (Ω assumption).

The first proof task consists in identifying the contents of the sets xy that are used as basis relations. These sets are defined relative to ik and to the parameters π, π'. As described in Sec. 11.1.1.2, we define the sets xy to be the smallest sets of message pairs that satisfy the (protocol- and property-specific) inclusion rules Φ_1–Φ_5, below. With these rules, we ensure that the domains and codomains of the basis relations xy include all derivable composed messages that possess protected sub-messages (cp. Ψ^c). Additionally, the domains and codomains of xy include all derivable atomic messages (cp. Ψ^a).

Φ_1 If there is a nonce nc with $enc(\pi, nc) \in ik$ or there is g, m_1, m_2 and a nonce nc with $obj^{dec}(dec(\pi, m_1), \pi, m_1)$ and $dh(gen(dh(g, dec(\pi, m_1)), m_2), nc) \in ik$, then the set xy includes the pair (π, π').

Φ_2 If there exists $pw(i)$ and $num(j)$ with $pair(ag(i), pair(pw(i), num(j))) \in ik$, then xy includes $(ag(i), ag(i))$.

If there is $ag(j)$ and $pw(j)$ with $pair(ag(j), pair(pw(j), num(i))) \in ik$ or there is $pw(j)$ with $pair(pw(j), num(i)) \in ik$, then xy includes $(num(i), num(i))$.

If there is $ag(i)$ and $num(j)$ with $pair(ag(i), pair(pw(i), num(j))) \in ik$ or there is $num(j)$ with $pair(pw(i), num(j)) \in ik$, then xy includes $(pw(i), pw(i))$.

Φ_3 If $nc(i) \in ik$ or there is m with $pair(m, nc(i)) \in ik$ or there exists $pw(j) \in DY(ik)$ with $enc(pw(j), nc(i)) \in ik$, then xy includes $(nc(i), nc(i))$.

Φ_4 If $enc(\pi, nc) \in ik$, then xy includes $(nc, dec(\pi', enc(\pi, nc)))$.

Φ_5 If $enc(pw(j),nc(i)) \in ik$ with $pw(i) \notin DY(ik)$, then the set xy includes the pair $(enc(pw(j),nc(i)),enc(pw(j),nc(i)))$.

If $dh(m_1,m_2) \in ik$ with $\neg isObj^{dh}(m_1)$ and $m_2 \notin DY(ik)$, then the set xy includes the pair $(dh(m_1,m_2),dh(m_1,m_2))$.

If $dh(dh(m_1,m_2),m_3) \in ik$ with $\neg isObj^{dh}(m_1)$ and $m_2,m_3 \notin DY(ik)$, then xy includes $(dh(dh(m_1,m_2),m_3),dh(dh(m_1,m_2),m_3))$.

If $mac(m_1,m_2) \in ik$ with $m_1 \notin DY(ik)$, then xy includes $(mac(m_1,m_2),mac(m_1,m_2))$.

According to Φ_1, the presence of a first or a fourth PACE message with an occurrence of π necessitates to include the pair (π,π').

Φ_2 states when pairs (c_i,c_i) are included for public atomic messages c_i. Φ_3 is similar but focuses on the public nonces.

Φ_4 covers the case where the added π permits to extract protected nonces in first PACE messages.

Φ_5 ensures the inclusion of pairs for the four possible composed messages with protected sub-messages. This includes the first PACE messages where the added π is used as the first *enc*-part.

In the following, we use $\Phi(xy,ik,\pi,\pi')$ (as in the basis simulation relation lemma 107) to abbreviate the definition of xy being the smallest set that satisfies the inclusion rules Φ_1–Φ_5.

12.2 Employed Regularity Properties

Besides the formalization of the required regularity properties, we explain how they are shown by trace induction.

12.2.1 Derivable Atomic Messages

In this section, we describe the employed regularity properties about the derivable atomic messages. They state where these atomic messages originate from.

In PACE, atomic messages are distinguished in agent names ($ag(j)$), nonces ($nc(j)$), passwords ($pw(j)$) and numerical data ($num(j)$) used as static DH generators. So, we obtain four regularity properties on the derivable, atomic messages.

Property$^{\text{VSE}}$ 115 (Derivable Atomic Messages):

1. $(tr \in \text{PACE} \wedge ag(j) \in DY(pw(i).spies(tr))) \Rightarrow$
   $(\exists k : pair(ag(j),pair(pw(j),num(k))) \in spies(tr))$

2. $(tr \in \text{PACE} \wedge num(j) \in DY(pw(i).spies(tr))) \Rightarrow$
   $(\exists k : pair(ag(k),pair(pw(k),num(j))) \in spies(tr) \vee$
   $pair(pw(k),num(j)) \in spies(tr))$

3. $(tr \in \text{PACE} \wedge pw(j) \in DY(pw(i).spies(tr))) \Rightarrow$
   $(pw(j) = pw(i) \vee$
   $(\exists k : pair(ag(j),pair(pw(j),num(k))) \in spies(tr) \vee$
   $pair(pw(j),num(k)) \in spies(tr)))$

4. $(tr \in \text{PACE} \wedge nc(j) \in DY(pw(i)\bullet spies(tr))) \Rightarrow$
$(enc(pw(i),nc(j)) \in spies(tr) \vee nc(j) \in spies(tr) \vee$
$(\exists m \in DY(spies(tr)) : pair(m,nc(j)) \in spies(tr)) \vee$
$(\exists k : enc(pw(k),nc(j)) \in spies(tr) \wedge pw(k) \in DY(spies(tr))))$

Properties 1–3 are shown according to a similar proof plan, with the help of ccl_2 and its correctness theorem 53. For instance, property 3 is shown by contradiction: We assume the negated conclusion and refute the assumption $pw(j) \in DY(pw(i)\bullet spies(tr))$, employing the following property together with the correctness theorem 53:

$$(tr \in \text{PACE} \wedge pw(j) \neq pw(i) \wedge$$
$$(\forall k : pair(ag(j),pair(pw(j),num(k))) \notin spies(tr) \wedge pair(pw(j),num(k)) \notin spies(tr)))$$
$$\Rightarrow (pw(i)\bullet spies(tr)) \cap ccl_2(pw(j)) = \emptyset$$

The proof of property 4 is slightly different, because nonces $nc(j)$ can occur as a protected crypt-part of a regular message $enc(pw(k),nc(j))$. So, we need to handle the case where $nc(j)$ could originate from such a regular message separately from the complementary case:

- There is k such that $enc(pw(k),nc(j))$ belongs to $spies(tr)$: Relying on property 3, which defines where $pw(k)$ for decrypting $nc(j)$ originates from, we distinguish the following cases:
  - $pw(k) = pw(i)$: This yields obviously to $enc(pw(i),nc(j)) \in spies(tr)$, i.e. case (1).
  - $pair(ag(k),pair(pw(k),num(l)))$ or $pair(pw(k),num(l))$ is in $spies(tr)$): This yields clearly to $pw(k) \in DY(spies(tr))$, i.e. case (4).
  - The negation of the previous cases permits to prove $nc(j) \notin DY(pw(i)\bullet spies(tr))$ with the help of ccl_1 using $S = \{pw(k),nc(j)\}$:

    $$(tr \in \text{PACE} \wedge enc(pw(k),nc(j)) \in spies(tr) \wedge pw(k) \neq pw(i) \wedge$$
    $$(\forall l : pair(ag(k),pair(pw(k),num(l))),pair(pw(k),num(l)) \notin spies(tr)))$$
    $$\Rightarrow (pw(i)\bullet spies(tr)) \cap ccl_1(\{pw(k),nc(j)\}) = \emptyset$$

    The confidentiality of $nc(j)$ permits to close this case by refuting the assumption $nc(j) \in DY(pw(i)\bullet spies(tr))$.
- $enc(pw(k),nc(j))$ does not belong to $spies(tr)$ for all k: We proceed like in properties 1–3, employing the following property:

  $$(tr \in \text{PACE} \wedge (\forall k : enc(pw(k),nc(j)) \notin spies(tr)) \wedge nc(j) \notin spies(tr) \wedge$$
  $$(\forall m \in DY(spies(tr)) : pair(m,nc(j)) \notin spies(tr)))$$
  $$\Rightarrow (pw(i)\bullet spies(tr)) \cap ccl_2(nc(j)) = \emptyset$$

12.2.2 Derivable *enc*-Messages

In this section, we describe the employed regularity properties about the derivable *enc*-messages. They permit to identify the derivable *enc*-messages whose *enc*-parts are confidential.

Property$^{\mathbf{VSE}}$ 116 (Derivable *enc*-Messages):

1. $(tr \in \text{PACE} \wedge m \in DY(pw(i).spies(tr)) \wedge obj^{enc}(m,m_1,m_2)) \Rightarrow$
   $((m_1,m_2 \in DY(pw(i).spies(tr))) \vee m \in DY(spies(tr)))$

2. $(tr \in \text{PACE} \wedge m \in DY(spies(tr)) \wedge obj^{enc}(m,m_1,m_2)) \Rightarrow$
   $((m_1,m_2 \in DY(spies(tr))) \vee$
   $(\exists j,k : m = enc(pw(j),nc(k)) \wedge m \in spies(tr)))$

These properties are shown by contradiction, according to a similar proof plan as in Sec. 12.2.1. For property (1), the case with an assumed negated conclusion is handled by refuting the assumption $m \in DY(pw(i).spies(tr))$, employing the following property together with a correctness theorem of ccl_2 similar to 54:

$$(tr \in \text{PACE} \wedge obj^{enc}(m,m_1,m_2) \wedge \neg(m_1,m_2 \in DY(pw(i).spies(tr))) \wedge m \notin DY(spies(tr))) \Rightarrow (pw(i).spies(tr)) \cap ccl_2(m) = \emptyset$$

For property (2), the case with an assumed negated conclusion is handled by refuting the assumption $m \in DY(spies(tr))$, employing the following property:

$$(tr \in \text{PACE} \wedge obj^{enc}(m,m_1,m_2) \wedge \neg(m_1,m_2 \in DY(spies(tr))) \wedge (\forall j,k : m \neq enc(pw(j),nc(k)) \vee m \notin spies(tr))) \Rightarrow spies(tr) \cap ccl_2(m) = \emptyset$$

12.2.3 Derivable *dh*-Messages

In this section, we describe the employed regularity properties about the derivable *dh*-messages. They permit to identify the derivable *dh*-messages having confidential *dh*-parts.

Property$^{\mathbf{VSE}}$ 117 (Derivable *dh*-Messages):

1. $(tr \in \text{PACE} \wedge dh(m_1,m_2) \in DY(pw(i).spies(tr))) \Rightarrow$
   $((\exists m_1',m_2' \in DY(pw(i).spies(tr)) : dh(m_1',m_2') = dh(m_1,m_2)) \vee$
   $dh(m_1,m_2) \in DY(spies(tr)))$

2. $(tr \in \text{PACE} \wedge dh(m_1,m_2) \in DY(spies(tr))) \Rightarrow$
   $((\exists m_1',m_2' \in DY(spies(tr)) : dh(m_1',m_2') = dh(m_1,m_2)) \vee$
   $(dh(m_1,m_2) \in spies(tr) \wedge$
   $((\neg isObj^{dh}(m_1) \wedge m_2 \notin DY(pw(i).spies(tr))) \vee$
   $(\exists m_3,m_4 : m_1 = dh(m_3,m_4) \wedge \neg isObj^{dh}(m_3) \wedge m_2,m_4 \notin DY(pw(i).spies(tr))))))$

The proof of property (1) is similar to that of property 116-(1). The refutation of the assumption $dh(m_1,m_2) \in DY(pw(i).spies(tr))$ is performed employing the following property together with the correctness theorem 55:

$$(tr \in \text{PACE} \wedge dh(m_1,m_2) \notin DY(spies(tr)) \wedge \\ \neg(\exists m_1',m_2' \in DY(pw(i).spies(tr)) : dh(m_1',m_2') = dh(m_1,m_2)))) \\ \Rightarrow (pw(i).spies(tr)) \cap ccl_2(dh(m_1,m_2)) = \emptyset$$

Property (2) is shown by case distinction: If $dh(m_1,m_2)$ can be derived from dh-parts in $DY(ik)$, the proof is trivial. Otherwise, we distinguish again two complementary cases:

- In case $dh(m_1,m_2) \in ik$ holds, we use the following PACE property to obtain the required cases:

$$(tr \in \text{PACE} \wedge \neg(\exists m_1',m_2' \in DY(ik) : dh(m_1',m_2') = dh(m_1,m_2)) \wedge \\ spies(tr) \cap ccl_2(dh(m_1,m_2)) \neq \emptyset) \Rightarrow \\ ((\neg isObj^{dh}(m_1) \wedge m_2 \notin DY(pw(i).spies(tr))) \vee \\ (\exists m_3,m_4 : m_1 = dh(m_3,m_4) \wedge \neg isObj^{dh}(m_3) \wedge \\ m_2,m_4 \notin DY(pw(i).spies(tr))))$$

  In PACE, regular dh-messages occur in steps 2–5 and in the oops-case. While the former dh-messages have *unique* right dh-parts, the latter dh-message has *exactly two* permutable right dh-parts. In particular, all these right dh-parts remain confidential also in case ik is extended (by the attacker) with a genuine password $pw(i)$.

- In case $dh(m_1,m_2) \notin ik$ holds, we refute the assumption $dh(m_1,m_2) \in DY(ik)$, employing the following property:

$$(tr \in \text{PACE} \wedge \neg(\exists m_1',m_2' \in DY(ik) : dh(m_1',m_2') = dh(m_1,m_2)) \wedge \\ dh(m_1,m_2) \notin spies(tr)) \\ \Rightarrow spies(tr) \cap ccl_2(dh(m_1,m_2)) = \emptyset$$

12.2.4 Derivable *mac*-Messages

In this section, we describe the employed regularity properties about the derivable *mac*-messages. They permit to identify the derivable *mac*-messages having confidential *mac*-parts.

Property$^{\text{VSE}}$ 118 (Derivable *mac*-Messages):

1. $(tr \in \text{PACE} \wedge mac(m_1,m_2) \in DY(pw(i).spies(tr))) \Rightarrow \\ ((m_1,m_2 \in DY(pw(i).spies(tr))) \vee mac(m_1,m_2) \in DY(spies(tr)))$

2. $(tr \in \text{PACE} \wedge mac(m_1,m_2) \in DY(spies(tr))) \Rightarrow \\ ((m_1,m_2 \in DY(spies(tr))) \vee \\ (mac(m_1,m_2) \in spies(tr) \wedge m_1 \notin DY(pw(i).spies(tr))))$

The proof of property (1) is similar to that of property 116-(1). The refutation of the assumption $mac(m_1,m_2) \in DY(pw(i).spies(tr))$ is performed employing the following property together with the correctness theorem 54:

$$(tr \in \text{PACE} \wedge \neg(m_1,m_2 \in DY(pw(i).spies(tr))) \wedge mac(m_1,m_2) \notin DY(spies(tr))) \\ \Rightarrow (pw(i).spies(tr)) \cap ccl_2(mac(m_1,m_2)) = \emptyset$$

Property (2) is shown by case distinction: If $mac(m_1,m_2)$ can be derived from mac-parts in $DY(ik)$, the proof is trivial. Otherwise, we distinguish again two complementary cases:

- In case $mac(m_1,m_2) \in ik$ holds, we use the following PACE property to obtain the premises of the forward secrecy property (of m_1):

$$\begin{aligned}
&(tr \in \text{PACE} \wedge \neg(m_1,m_2 \in DY(spies(tr))) \wedge \\
&\ spies(tr) \cap ccl_2(mac(m_1,m_2)) \neq \emptyset) \Rightarrow \\
&(\exists ag_1,ag_2,ag,ag',nc_1,nc_2,m_3 : ag_1 \notin bad \wedge ag_2 \notin bad \wedge isObj^{gen}(m_3) \wedge \\
&\qquad m_2 = dh(m_3,nc_2) \wedge m_1 = dh(m_2,nc_1) \wedge \\
&\qquad send(ag_1,ag,nc_1,dh(m_3,nc_1)) \in tr \wedge \\
&\qquad send(ag_2,ag',nc_2,m_2) \in tr \wedge note(spy,dh(m_2,nc_1)) \notin tr)
\end{aligned}$$

  Applying the forward secrecy property permits to obtain $m_1 \notin DY(pw(i).spies(tr))$, as required.

- In case $mac(m_1,m_2) \notin ik$ holds, we refute the assumption $mac(m_1,m_2) \in DY(ik)$, employing the following property:

$$\begin{aligned}
&(tr \in \text{PACE} \wedge \neg(m_1,m_2 \in DY(spies(tr))) \wedge mac(m_1,m_2) \notin spies(tr)) \\
&\Rightarrow spies(tr) \cap ccl_2(mac(m_1,m_2)) = \emptyset
\end{aligned}$$

12.2.5 Derivable *dec*-, *fst*-, *snd*- and *gen*-Messages

In this section, we describe the employed regularity properties about the derivable composed messages that can be derived *only* by composition.

According to the regular messages in PACE, the derivable *dec*-, *fst*-, *snd*- and *gen*-objects do not possess confidential *dec*-, *fst*-, *snd*- and respectively *gen*-parts:

Property$^{\text{VSE}}$ 119 (Trivial Composed Messages):

1. $(tr \in \text{PACE} \wedge obj^{dec}(m,m_1,m_2) \wedge m \in DY(pw(i).spies(tr))) \Rightarrow$
   $m_1,m_2 \in DY(pw(i).spies(tr))$

2. $(tr \in \text{PACE} \wedge obj^{dec}(m,m_1,m_2) \wedge m \in DY(spies(tr))) \Rightarrow m_1,m_2 \in DY(spies(tr))$

3. $(tr \in \text{PACE} \wedge obj^{fst}(m,m_1) \wedge m \in DY(pw(i).spies(tr))) \Rightarrow m_1 \in DY(pw(i).spies(tr))$

4. $(tr \in \text{PACE} \wedge obj^{fst}(m,m_1) \wedge m \in DY(spies(tr))) \Rightarrow m_1 \in DY(spies(tr))$

5. $(tr \in \text{PACE} \wedge obj^{snd}(m,m_1) \wedge m \in DY(pw(i).spies(tr))) \Rightarrow m_1 \in DY(pw(i).spies(tr))$

6. $(tr \in \text{PACE} \wedge obj^{snd}(m,m_1) \wedge m \in DY(spies(tr))) \Rightarrow m_1 \in DY(spies(tr))$

7. $(tr \in \text{PACE} \wedge gen(m_1,m_2) \in DY(pw(i).spies(tr))) \Rightarrow m_1,m_2 \in DY(pw(i).spies(tr))$

8. $(tr \in \text{PACE} \wedge gen(m_1,m_2) \in DY(spies(tr))) \Rightarrow m_1,m_2 \in DY(spies(tr))$

The proof of properties 1–8 is similar to that of property 116-(1). In the proof of property (1), for instance, the case with an assumed negated conclusion is handled by refuting the assumption $m \in DY(pw(i).spies(tr))$, employing the following property together with a correctness theorem of ccl_2 similar to 54:

$$(tr \in \text{PACE} \wedge obj^{dec}(m, m_1, m_2) \wedge \neg(m_1, m_2 \in DY(pw(i).spies(tr))))$$
$$\Rightarrow (pw(i).spies(tr)) \cap ccl_2(m) = \emptyset$$

Accordingly, the proof of property (2) employs the following similar property and the same correctness theorem:

$$(tr \in \text{PACE} \wedge obj^{dec}(m, m_1, m_2) \wedge \neg(m_1, m_2 \in DY(spies(tr))))$$
$$\Rightarrow spies(tr) \cap ccl_2(m) = \emptyset$$

12.3 Proof of the Basis Simulation Relation Lemma

The basis simulation relation lemma 107 is shown by induction on PACE traces.

The base case is trivial: The empty trace does not belong to the relevant traces, because it does not include neither a first nor a fourth PACE message.

In the step case, we distinguish whether there is a regular occurrence of π in ik.

Case I: If there is a regular occurrence of π in ik, the induction hypothesis provides xy_{ik} that satisfies $\Phi(xy_{ik}, ik, \pi, \pi')$. For the extended observable messages $ik_{ex} = ik \cup ik_{ev}$, we need to define xy_{ex} relative to ik and ik_{ev} and show that $\Phi(xy_{ex}, ik \cup ik_{ev}, \pi, \pi')$ holds:

1. In case $ev = note(ag', pair(ag(i), pair(pw(i), num(j))))$ and $ag' \in bad$, this event yields $ik_{ev} = \{pair(ag(i), pair(pw(i), num(j)))\}$ and we distinguish two cases:

   (a) If $pw(i) \notin DY(ik)$, we define xy_{ex} by

   $$\begin{aligned} xy_{ex} := (&xy_{ik} \setminus \{(m, m) \mid m = enc(pw(i), nc) \wedge m \in ik\}) \\ &\cup \{(nc, nc) \mid enc(pw(i), nc) \in ik\} \\ &\cup \{(pw(i), pw(i)), (ag(i), ag(i)), (num(j), num(j))\}. \end{aligned}$$

   (b) Otherwise, we define xy_{ex} by

   $$xy_{ex} := xy_{ik} \cup \{(ag(i), ag(i))\}.$$

   Note that the pairs $(enc(pw(i), nc), enc(pw(i), nc))$ in xy_{ik} must be replaced with (nc, nc), in order not to violate Φ_3 and the first rule in Φ_5 (while preserving the minimality of xy_{ex}).

   In case $ev = note(ag, pair(pw(i), num(j)))$ and $ag \in bad$, we proceed according to the same principle.

2. In case $ev = says(ag, ag', enc(pw(i), nc(j)))$, we have $ik_{ev} = \{enc(pw(i), nc(j))\}$ and we distinguish three cases:

   (a) If $pw(i) = \pi$, we define xy_{ex} by

   $$xy_{ex} := xy_{ik} \cup \{(enc(\pi, nc(j)), enc(\pi, nc(j))), (nc(j), dec(\pi', enc(\pi, nc(j))))\}.$$

(b) Otherwise, if $pw(i) \notin DY(ik)$, we define xy_{ex} by

$$xy_{ex} := xy_{ik} \cup \{(enc(pw(i),nc(j)),enc(pw(i),nc(j)))\}.$$

(c) Else, we define xy_{ex} by

$$xy_{ex} := xy_{ik} \cup \{(nc(j),nc(j))\}.$$

3. In case $ev = send(ag,ag',pair(m_1,nc(i)),dh(num(j),nc(i)))$, we have $ik_{ev} = \{pair(m_1,nc(i)),dh(num(j),nc(i))\}$, if $ag \in bad$ and $ik_{ev} = \{dh(num(j),nc(i))\}$, otherwise. In the former case, we define xy_{ex} by

$$xy_{ex} := xy_{ik} \cup \{(nc(i),nc(i))\}.$$

In the latter case, we define xy_{ex} by

$$xy_{ex} := xy_{ik} \cup \{(dh(num(j),nc(i)),dh(num(j),nc(i)))\}.$$

In case $ev = send(ag,ag',nc(i),dh(gen(m_1,m_2),nc(i)))$, we proceed according to the same principle.

4. In case $ev = says(ag,ag',mac(dh(m_1,nc(i)),m_1))$, we have $ik_{ev} = \{mac(dh(m_1,nc(i)),m_1)\}$.
We set

$$xy_{ex} = xy_{ik} \cup \{(mac(dh(m_1,nc(i)),m_1),mac(dh(m_1,nc(i)),m_1))\}$$

if $dh(m_1,nc(i)) \notin DY(ik)$, and $xy_{ex} = xy_{ik}$ otherwise.

5. In case $ev = note(spy,dh(dh(gen(m_1,m_2),nc(i)),nc(j)))$, we have $ik_{ev} = \{dh(dh(gen(m_1,m_2),nc(i)),nc(j))\}$ and we distinguish two cases:

(a) If $nc(i),nc(j) \notin DY(ik)$, then we define xy_{ex} by

$$\begin{aligned} xy_{ex} := (xy_{ik} \setminus \{(m,m) \mid m \in ik \wedge \\ m = mac(dh(dh(gen(m_1,m_2),nc(i)),nc(j)),dh(gen(m_1,m_2),nc(i)))\}) \\ \cup \{(dh(dh(gen(m_1,m_2),nc(i)),nc(j)), \\ dh(dh(gen(m_1,m_2),nc(i)),nc(j)))\}. \end{aligned}$$

(b) Otherwise, we set $xy_{ex} = xy_{ik}$.

Note that the pairs in xy_{ik} that include the matching *mac*-messages must be removed, in order not to violate the fourth rule in Φ_5 (while preserving the minimality of xy_{ex}).

6. In the fake case, i.e. $ik_{ev} = \{m\}$ for $m \in DY(ik)$, we set $xy_{ex} = xy_{ik}$.

In all these cases, we prove that $\Phi(xy_{ex}, ik \cup ik_{ev}, \pi, \pi')$ ensues from $\Phi(xy_{ik}, ik, \pi, \pi')$ and the corresponding case context. In particular, we have to show $\Phi(xy_{ik}, ik \cup \{m\}, \pi, \pi')$ in the fake case, which is mainly done with the help of the regularity properties (in Sec. 12.2) and the definition of Φ: We consider the cases where m matches any assumption of the rules Φ_1–Φ_5; Then, we apply in every case a corresponding regularity property permitting to obtain an equivalent message in ik; This implies that each pair that shall be added due to m (from $DY(ik)$) is already in xy_{ik}, as required.

For instance, the first case of Φ_1 provides $m = enc(\pi,nc)$. Using $m \in DY(ik)$ and $\pi \notin DY(ik)$ from the assumption Ω, regularity property 116-(2) permits to obtain $m \in ik$, as required.

Similarly, the second case of Φ_5 provides $m = dh(m_1,m_2)$ together with the conditions $\neg isObj^{dh}(m_1)$ and $m_2 \notin DY(ik)$. Using these conditions and $m \in DY(ik)$, regularity property 117-(2) permits to obtain $m \in ik$, as required.

Case II: In the complementary case, i.e. ik does not include neither a first nor a fourth PACE message using π, we proceed as follows:

1. There is nc with $ik_{ev} = \{enc(\pi,nc)\}$: First, we use a lemma that provides a set xy_{in} containing (π,π') and all pairs obtained according to Φ_2, Φ_3 and Φ_5 relative to ik. Then, we set $xy_{ex} = xy_{in} \cup \{(enc(\pi,nc),enc(\pi,nc)),(nc,dec(\pi',enc(\pi,nc)))\}$
2. There is g,m_1,m_2,nc with $obj^{dec}(dec(\pi,m_1),\pi,m_1)$ and $ik_{ev} = \{dh(gen(dh(g,dec(\pi,m_1)),m_2),nc)\}$: We use the same lemma as in the previous case to obtain a set xy_{in} containing (π,π') and all pairs obtained according to Φ_2, Φ_3 and Φ_5 relative to ik. Then, we set
$$xy_{ex} = xy_{in} \cup \{(dh(gen(dh(g,dec(\pi,m_1)),m_2),nc), dh(gen(dh(g,dec(\pi,m_1)),m_2),nc))\}.$$
3. The complementary case is handled similar to the base case. In the fake case, we need additionally to show that m (from $DY(ik)$) does not match the previous cases (1) and (2). This is done with help of the regularity properties 116-(2) and 117-(2). They permit to deduce $m \in ik$ when m matches a first or a fourth PACE message with a protected occurrence of π. This permits to conclude by refutation, as ik does not include neither a first nor a fourth PACE message using π.

In cases (1) and (2), we prove that $\Phi(xy_{ex}, ik \cup ik_{ev}, \pi, \pi')$ ensues from the applied lemma and the corresponding case context.

12.4 Handling of the Proof Obligations

In this section, we describe how the proof obligations are shown from the assumption Ω, the definition Φ of the basis relations (in Sec. 12.1) and the regularity properties (in Sec. 12.2).

12.4.1 Handling of Ψ^a:

Proof obligation Ψ^a requires that all derivable atomic messages c_i occur in the domain (resp. codomain) of the basis relation xy. It is shown with the help of the regularity properties in 115. They provide for each kind of c_i in $DY(\pi\bullet ik)$ (resp. $DY(\pi'\bullet ik)$) the premises for the inclusion rules of Φ, which in turn imply the occurrence of c_i in the domain (resp. codomain) of xy.

- The case for agent names $ag(j)$ is handled as sketched in the following table:

$ag(j) \in DY(\pi\bullet ik) \stackrel{?}{\vdash} ag(j) \in dom(xy)$ and $ag(j) \in DY(\pi'\bullet ik) \stackrel{?}{\vdash} ag(j) \in codom(xy)$
Property 115-(1) yields one case: $pair(ag(j),pair(pw(j),num(k))) \in ik \stackrel{\Phi_2}{\vdash} (ag(j),ag(j)) \in xy$

- The case for numerical data $num(j)$ is handled as sketched in the following table:

$num(j) \in DY(\pi\bullet ik) \stackrel{?}{\vdash} num(j) \in dom(xy)$ and $num(j) \in DY(\pi'\bullet ik) \stackrel{?}{\vdash} num(j) \in codom(xy)$
Property 115-(2) yields two cases: (1) $pair(ag(k),pair(pw(k),num(j))) \in ik \stackrel{\Phi_2}{\vdash} (num(j),num(j)) \in xy$ (2) $pair(pw(k),num(j)) \in ik \stackrel{\Phi_2}{\vdash} (num(j),num(j)) \in xy$

- The case for passwords $pw(j)$ is handled as sketched in the following table:

$pw(j) \in DY(\pi\bullet ik) \stackrel{?}{\vdash} pw(j) \in dom(xy)$
Property 115-(3) yields three cases: (1) $pw(j) = \pi \stackrel{\Omega,\Phi_1}{\vdash} (pw(j),\pi') \in xy$ (2) $pair(ag(j),pair(pw(j),num(k))) \in ik \stackrel{\Phi_2}{\vdash} (pw(j),pw(j)) \in xy$ (3) $pair(pw(j),num(k)) \in ik \stackrel{\Phi_2}{\vdash} (pw(j),pw(j)) \in xy$
$pw(j) \in DY(\pi'\bullet ik) \stackrel{?}{\vdash} pw(j) \in codom(xy)$
Property 115-(3) yields three cases: (1) $pw(j) = \pi' \stackrel{\Omega,\Phi_1}{\vdash} (\pi,pw(j)) \in xy$ (2) $pair(ag(j),pair(pw(j),num(k))) \in ik \stackrel{\Phi_2}{\vdash} (pw(j),pw(j)) \in xy$ (3) $pair(pw(j),num(k)) \in ik \stackrel{\Phi_2}{\vdash} (pw(j),pw(j)) \in xy$

In case $pw(j) = \pi$ (resp. $pw(j) = \pi'$), the assumption Ω ensures that ik includes the regular messages necessary for the application of Φ_1.

- The case for nonces $nc(j)$ is handled as sketched in the following table:

$nc(j) \in DY(\pi\bullet ik) \stackrel{?}{\vdash} nc(j) \in dom(xy)$
Property 115-(4) yields four cases: (1) $enc(\pi,nc(j)) \in ik \stackrel{\Phi_4}{\vdash} (nc(j),dec(\pi',enc(\pi,nc(j)))) \in xy$ (2) $nc(j) \in ik \stackrel{\Phi_3}{\vdash} (nc(j),nc(j)) \in xy$ (3) $pair(m,nc(j)) \in ik \stackrel{\Phi_3}{\vdash} (nc(j),nc(j)) \in xy$ (4) $enc(pw(k),nc(j)) \in ik, pw(k) \in DY(ik) \stackrel{\Phi_3}{\vdash} (nc(j),nc(j)) \in xy$
$nc(j) \in DY(\pi'\bullet ik) \stackrel{?}{\vdash} nc(j) \in codom(xy)$
Property 115-(4) yields four cases: (1) $enc(\pi',nc(j)) \in ik \stackrel{\Omega}{\vdash} \bot$ (2) $nc(j) \in ik \stackrel{\Phi_3}{\vdash} (nc(j),nc(j)) \in xy$ (3) $pair(m,nc(j)) \in ik \stackrel{\Phi_3}{\vdash} (nc(j),nc(j)) \in xy$ (4) $enc(pw(k),nc(j)) \in ik, pw(k) \in DY(ik) \stackrel{\Phi_3}{\vdash} (nc(j),nc(j)) \in xy$

The case where π' occurs in $enc(\pi',nc(j))$ from ik is closed by refutation based on the assumption Ω, which excludes any occurrence of π' in ik.

12.4.2 Handling of Ψ^c:

According to proof obligation Ψ^c, all derivable composed messages that cannot be derived by composition must occur in the domain (resp. codomain) of the basis relation xy. This is shown with the help of the regularity properties about the derivable composed messages.

The regularity properties about *enc*-, *dh*- and *mac*-messages provide the premises for the inclusion rule Φ_5, which in turn imply the occurrence of these composed messages in the domain (resp. codomain) of xy.

- The proof for $enc(m_1,m_2) \in dom(xy)$ with the help of the regularity properties in 116 is sketched in the following table:

$enc(m_1,m_2) \in DY(\pi\bullet ik), \neg(m_1,m_2 \in DY(\pi\bullet ik)) \stackrel{?}{\vdash} enc(m_1,m_2) \in dom(xy)$
Property 116-(1) yields: $enc(m_1,m_2) \in DY(ik), \neg(m_1,m_2 \in DY(ik)) \stackrel{?}{\vdash} enc(m_1,m_2) \in dom(xy)$ Property 116-(2) yields: $m_1 = pw(i), m_2 = nc(j), enc(m_1,m_2) \in ik, m_1 \notin DY(ik) \stackrel{\Phi_5}{\vdash}$ $(enc(m_1,m_2),enc(m_1,m_2)) \in xy$

- The proof for $dh(m_1,m_2) \in codom(xy)$ with the help of the regularity properties in 117 is sketched in the following table:

$dh(m_1,m_2) \in DY(\pi'.ik), \Theta_1^{dh}(dh(m_1,m_2),\pi'.ik) \vdash^? dh(m_1,m_2) \in codom(xy)$
Property 117-(1) yields:
$dh(m_1,m_2) \in DY(ik), \Theta_1^{dh}(dh(m_1,m_2),\pi'.ik) \vdash^? dh(m_1,m_2) \in codom(xy)$
Property 117-(2) yields two cases:
(1) $dh(m_1,m_2) \in ik, \neg isObj^{dh}(m_1), m_2 \notin DY(pw(i).ik)$
$\vdash^{\Phi_5} (dh(m_1,m_2), dh(m_1,m_2)) \in xy$
(2) $dh(m_1,m_2) \in ik, m_1 = dh(m_3,m_4), \neg isObj^{dh}(m_3), (m_2,m_4 \notin DY(pw(i).ik))$
$\vdash^{\Phi_5} (dh(m_1,m_2), dh(m_1,m_2)) \in xy$

Recall, $\Theta_1^{dh}(dh(m_1,m_2),\pi'.ik)$ means that $dh(m_1,m_2)$ does not possess any pair of dh-parts in $DY(\pi'.ik)$.

- The proof for $mac(m_1,m_2) \in dom(xy)$ with the help of the regularity properties in 118 is sketched in the following table:

$mac(m_1,m_2) \in DY(\pi.ik), \neg(m_1,m_2 \in DY(\pi.ik)) \vdash^? mac(m_1,m_2) \in dom(xy)$
Property 118-(1) yields:
$mac(m_1,m_2) \in DY(ik), \neg(m_1,m_2 \in DY(\pi.ik)) \vdash^? mac(m_1,m_2) \in dom(xy)$
Property 118-(2) yields:
$mac(m_1,m_2) \in ik, m_1 \notin DY(pw(i).ik) \vdash^? mac(m_1,m_2) \in dom(xy)$
$DY(ik) \subseteq DY(pw(i).ik)$ yields:
$mac(m_1,m_2) \in ik, m_1 \notin DY(ik) \vdash^{\Phi_5} (mac(m_1,m_2), mac(m_1,m_2)) \in xy$

The regularity properties about dec-, fst-, snd- and gen-messages, which do not possess confidential dec-, fst-, snd- and respectively gen-parts, permit to handle the corresponding cases in the proof of Ψ^c by refutation. For instance, the proof for $gen(m_1,m_2) \notin dom(xy)$ with the help of the regularity property 119-(4) is sketched in the following table:

$gen(m_1,m_2) \in DY(\pi.ik), \neg(m_1,m_2 \in DY(\pi.ik)) \vdash^? gen(m_1,m_2) \in dom(xy)$
Property 119-(4) yields:
$(m_1,m_2 \in DY(\pi.ik)), \neg(m_1,m_2 \in DY(\pi.ik)) \vdash^{\perp} \perp$

12.4.3 Handling of Ψ^b:

The proof of Ψ^b consists in showing $xy \subset DY(\pi.ik) \times DY(\pi'.ik)$ and that the restriction of xy on its domain and codomain is bijective. The former trivially ensues from the definition of Φ. The latter proof task is handled as follows:

- According to Φ, all pairs in xy match (m,m), $(nc,dec(\pi',enc(\pi,nc)))$ or (π,π').
- First, we show $(m,m_y),(m,m_z) \in xy \Rightarrow m_y = m_z$ by contradiction. For that purpose, we assume $(m,m_y),(m,m_z) \in xy$ with $m_y \neq m_z$, which yields three cases closed by refutation as follows:
  1. $(m,m_y) = (m,m)$ and $(m,m_z) = (nc,dec(\pi',enc(\pi,nc)))$: This yields $m_y = m$, $m = nc$ and $m_z = dec(\pi',enc(\pi,nc))$, and thus $(nc,nc),(nc,dec(\pi',enc(\pi,nc))) \in xy$. On the one side, $(nc,nc) \in xy$ and Φ_3 imply $nc \in DY(ik)$. On the other side, $(nc,dec(\pi',enc(\pi,nc))) \in xy$, Φ_4 and Ω imply $nc \notin DY(ik)$.
  2. $(m,m_y) = (m,m)$ and $(m,m_z) = (\pi,\pi')$: This yields $m_y = m$, $m = \pi$ and $m_z = \pi'$, and thus $(\pi,\pi),(\pi,\pi') \in xy$. On the one side, $(\pi,\pi) \in xy$ and Φ_2 imply $\pi \in DY(ik)$. On the other side, $(\pi,\pi') \in xy$, Φ_1 and Ω imply $\pi \notin DY(ik)$.
  3. $(m,m_y) = (nc,dec(\pi',enc(\pi,nc)))$ and $(m,m_z) = (\pi,\pi')$: This yields $m = nc$ and $m = \pi$, and thus $nc = \pi$. But, nonces differ from passwords.

- Next, we show $(m_y,m),(m_z,m) \in xy \Rightarrow m_y = m_z$ by contradiction. For that purpose, we assume $(m_y,m),(m_z,m) \in xy$ with $m_y \neq m_z$, which yields three cases closed by refutation as follows:

  1. $(m_y,m) = (m,m)$ and $(m_z,m) = (nc,dec(\pi',enc(\pi,nc)))$: This case yields the equalities $m_y = m$, $m = dec(\pi',enc(\pi,nc))$ and $m_z = nc$, and thus we obtain the pair $(dec(\pi',enc(\pi,nc)),dec(\pi',enc(\pi,nc))) \in xy$. But, Φ excludes any pair (m,m) in xy where m is a *dec*-object.
  2. $(m_y,m) = (m,m)$ and $(m_z,m) = (\pi,\pi')$: This yields $m_y = m$, $m = \pi'$ and $m_z = \pi$, and thus $(\pi',\pi') \in xy$. But, Ω and Φ exclude any pair (π',π') in xy.
  3. $(m_y,m) = (nc,dec(\pi',enc(\pi,nc)))$ and $(m_z,m) = (\pi,\pi')$: This case yields the equalities $m = dec(\pi',enc(\pi,nc))$ and $m = \pi'$, and thus $dec(\pi',enc(\pi,nc)) = \pi'$. But, *dec*-objects differ from passwords.

12.4.4 Handling of Ψ^2_{dec} and Ψ^1_{enc}:

The remaining conditions of Ψ are inclusion rules where the premises require a composed message in the domain or the codomain of xy that is derivable by composition. According to Φ, this requirement holds for *dec*-messages in the codomain of xy and for certain *enc*-messages in the domain of xy.

The corresponding proof obligation Ψ^2_{dec} for *dec*-messages is shown with the help of Φ and Ω, as sketched in the following table:

$(m_x,dec(m_1,m_2)) \in xy, m_1 \in DY(\pi'.ik) \stackrel{?}{\vdash} (m_{x1},m_1),(enc(m_{x1},m_x),m_2) \in xy$
Φ yields:
$m_1 = \pi', m_2 = enc(\pi,nc(j)), m_x = nc(j), enc(\pi,nc(j)) \in ik, \pi' \in DY(\pi'.ik)$
$\stackrel{?}{\vdash} (m_{x1},\pi'),(enc(m_{x1},nc(j)),enc(\pi,nc(j))) \in xy$
Setting $m_{x1} = \pi$, yields:
$enc(\pi,nc(j)) \in ik \vdash^{\Omega,\Phi_1,\Phi_5} (\pi,\pi'),(enc(\pi,nc(j)),enc(\pi,nc(j))) \in xy$

The assumptions of Ψ^2_{dec} and Φ imply $enc(\pi,nc(j)) \in ik$. This permits to set $m_{x1} = \pi$ and then prove the required pairs in xy based on Ω,Φ_1 and Φ_5.

The proof obligation Ψ^1_{enc} for *enc*-messages is shown with the help of Φ and property 115-(3), as sketched in the following table:

$(enc(m_1,m_2),m_y) \in xy, m_1 \in DY(\pi.ik) \stackrel{?}{\vdash} (m_1,m_{y1}),(m_2,dec(m_{y1},m_y)) \in xy$
Φ yields:
$m_1 = pw(i), m_2 = nc(j), m_y = enc(pw(i),nc(j)), m_y \in ik, pw(i) \notin DY(ik),$
$pw(i) \in DY(\pi.ik) \stackrel{?}{\vdash} (pw(i),m_{y1}),(nc(j),dec(m_{y1},enc(pw(i),nc(j)))) \in xy$
Property 115-(3) yields:
$\pi = pw(i), enc(\pi,nc(j)) \in ik \stackrel{?}{\vdash} (\pi,m_{y1}),(nc(j),dec(m_{y1},enc(\pi,nc(j)))) \in xy$
$enc(\pi,nc(j)) \in ik \vdash^{\Phi_1,\Phi_4} (\pi,\pi'),(nc(j),dec(\pi',enc(\pi,nc(j)))) \in xy$

Given an arbitrary $(enc(m_1,m_2),m_y) \in xy$, the definition of Φ binds $enc(m_1,m_2)$ and m_y to a message $enc(pw(i),nc(j)) \in ik$ with $pw(i) \notin DY(ik)$. Then, $m_1 \in DY(\pi.ik)$ rewrites to $pw(i) \in DY(\pi.ik)$ and this permits to use property 115-(3) and bind $pw(i)$ to π, due to $pw(i) \notin DY(ik)$. Hence, $enc(pw(i),nc(j)) \in ik$ rewrites to $enc(\pi,nc(j)) \in ik$, used as premise for Φ_1 and Φ_4 to obtain the required pairs in xy.

12.4.5 Proof Obligations Handled by Refutation:

Except of Ψ^a, Ψ^c, Ψ^b, Ψ^1_{enc} and Ψ^2_{dec}, all other conditions of Ψ are handled by refutation. They correspond to inclusion rules where the premises require a composed message in the

domain or the codomain of xy that is derivable by composition. Except of *enc*-messages in the domain of xy and *dec*-messages in the codomain of xy, Φ ensures for all other occurrences of composed messages that these cannot be derived by composition. Furthermore, Φ restricts the occurrences of composed messages to *enc*-, *dh*-, and *mac*-objects both in the domain and codomain of xy and to *dec*-messages only in the codomain of xy.

Consequently, the proof obligations Ψ^i_{fst}, Ψ^i_{snd} and Ψ^i_{gen} for $i \in \{1,2\}$, together with Ψ^1_{dec} are shown immediately by Φ, which excludes pairs in xy with a corresponding occurrence of *fst*-, *snd*-, *gen*- and respectively *dec*-objects. Contrarily, the refutation proofs of Ψ^i_{dh}, $\Psi^{i,2}_{dh}$ and Ψ^i_{mac} for $i \in \{1,2\}$, together with Ψ^2_{enc}, are based not only on Φ but also on Ω and corresponding regularity properties.

The proof of Ψ^1_{dh} is based on Φ and the regularity property 117-(2), as sketched in the following table:

$(dh(m_1,m_2),m_y) \in xy, (m_1,m_2 \in DY(\pi\bullet ik))$ $\vdash^? (m_1,m_{y1}),(m_2,m_{y2}) \in xy \wedge m_y = dh(m_{y1},m_{y2})$
Φ yields two cases: (1) $\neg isObj^{dh}(m_1), m_y = dh(m_1,m_2), m_y \in ik, m_2 \notin DY(ik), (m_1,m_2 \in DY(\pi\bullet ik))$ $\vdash^? (m_1,m_{y1}),(m_2,m_{y2}) \in xy \wedge dh(m_1,m_2) = dh(m_{y1},m_{y2})$ (2) $m_1 = dh(m_3,m_4), \neg isObj^{dh}(m_3), m_y = dh(m_1,m_2), m_y \in ik,$ $(m_4,m_2 \notin DY(ik)), (m_1,m_2 \in DY(\pi\bullet ik)) \vdash^?$ $(m_1,m_{y1}),(m_2,m_{y2}) \in xy \wedge dh(m_1,m_2) = dh(m_{y1},m_{y2})$ Property 117-(2) yields: (1) $dh(m_1,m_2) \in ik, \neg isObj^{dh}(m_1), m_2 \notin DY(pw(k)\bullet ik), (m_1,m_2 \in DY(\pi\bullet ik)) \vdash^! \bot$ (2) $dh(dh(m_3,m_4),m_2) \in ik, \neg isObj^{dh}(m_3), (m_4,m_2 \notin DY(pw(k)\bullet ik)),$ $(m_1,m_2 \in DY(\pi\bullet ik)) \vdash^! \bot$

Given an arbitrary $(dh(m_1,m_2),m_y) \in xy$, the definition of Φ binds $dh(m_1,m_2)$ and m_y to *dh*-messages in ik having either unique right *dh*-parts or exactly two permutable right *dh*-parts. These right *dh*-parts do not belong to $DY(ik)$, which permits to deduce by property 117-(2) that they cannot occur in $DY(pw(k)\bullet ik)$ for all passwords $pw(k)$. Hence, this allows us to conclude by refuting the assumption $m_2 \in DY(\pi\bullet ik)$, as m_2 matches one of these protected right *dh*-parts.

The proof obligations Ψ^2_{dh}, $\Psi^{1,2}_{dh}$ and $\Psi^{2,2}_{dh}$ are shown similar to Ψ^1_{dh}, by refutation after the use of Φ and property 117-(2). The common proof argument relies on the protection of the right *dh*-parts in all regular PACE messages. If they are not compromised, there is no way for the attacker to derive them also when ik is extended with a genuine password.

The proof of Ψ^1_{mac} is based on Φ and the regularity property 118-(2), as sketched in the following table:

$(mac(m_1,m_2),m_y) \in xy, (m_1,m_2 \in DY(\pi\bullet ik))$ $\vdash^? (m_1,m_{y1}),(m_2,m_{y2}) \in xy \wedge m_y = mac(m_{y1},m_{y2})$
Φ yields: $m_y = mac(m_1,m_2), m_y \in ik, m_1 \notin DY(ik), (m_1,m_2 \in DY(\pi\bullet ik))$ $\vdash^? (m_1,m_{y1}),(m_2,m_{y2}) \in xy \wedge mac(m_1,m_2) = mac(m_{y1},m_{y2})$ Property 118-(2) yields: $mac(m_1,m_2) \in ik, m_1 \notin DY(pw(k)\bullet ik), (m_1,m_2 \in DY(\pi\bullet ik)) \vdash^! \bot$

Given an arbitrary $(mac(m_1,m_2),m_y) \in xy$, the definition of Φ binds $mac(m_1,m_2)$ and m_y to *mac*-messages in ik having confidential first *mac*-part m_1. This permits to deduce by property 118-(2) that m_1 cannot occur in $DY(pw(k)\bullet ik)$ for all passwords $pw(k)$. Hence, this allows us to conclude by refuting the assumption $m_1 \in DY(\pi\bullet ik)$.

The proof obligation Ψ^2_{mac} is shown similar to Ψ^1_{mac}.

Finally, the proof of Ψ^2_{enc} is based on Φ, Ω and the regularity property 115-(3), as sketched in the following table:

$(m_x,enc(m_1,m_2)) \in xy, m_1 \in DY(\pi'.ik) \stackrel{?}{\vdash} (m_{x1},m_1),(dec(m_{x1},m_x),m_2) \in xy$
Φ yields:
$m_1 = pw(i), m_2 = nc(j), m_x = enc(pw(i),nc(j)), m_x \in ik, pw(i) \notin DY(ik),$
$pw(i) \in DY(\pi'.ik) \stackrel{?}{\vdash} (m_{x1},pw(i)),(dec(m_{x1},enc(pw(i),nc(j))),nc(j)) \in xy$
Property 115-(3) yields:
$\pi' = pw(i), enc(\pi',nc(j)) \in ik \vdash^{\Omega} \bot$

The assumptions of Ψ^2_{enc} are reduced using the definition of Φ and property 115-(3) to $enc(\pi',nc(j)) \in ik$. This permits to conclude by refutation based on the assumption Ω, which excludes any occurrence of π' in ik.

12.4.6 Handling of Γ:

Proof obligation Γ requires that every element in $kb = \pi.ik$ is mapped by $\stackrel{xy}{\rightsquigarrow}$ to the element in $kb' = \pi'.ik$ at the same position. This holds obviously for the first elements π and π', as Φ ensures $(\pi,\pi') \in xy$. For the remaining elements, the proof is by the following lemma:

Property$^{\mathbf{VSE}}$ 120 (Γ Property of $\stackrel{xy}{\rightsquigarrow}$):

$$
\begin{aligned}
&(tr \in \text{PACE} \wedge \pi \notin DY(spies(tr)) \wedge (\forall m \in spies(tr) : \neg uses(m,\pi')) \wedge \\
&\ ((\exists nc : enc(\pi,nc) \in spies(tr)) \vee \\
&\ \ (\exists g,m_1,m_2,nc : obj^{dec}(dec(\pi,m_1),\pi,m_1) \wedge \\
&\ \ \ dh(gen(dh(g,dec(\pi,m_1)),m_2),nc) \in spies(tr))) \wedge \\
&\ \Phi(xy,spies(tr),\pi,\pi') \wedge m \in DY(spies(tr)))) \\
&\Rightarrow m \stackrel{xy}{\rightsquigarrow} m
\end{aligned}
$$

This lemma is shown by induction on $|m|$.

Base Case: For $m = c_i$ and $c_i \in DY(ik)$, lemma 109 provides $c_i \stackrel{xy}{\leftrightarrow} m'$ with $(c_i,m') \in xy$. According to Φ, we distinguish mainly three cases:

- $(c_i,m') = (\pi,\pi')$: This yields $c_i = \pi$ and thus $\pi \in DY(ik)$, which permits to conclude by refutation based on Ω.
- $(c_i,m') = (nc,dec(\pi',enc(\pi,nc)))$, for $enc(\pi,nc) \in ik$: This yields $c_i = nc$ and consequently $nc \in DY(ik)$, which permits to conclude by refutation based on the basic confidentiality property of PACE (If π is not compromised, then nc is confidential, i.e. $\pi \notin DY(ik) \Rightarrow nc \notin DY(ik)$).
- (c_i,m') is of the form $(ag(j),ag(j))$, $(num(j),num(j))$, $(nc(j),nc(j))$ or $(pw(j),pw(j))$: This yields $m' = c_i$, as required.

Step Case: For an arbitrary composed message $m \in DY(ik)$, we show $m \overset{xy}{\rightsquigarrow} m$ simply by Φ when $m \in dom(xy)$: According to Φ, xy includes only pairs of the form (π,π'), $(nc,dec(\pi',enc(\pi,nc)))$ and (m',m'). As a *composed* message, m occurs only in a pair (m',m') with $m' = m$, which implies $m \overset{xy}{\leftrightarrow} m$, as required.

In the complementary case, Ψ^c ensures that m can be composed from sub-messages in $DY(\pi\bullet ik)$. We want to show (in the following case distinction) that these sub-messages belong to $DY(ik)$, in order to obtain $m \overset{xy}{\rightsquigarrow} m$ by the induction hypothesis.

- In case $obj^{enc}(m,m_1,m_2)$, the regularity property 116-(2) yields $m_1,m_2 \in DY(ik)$ or $m = enc(pw(i),nc(j))$ and $enc(pw(i),nc(j)) \in ik$. Using $enc(pw(i),nc(j)) \notin dom(xy)$ and Φ, we show that $m_1,m_2 \in DY(ik)$ holds in both cases: For $enc(pw(i),nc(j)) \in ik$ and $enc(pw(i),nc(j)) \notin dom(xy)$, it follows $pw(i) \in DY(ik)$ by Φ and thus $pw(i),nc(j) \in DY(ik)$, i.e. $m_1,m_2 \in DY(ik)$.
- In case $m = dh(m_1,m_2)$, the regularity property 117-(2) provides $m_1',m_2' \in DY(ik)$ with $m = dh(m_1',m_2')$, because otherwise m must belong to $dom(xy)$.
- In case $m = mac(m_1,m_2)$, the regularity property 118-(2) implies $m_1,m_2 \in DY(ik)$, because otherwise m must belong to $dom(xy)$.
- In case $obj^{dec}(m,m_1,m_2)$, the regularity property 119-(2) yields $m_1,m_2 \in DY(ik)$.
- All other cases are similar to the previous case, employing corresponding regularity properties in 119.

That is, we have in all cases sub-messages $m_1,m_2 \in DY(ik)$ (or $m_1 \in DY(ik)$ for fst- and snd-messages), satisfying $m_1 \overset{xy}{\rightsquigarrow} m_1$ and $m_2 \overset{xy}{\rightsquigarrow} m_2$ based on the induction hypothesis. Hence, the definition of $\overrightarrow{rec}$ permits to obtain a mapping $m \overset{xy}{\rightsquigarrow} m$, as required.

Chapter 13

The Central Indistinguishability Theorem in TC-AMP

In this chapter, we present the central indistinguishability theorem in the TC-AMP algebra. We recall the proof plan for indistinguishability properties (in Sec. 13.1) and introduce the algebra-specific conditions (in Sec. 13.2). After that, we describe the handling of the structural mapping lemma (the canonical cases in Sec. 13.3 and the non-canonical cases in Sec. 13.4– 13.6) and then the proof of the central indistinguishability theorem (in Sec. 13.7).

13.1 The Building Blocks of Indistinguishability Proofs

The proof plan for indistinguishability properties in Sec. 11.1 applies also to the TC-AMP algebra. Similarly, we use a recursive function $\widetilde{rec}$ for the definition of simulation relations $\rightsquigarrow$ relative to finite sets xy of message pairs. The following definition of $\widetilde{rec}$ is obtained by simple adaptations of Def. 106.

Definition[VSE] **121 ($\widetilde{rec}$; TC-AMP):**
Let xy be a finite set of message pairs. Then, we have for two arbitrary messages m and m':

$$
\begin{array}{l}
m' \in \widetilde{rec}\,(xy,m) \Leftrightarrow \\
((m,m') \in xy \vee \\
\quad ((\forall z : (m,z) \notin xy) \wedge \\
\quad\quad ((\exists m_0,m_0' : m_0' \in \widetilde{rec}\,(xy,m_0) \wedge \\
((obj^{fst}(m,m_0) \wedge m' = fst(m_0')) \vee (obj^{snd}(m,m_0) \wedge m' = snd(m_0')) \vee \\
(obj^{inv}(m,m_0) \wedge m' = inv(m_0')) \vee ((obj^{\ominus}(m,m_0) \vee syn^{\ominus}_{\oplus}(m,m_0)) \wedge m' = \ominus(m_0')))) \vee \\
\quad\quad\quad (\exists m_0,m_1,m_0',m_1' : m_0' \in \widetilde{rec}\,(xy,m_0) \wedge m_1' \in \widetilde{rec}\,(xy,m_1) \wedge \\
((obj^{pair}(m,m_0,m_1) \wedge m' = pair(m_0',m_1')) \vee (obj^{\oplus}(m,m_0,m_1) \wedge m' = \oplus(m_0',m_1')) \vee \\
((obj^{*}(m,m_0,m_1) \vee syn^{*}_{\ominus}(m,m_0,m_1) \vee syn^{*}_{\oplus}(m,m_0,m_1)) \wedge m' = *(m_0',m_1')))) \vee \\
\quad\quad\quad (\exists m_0,m_1,m_2,m_0',m_1',m_2' : m_0' \in \widetilde{rec}\,(xy,m_0) \wedge m_1' \in \widetilde{rec}\,(xy,m_1) \wedge m_2' \in \widetilde{rec}\,(xy,m_2) \wedge \\
((obj^{h_1}(m,m_0,m_1,m_2) \wedge m' = h_1(m_0',m_1',m_2')) \vee \\
(obj^{h_2}(m,m_0,m_1,m_2) \wedge m' = h_2(m_0',m_1',m_2'))))))).
\end{array}
$$

The recursive case of $\widetilde{rec}$ considers for $*$-objects m implicitly *several* alternatives of submessages, as they are composed by constructor-type applications of the *permutative* func-

tion symbol $*$. In case of $\ominus$-objects m, two composition alternatives are considered, i.e. by constructor-type application of $\ominus$ and by the synthesis operation $syn^*_\ominus$. For $\oplus$-objects m, we consider *several* composition alternatives by constructor-type applications of the *permutative* function symbol $\oplus$, and by synthesis operations $syn^*_\oplus$ and $syn^\ominus_\oplus$. For that reason, the structural mapping lemma includes three non-canonical cases for the handling of these composition operations (see Sec. 13.4, 13.5 and 13.6).

For instance, in case of a $\oplus$-object $m = \oplus(*(a,b_1),*(a,b_2))$ with $m \notin dom(xy)$, the function $\widetilde{rec}$ is invoked recursively for the $\oplus$-parts $(*(a,b_1),*(a,b_2))$ and $(*(a,b_2),*(a,b_1))$, corresponding to two $obj^\oplus$-cases, and for the $*$-sub-messages $(a,\oplus(b_1,b_2))$, corresponding to one $syn^*_\oplus$-case.

The function $\widetilde{rec}$ is used to extend given basis relations xy to simulation relations $\rightsquigarrow$. For $\rightsquigarrow$ to be total, the used finite sets xy of message pairs need to satisfy the following conditions, obtained by simple adaptations of Ψ^a and Ψ^c in Sec. 11.1.1.2 to the TC-AMP algebra.

$$\begin{aligned}
\Psi^a: \quad & ((m \in At \cup \Sigma\langle 0\rangle \wedge m \in DY(kb)) \Rightarrow (\exists m' : (m,m') \in xy)) \wedge \\
& ((m \in At \cup \Sigma\langle 0\rangle \wedge m \in DY(kb')) \Rightarrow (\exists m' : (m',m) \in xy))
\end{aligned}$$

$$\begin{aligned}
\Psi^c: \quad & ((m \notin At \cup \Sigma\langle 0\rangle \wedge m \in DY(kb) \wedge \neg isObj^{pair}(m) \wedge \neg isObj^{inv}(m) \wedge \neg isObj^{\ominus}(m) \\
& \quad \wedge (\forall m_0 : \neg syn^\ominus_\oplus(m,m_0)) \wedge ((obj^{fst}(m,m_0) \vee obj^{snd}(m,m_0)) \Rightarrow m_0 \notin DY(kb)) \\
& \quad \wedge ((obj^\oplus(m,m_0,m_1) \vee obj^*(m,m_0,m_1) \vee \\
& \qquad syn^*_\ominus(m,m_0,m_1) \vee syn^*_\oplus(m,m_0,m_1)) \Rightarrow \neg(m_0,m_1 \in DY(kb))) \\
& \quad \wedge ((obj^{h_1}(m,m_0,m_1,m_2) \vee obj^{h_2}(m,m_0,m_1,m_2)) \Rightarrow \neg(m_0,m_1,m_2 \in DY(kb)))) \\
& \quad \Rightarrow (\exists m' : (m,m') \in xy)) \wedge \\
& ((m \notin At \cup \Sigma\langle 0\rangle \wedge m \in DY(kb') \wedge \neg isObj^{pair}(m) \wedge \neg isObj^{inv}(m) \wedge \neg isObj^{\ominus}(m) \\
& \quad \wedge (\forall m_0 : \neg syn^\ominus_\oplus(m,m_0)) \wedge ((obj^{fst}(m,m_0) \vee obj^{snd}(m,m_0)) \Rightarrow m_0 \notin DY(kb')) \\
& \quad \wedge ((obj^\oplus(m,m_0,m_1) \vee obj^*(m,m_0,m_1) \vee \\
& \qquad syn^*_\ominus(m,m_0,m_1) \vee syn^*_\oplus(m,m_0,m_1)) \Rightarrow \neg(m_0,m_1 \in DY(kb'))) \\
& \quad \wedge ((obj^{h_1}(m,m_0,m_1,m_2) \vee obj^{h_2}(m,m_0,m_1,m_2)) \Rightarrow \neg(m_0,m_1,m_2 \in DY(kb')))) \\
& \quad \Rightarrow (\exists m' : (m',m) \in xy))
\end{aligned}$$

As defined in Sec. 11.1, we prove indistinguishability properties by

1. providing a definition $\Phi(xy,ik,\overline{x})$ of the basis relations xy being the smallest sets that satisfy protocol- and property-specific inclusion rules,
2. showing (in the basis relation lemma) that the property contexts $\Omega(tr,\overline{x},kb,kb')$ imply the existence of xy satisfying $\Phi(xy,ik,\overline{x})$,
3. applying the central indistinguishability theorem (similar to) 108 to reduce the required properties of $\rightsquigarrow$ to proof obligations,
4. and proving the proof obligations using regularity properties on derivable composed messages and the protection of their sub-messages.

In the proof of the central indistinguishability theorem, we apply the domain restriction lemma (similar to) 110 and the following structural mapping lemma, which is obtained by slight adaptations of its pendant 109 in PACE:

Lemma$^{\text{VSE}}$ **122 (**$DY(kb) \overset{xy}{\leftrightarrow} DY(kb')$**; TC-AMP):**

$$
\begin{aligned}
&(\Psi(xy,kb,kb') \wedge m \in DY(kb)) \Rightarrow \\
&(\exists m' \in DY(kb') : m \overset{xy}{\leftrightarrow} m' \wedge ((m,m') \in xy \vee \\
&\quad \Xi_{inv}(m,m',xy,kb,kb') \vee \Xi_{fst}(m,m',xy,kb,kb') \vee \\
&\quad \Xi_{snd}(m,m',xy,kb,kb') \vee \Xi_{pair}(m,m',xy,kb,kb') \vee \\
&\quad \Xi_{h_1}(m,m',xy,kb,kb') \vee \Xi_{h_2}(m,m',xy,kb,kb') \vee \\
&\quad \Xi_{\ominus}(m,m',xy,kb,kb') \vee \Xi_{*}(m,m',xy,kb,kb') \vee \\
&\quad \Xi_{\oplus}(m,m',xy,kb,kb')))
\end{aligned}
$$

Besides six canonical cases given by Ξ_{inv}–Ξ_{h_2}, we distinguish three non-canonical cases (the $\Xi_{\ominus}$-, Ξ_{*}- and the $\Xi_{\oplus}$-case) for the handling of f-objects that can be composed by $\ominus$, $*$ and/or $\oplus$. The existence of arbitrary many decomposition alternatives for these f-objects necessitates a non-canonical definition for the used predicates $\Xi_{\ominus}$, Ξ_{*} and $\Xi_{\oplus}$ (see Sec. 13.4–13.6).

Before we describe the proof of the structural mapping lemma 122 (the handling of the canonical cases in Sec. 13.3 and of the non-canonical cases in Sec. 13.4–13.6), we introduce the algebra-specific necessary conditions.

13.2 Algebra-specific Conditions

In this section, we present the algebra-specific conditions for indistinguishability in the TC-AMP algebra.

The equational theory on *pair*, *fst* and *snd* is the same as in PACE, which yields to the same conditions Ψ^1_{pair}, Ψ^2_{pair}, Ψ^1_{fst}, Ψ^2_{fst}, Ψ^1_{snd} and Ψ^2_{snd}.

The additional conditions are generated based on the general principles in Sec. 11.1.2.1. Recall that the first principle is about the mapping of the items involved in a constructor-type application of function symbols f. It is generalized below to synthesis operations. The second principle is about the use of additional conditions to capture the allowed algebraic effects of f and exclude the critical ones.

We start with the most simple conditions, which are similar to Ψ^1_{mac} and Ψ^2_{mac} in PACE.

$$
\begin{aligned}
\Psi^1_{h_1}: \quad & ((h_1(m_{x1},m_{x2},m_{x3}),m_y) \in xy \wedge m_{x1},m_{x2},m_{x3} \in DY(kb)) \Rightarrow \\
& (\exists m_{y1},m_{y2},m_{y3} \in DY(kb') : (m_{x1},m_{y1}),(m_{x2},m_{y2}),(m_{x3},m_{y3}) \in xy \\
& \quad \wedge m_y = h_1(m_{y1},m_{y2},m_{y3}))
\end{aligned}
$$

$$
\begin{aligned}
\Psi^2_{h_1}: \quad & ((m_x,h_1(m_{y1},m_{y2},m_{y3})) \in xy \wedge m_{y1},m_{y2},m_{y3} \in DY(kb')) \Rightarrow \\
& (\exists m_{x1},m_{x2},m_{x3} \in DY(kb) : (m_{x1},m_{y1}),(m_{x2},m_{y2}),(m_{x3},m_{y3}) \in xy \\
& \quad \wedge m_x = h_1(m_{x1},m_{x2},m_{x3}))
\end{aligned}
$$

$$
\begin{aligned}
\Psi^1_{h_2}: \quad & ((h_2(m_{x1},m_{x2},m_{x3}),m_y) \in xy \wedge m_{x1},m_{x2},m_{x3} \in DY(kb)) \Rightarrow \\
& (\exists m_{y1},m_{y2},m_{y3} \in DY(kb') : (m_{x1},m_{y1}),(m_{x2},m_{y2}),(m_{x3},m_{y3}) \in xy \\
& \quad \wedge m_y = h_2(m_{y1},m_{y2},m_{y3}))
\end{aligned}
$$

$$\Psi^2_{h_2}: \quad ((m_x, h_2(m_{y1}, m_{y2}, m_{y3})) \in xy \wedge m_{y1}, m_{y2}, m_{y3} \in DY(kb')) \Rightarrow$$
$$(\exists m_{x1}, m_{x2}, m_{x3} \in DY(kb) : (m_{x1}, m_{y1}), (m_{x2}, m_{y2}), (m_{x3}, m_{y3}) \in xy$$
$$\wedge m_x = h_2(m_{x1}, m_{x2}, m_{x3}))$$

The next conditions are specific to function symbols f that are self-reversing.

$$\Psi^1_{inv}: \quad (obj^{inv}(m_x, m_{x1}) \wedge (m_x, m_y) \in xy) \Rightarrow$$
$$(\exists m_{y1} \in DY(kb') : (m_{x1}, m_{y1}) \in xy \wedge m_y = inv(m_{y1}))$$

$$\Psi^2_{inv}: \quad (obj^{inv}(m_y, m_{y1}) \wedge (m_x, m_y) \in xy) \Rightarrow$$
$$(\exists m_{x1} \in DY(kb) : (m_{x1}, m_{y1}) \in xy \wedge m_x = inv(m_{x1}))$$

$$\Psi^1_{\ominus}: \quad (isObj^{\ominus}(m_x) \wedge (m_x, m_y) \in xy) \Rightarrow (\ominus(m_x), \ominus(m_y)) \in xy$$

$$\Psi^2_{\ominus}: \quad (isObj^{\ominus}(m_y) \wedge (m_x, m_y) \in xy) \Rightarrow (\ominus(m_x), \ominus(m_y)) \in xy$$

$$\Psi^1_{\ominus,*}: \quad (isObj^{*}(m_x) \wedge (m_x, m_y) \in xy) \Rightarrow (\ominus(m_x), \ominus(m_y)) \in xy$$

$$\Psi^2_{\ominus,*}: \quad (isObj^{*}(m_y) \wedge (m_x, m_y) \in xy) \Rightarrow (\ominus(m_x), \ominus(m_y)) \in xy$$

$$\Psi^1_{\ominus,\oplus}: \quad (isObj^{\oplus}(m_x) \wedge (m_x, m_y) \in xy) \Rightarrow (\ominus(m_x), \ominus(m_y)) \in xy$$

$$\Psi^2_{\ominus,\oplus}: \quad (isObj^{\oplus}(m_y) \wedge (m_x, m_y) \in xy) \Rightarrow (\ominus(m_x), \ominus(m_y)) \in xy$$

These conditions ensure that if an application of inv (resp. $\ominus$) reverses a previous application on one side of a mapping, the same effect happens on the other side. That is, the image of an inv-object can be obtained by a constructor-type or a selector-type application of inv to the image of its inv-part. The images of a $\ominus$-object and its $\ominus$-part are related equally by an application of $\ominus$. Similarly, the images of a $*$-object (resp. of a $\oplus$-object) and its pendant resulting by an application of $\ominus$ are related equally by an application of $\ominus$.

The following conditions restrict the allowed mappings of constants that are derivable by function symbols independent of used items.

$$\Psi^1_{\infty}: \quad (\infty, m_y) \in xy \Rightarrow m_y = \infty$$

$$\Psi^2_{\infty}: \quad (m_x, \infty) \in xy \Rightarrow m_x = \infty$$

The constant ∞ must be mapped to itself.

The following conditions integrate allowed reversing effects of $*$ and $\oplus$.

$$\Psi^1_{*}: \quad (syn^{*}(m_x, m_{x1}, m_{x2}) \wedge (m_x, m_y) \in xy \wedge m_{x1} \in DY(kb)) \Rightarrow$$
$$(\exists m_{y1} \in DY(kb') : (m_{x1}, m_{y1}), (m_{x2}, *(inv(m_{y1}), m_y)) \in xy)$$

$$\Psi^2_{*}: \quad (syn^{*}(m_y, m_{y1}, m_{y2}) \wedge (m_x, m_y) \in xy \wedge m_{y1} \in DY(kb')) \Rightarrow$$
$$(\exists m_{x1} \in DY(kb) : (m_{x1}, m_{y1}), (*(inv(m_{x1}), m_x), m_{y2}) \in xy)$$

$$\Psi^1_{\oplus}: \quad (obj^{\oplus}(m_x, m_{x1}, m_{x2}) \wedge (m_x, m_y) \in xy \wedge m_{x1} \in DY(kb)) \Rightarrow \\ (\exists m_{y1} \in DY(kb') : (m_{x1}, m_{y1}), (m_{x2}, \oplus(\ominus(m_{y1}), m_y)) \in xy)$$

$$\Psi^2_{\oplus}: \quad (obj^{\oplus}(m_y, m_{y1}, m_{y2}) \wedge (m_x, m_y) \in xy \wedge m_{y1} \in DY(kb')) \Rightarrow \\ (\exists m_{x1} \in DY(kb) : (m_{x1}, m_{y1}), (\oplus(\ominus(m_{x1}), m_x), m_{y2}) \in xy)$$

A composition by $*$ on one side may have a decrypt-type or a key-ed transformation effect on the other side, only if the left part in the composition is mapped (using additionally *inv*) to the crypt-key or respectively the trans-key. A composition by $\oplus$ on one side may have a decrypt-type effect on the other side, only if the left part in the composition is mapped (using additionally $\ominus$) to the crypt-key.

The following conditions are partly redundant with Ψ^1_* and Ψ^2_*, but cover additional transform operations by $*$.

$$\Psi^1_{*,\oplus}: \quad (obj^{\oplus}(m_x, m_{x1}, m_{x2}) \wedge syn^*(m_{x1}, m_{x3}, m_{x4}) \wedge (m_x, m_y) \in xy \wedge \\ m_{x3} \in DY(kb)) \Rightarrow \\ (\exists m_{y1} \in DY(kb') : \\ (m_{x3}, m_{y1}), (\oplus(m_{x4}, *(inv(m_{x3}), m_{x2})), *(inv(m_{y1}), m_y)) \in xy)$$

$$\Psi^2_{*,\oplus}: \quad (obj^{\oplus}(m_y, m_{y1}, m_{y2}) \wedge syn^*(m_{y1}, m_{y3}, m_{y4}) \wedge (m_x, m_y) \in xy \wedge \\ m_{y3} \in DY(kb')) \Rightarrow \\ (\exists m_{x1} \in DY(kb) : \\ (m_{x1}, m_{y3}), (*(inv(m_{x1}), m_x), \oplus(m_{y4}, *(inv(m_{y3}), m_{y2}))) \in xy)$$

These conditions ensure that if an application of $*$ reverses a previous application on one side of a mapping, the same effect happens on the other side. The used left $*$-sub-message, crypt-key or trans-key on the one-side must be mapped to a pendant on the other side, which can be independently left $*$-sub-message, crypt-key or trans-key.

The following conditions cover the merging effect by $\oplus$.

$$\Psi^1_{\oplus,mrg}: \quad (obj^{\oplus}(m_{x0}, m_{x2}, m_x) \wedge obj^{\oplus}(m_{x1}, m_{x3}, \ominus(m_x)) \wedge \\ (m_{x0}, m_{y0}), (m_{x1}, m_{y1}) \in xy) \\ \Rightarrow (\oplus(m_{x2}, m_{x3}), \oplus(m_{y0}, m_{y1})) \in xy$$

$$\Psi^2_{\oplus,mrg}: \quad (obj^{\oplus}(m_{y0}, m_{y2}, m_y) \wedge obj^{\oplus}(m_{y1}, m_{y3}, \ominus(m_y)) \\ \wedge (m_{x0}, m_{y0}), (m_{x1}, m_{y1}) \in xy) \\ \Rightarrow (\oplus(m_{x0}, m_{x1}), \oplus(m_{y2}, m_{y3})) \in xy$$

If two mapped $\oplus$-objects have inverse $\oplus$-parts, the result of their merging must be mapped to the result of $\oplus$ applied to their images.

13.3 Proof of the Structural Mapping Lemma

The structural mapping lemma 122 is shown practically the same way as described in Sec. 11.2. The proof is by induction on the structure of m, where the base case and most proof situations in the step case are almost identical. Clearly, the necessary and sufficient conditions in Ψ that we employ in case $m \notin dom(xy)$ of the step case must be adapted to

the type of messages in the TC-AMP algebra (see Sec. 13.2). In this section, we focus on the handling of the canonical cases. The handling of the non-canonical cases is described in Sec. 13.4–13.6.

Cases Ξ_{fst}, Ξ_{snd} and Ξ_{pair} are handled the same way as in the PACE algebra. For the remaining canonical cases, we proceed as described in Sec. 11.2.2.2.

13.3.1 Proof Task (i):

According to the proof plan in Sec. 11.2.2.2.1, this proof task is relevant only for *inv*-objects. It is handled by assuming $\neg obj^{inv}(inv(m'_0),m'_0)$ and using that to obtain the structure for m'_0. After that, the structural condition for the mapping $m_0 \overset{xy}{\leftrightarrow} m'_0$ permits to propagate the structure of m'_0 to m_0 and then to refute one of the assumptions on $inv(m_0)$. This is done with the help of the necessary condition Ψ^2_{inv}.

Proof Details: We have $(m_0,m'_0) \in xy$ or $\Xi(m_0,m'_0,xy,kb,kb')$ according to the induction hypothesis. This permits to proceed by the following case distinction:

1. $(m_0,m'_0) \in xy$: We first apply Ψ^2_{inv} using $(m_0,inv(m'_1)) \in xy$ to obtain $m_0 = inv(m_{x1})$ and $(m_{x1},m'_1) \in xy$. This implies $inv(m_0) = inv(inv(m_{x1})) = m_{x1}$ and $(inv(m_0),m'_1) \in xy$, which refutes $inv(m_0) \notin dom(xy)$.

2. $(m_0,m'_0) \notin xy$: Here, we base the proof on the structure of m'_0, being an *inv*-object. For that purpose, we require the *definition of* Ξ *to respect a certain mutual-exclusion principle* so that $\Xi(m_0,m'_0,xy,kb,kb')$ for an *inv*-object m'_0 can be reduced to the canonical case Ξ_{inv}. That is, we obtain $m_2 \in DY(kb)$, $obj^{inv}(m_0,m_2)$, $m'_2 \in DY(kb')$, $m_2 \overset{xy}{\leftrightarrow} m'_2$ and $obj^{inv}(m'_0,m'_2)$. This permits to refute $obj^{inv}(inv(m_0),m_0)$. □

13.3.2 Proof Task (ii):

According to the proof plan in Sec. 11.2.2.2.2, the handling of this task is mainly by assuming $(m_x,f(m'_0,\ldots,m'_{n-1})) \in xy$ and then using an appropriate necessary condition to instantiate m_x with $f(m_0,\ldots,m_{n-1})$, which permits to refute $f(m_0,\ldots,m_{n-1}) \notin dom(xy)$.

The handling of *inv*-objects is based on Ψ^2_{inv}. For the handling of h_1- and h_2-objects, we use the necessary conditions $\Psi^2_{h_1}$ and respectively $\Psi^2_{h_2}$.

Proof Details: For *inv*-objects, we use Ψ^2_{inv} and $(m_x,inv(m'_0)) \in xy$ to get (m_{x1},m'_0) and $m_x = inv(m_{x1})$. Combining $m_0 \overset{xy}{\leftrightarrow} m'_0$ with the pair (m_{x1},m'_0) in xy implies $m_{x_1} = m_0$, $m_x = inv(m_0)$ and thus $(inv(m_0),inv(m'_0)) \in xy$, which refutes the assumption $inv(m_0) \notin dom(xy)$.

For h_1- and h_2-objects, we proceed the same way using $\Psi^2_{h_1}$ and respectively $\Psi^2_{h_2}$. □

13.4 Handling of the non-canonical $\Xi_{\ominus}$ Case

The $\Xi_{\ominus}$-case defines the mapping by decomposition for $\ominus$-objects. To simplify matters, we exclude the case where $\ominus$-objects result by synthesis operations ($syn^*_{\ominus}$). The mappings for such $\ominus$-objects are covered in the Ξ_*-case or by appropriate inclusion rules (cp. $\Psi^1_{\ominus,*}$ and $\Psi^2_{\ominus,*}$ in Sec. 13.2).

The predicate $\Xi_\ominus$ is defined by a slight adaptation of the canonical definition (for Ξ_f) in Sec. 11.1.2.3:

$$\Xi_\ominus: \quad \exists m_0 \in DY(kb), m_0' \in DY(kb') : \\ obj^\ominus(m, m_0) \wedge \neg isObj^*(m_0) \wedge m_0 \overset{xy}{\leftrightarrow} m_0' \wedge \neg isObj^*(m_0') \wedge obj^\ominus(m', m_0')$$

The use of $\neg isObj^*(m_0)$, which excludes further compositions of m_0 (and m) by $*$, allows us to handle the $\Xi_\ominus$-case in the step case of lemma 122 like a canonical case: For $m \in DY(kb)$ and $m \notin dom(xy)$, the proof consists in providing $m' \in DY(kb')$ where $m \overset{xy}{\leftrightarrow} m'$ and one of the Ξ-cases hold. We focus on the $\Xi_\ominus$-case when m is a $\ominus$-object and its $\ominus$-part m_0 is not a $*$-object. Since m_0 is smaller than m, the induction hypothesis provides $m_0' \in DY(kb')$ with $m_0 \overset{xy}{\leftrightarrow} m_0'$. Using $m \notin dom(xy)$, we thus apply the definition of $\widetilde{rec}$ to obtain $m \overset{xy}{\mapsto} \ominus(m_0')$ and set $m' = \ominus(m_0')$.

For $m \overset{xy}{\leftrightarrow} \ominus(m_0')$ it remains to show $m \overset{xy}{\leftarrow} \ominus(m_0')$. To have a practically the same proof as in a canonical case, we use the condition $\neg isObj^*(m_0')$ (in addition to $obj^\ominus(m', m_0')$). So, this proof consists in showing $obj^\ominus(m', m_0')$, $\neg isObj^*(m_0')$ and $m \overset{xy}{\leftarrow} \ominus(m_0')$, based in particular on the structural condition of $m_0 \overset{xy}{\leftrightarrow} m_0'$.

13.4.1 Proof Task (i):

First, we prove $obj^\ominus(m', m_0')$, i.e. $obj^\ominus(\ominus(m_0'), m_0')$, as described in Sec. 11.2.2.1.1 by contradiction, based on the necessary conditions Ψ^2_∞, $\Psi^2_\ominus$ and $\Psi^2_{\ominus,\oplus}$.

Proof Details: We need to refute the following cases resulting from the expansion of $\neg obj^\ominus(\ominus(m_0'), m_0')$:

1. $m_0' = \infty$ and $\ominus(m_0') = \infty$: In case $(m_0, m_0') \in xy$, we have thus $(m_0, \infty) \in xy$ and the application of Ψ^2_∞ yields $m_0 = \infty$. This permits to refute $obj^\ominus(m, m_0)$.

   The complementary case, i.e. $\Xi(m_0, m_0', xy, kb, kb')$, must not hold for $m_0' = \infty$.

2. $obj^\ominus(m_0', m_1')$ and $\ominus(m_0') = m_1'$: In case $(m_0, m_0') \in xy$, the application of $\Psi^2_\ominus$ yields $(\ominus(m_0), \ominus(m_0')) \in xy$, i.e. $(m, \ominus(m_0')) \in xy$, which permits to refute $m \notin dom(xy)$.

   In the second case for $m_0 \overset{xy}{\leftrightarrow} m_0'$, i.e. $\Xi(m_0, m_0', xy, kb, kb')$, the structure of m_0' excludes all cases, except of the $\Xi_\ominus$-, Ξ_* and the $\Xi_\oplus$-case. In all three cases, the obtained conditions on m_0 permit to refute $obj^\ominus(m, m_0)$ and/or $\neg isObj^*(m_0)$, as explained in the following:

   - In the $\Xi_\ominus$-case, we obtain property $obj^\ominus(m_0, m_1)$ and this can be used to show $\neg obj^\ominus(\ominus(m_0), m_0)$, i.e. $\neg obj^\ominus(m, m_0)$.
   - In the Ξ_*-case (see below), the message m_0 satisfies $isObj^*(m_0)$, $syn^*_\ominus(m_0, m_1, m_2)$ or $syn^*_\oplus(m_0, m_1, m_2)$, which refutes $\neg isObj^*(m_0)$ and/or $obj^\ominus(m, m_0)$.
   - In the $\Xi_\oplus$-case (see below), m_0 satisfies $isObj^\oplus(m_0)$, which refutes $obj^\ominus(m, m_0)$.

3. $obj^\oplus(m_0', m_1', m_2')$ and $obj^\oplus(\ominus(m_0'), \ominus(m_1'), \ominus(m_2'))$: In case $(m_0, m_0') \in xy$, the application of $\Psi^2_{\ominus,\oplus}$ permits to refute $m \notin dom(xy)$, as in (2).

   In the second case for $m_0 \overset{xy}{\leftrightarrow} m_0'$, i.e. $\Xi(m_0, m_0', xy, kb, kb')$, the structure of m_0' excludes all cases, except of the $\Xi_\ominus$-, Ξ_*- and $\Xi_\oplus$-case. In all three cases, the obtained conditions on m_0 permit to refute $obj^\ominus(m, m_0)$ and/or $\neg isObj^*(m_0)$ as explained above in (2). □

13.4.2 Proof Task (ii):

Next, we prove $\neg isObj^*(m_0')$ by assuming $isObj^*(m_0')$ and refuting this assumption, based on the necessary condition $\Psi^2_{\ominus,*}$.

Proof Details: The assumption $isObj^*(m_0')$ permits to reduce the structural condition on $m_0 \overset{xy}{\leftrightarrow} m_0'$ to $(m_0, m_0') \in xy$. The alternative case $\Xi(m_0, m_0', xy, kb, kb')$ can be excluded based on the same arguments as in proof situations (2) and (3) above (in Sec. 13.4.1). This permits to handle our proof task by the application of $\Psi^2_{\ominus,*}$, which yields $(\ominus(m_0), \ominus(m_0'))$ in xy. Hence, $m \notin dom(xy)$ is refuted as in proof situation (2) above (in Sec. 13.4.1). □

13.4.3 Proof Task (iii):

Finally, we prove $\ominus(m_0) \overset{xy}{\leftarrow} \ominus(m_0')$, i.e. $m \overset{xy}{\leftarrow} \ominus(m_0')$, with the help of $obj^{\ominus}(\ominus(m_0'), m_0')$ and $\neg isObj^*(m_0')$, as in Sec. 11.2.2.1.2: Since $\ominus(m_0')$ can be decomposed only by $\ominus$, the definition of $\widetilde{rec}$ yields $\ominus(m_0) \overset{xy}{\leftarrow} \ominus(m_0')$, provided we refute $\ominus(m_0') \in codom(xy)$. Here, we assume $(m_x, \ominus(m_0')) \in xy$ and apply $\Psi^2_{\ominus}$ to obtain $(\ominus(m_x), \ominus(\ominus(m_0'))) \in xy$, i.e. $(\ominus(m_x), m_0') \in xy$. This allows us to deduce $\ominus(m_x) = m_0$, i.e. $m_x = \ominus(m_0) = m$, and thus $(m, \ominus(m_0')) \in xy$, which refutes $m \notin dom(xy)$. □

13.5 Handling of the non-canonical Ξ_* Case

The Ξ_*-case defines the mapping by decomposition for messages that can be composed by "$*$", i.e. for $*$-objects as well as for $\ominus$- and $\oplus$-objects, which could also result by synthesis operations $syn^*_{\ominus}$ and $syn^*_{\oplus}$. To simplify matters, we restrict the Ξ_*-case to messages whose sub-messages required for a composition by "$\oplus$" are not available in $DY(kb)$ (see the $\Theta^{\oplus}_1$ predicate). Since multiple decomposition alternatives (into $*$-sub-messages) are possible, we use a definition of Ξ_* that covers also nested $*$-sub-messages obtained by successive decomposition. This definition employs the Θ^*_1 predicate, introduced below, to qualify the sub-message where the successive decomposition halts.

Before we define the Ξ_* predicate, we introduce further predicates used as short-cuts.

$$isSyn^*(m) \Leftrightarrow (\exists m_0, m_1 : syn^*(m, m_0, m_1))$$

$$syn^{\ominus}(m, m_0) \Leftrightarrow (obj^{\ominus}(m, m_0) \vee syn^{\ominus}_{\oplus}(m, m_0))$$

$$isSyn^{\ominus}(m) \Leftrightarrow (\exists m_0 : syn^{\ominus}(m, m_0))$$

$$\Theta^*_1(m, ik) \Leftrightarrow \forall m_0, m_1 \in DY(ik) : \neg syn^*(m, m_0, m_1)$$

$$\Theta^{\oplus}_1(m, ik) \Leftrightarrow \forall m_0, m_1 \in DY(ik) : \neg obj^{\oplus}(m, m_0, m_1)$$

Recall the predicates syn^* and $syn^{\overline{*}}$ introduced in Sec. 6.5.1.1: $syn^*(m, m_0, m_1)$ holds when m results by a constructor-type application or a synthesis operation of $*$ using m_0 and m_1 as arguments. $syn^{\overline{*}}(m, ms, x)$ holds when m results by such successive applications of $*$ using the elements in ms and starting with x. Accordingly, $syn^{\overline{*}}(m, ms, x)$ identifies in the following definition of Ξ_* *all* decompositions of m, which result in a left $*$-sub-message from ms and in a right $*$-sub-message given by the rest of ms and x.

$$\begin{aligned}\Xi_*: \quad & \exists ms \subset DY(kb), x \in DY(kb), ms' \subset DY(kb'), x' \in DY(kb') : \\ & syn^{\overline{*}}(m, ms, x) \wedge ms \neq \emptyset \wedge \Theta_1^*(x, kb) \wedge \Theta_1^{\oplus}(x, kb) \wedge \\ & \wp(ms, ms') \subset^{\overset{xy}{\leftrightarrow}} \wedge x \overset{xy}{\leftrightarrow} x' \wedge m' = \overline{*}(ms', x') \wedge isSyn^*(m')\end{aligned}$$

The predicate Ξ_* defines the mapped message m' to a syn^*-message m relative to the *arbitrarily many* available $*$-sub-messages (the elements of ms and x): m' is the result of successive applications of $*$ using the messages mapped to the elements of ms and to x.

Before we use the predicate Ξ_* in the proof of theorem 108 in TC-AMP (see Sec. 13.7), we describe the handling of syn^*-messages, i.e. the Ξ_*-case, in the step case of lemma 122: For $m \in DY(kb)$, $m \notin dom(xy)$, $\neg\Theta_1^*(m, kb)$ and $\Theta_1^{\oplus}(m, kb)$, we must provide $m' \in DY(kb')$ satisfying $m \overset{xy}{\leftrightarrow} m'$ and $\Xi_*(m, m', xy, kb, kb')$.

The proof starts by applying lemma 123 for *a successive decomposition of m into the available $*$-sub-messages in $DY(kb)$*. It provides $ms \subset DY(kb)$ and $x \in DY(kb)$ such that $syn^{\overline{*}}(m, ms, x)$, $ms \neq \emptyset$ and $\Theta_1^*(x, kb)$ hold. The additional condition $\Theta_1^{\oplus}(x, kb)$ ensues immediately from $\Theta_1^{\oplus}(m, kb)$. Since all elements in ms and x are smaller than m, we may apply the induction hypothesis to obtain $x' \in DY(kb')$ and $ms' \subset DY(kb')$ with $x \overset{xy}{\leftrightarrow} x'$ and $\wp(ms, ms') \subset^{\overset{xy}{\leftrightarrow}}$.

We continue the proof by setting $m' = \overline{*}(ms', x')$ and showing $m \overset{xy}{\leftrightarrow} m'$ as follows:

- First, we apply lemma 124 *to obtain the mapping of m by $\overset{xy}{\mapsto}$ using the available $*$-sub-messages*. It permits to show $\overline{*}(ms, x) \overset{xy}{\mapsto} \overline{*}(ms', x')$, i.e. $m \overset{xy}{\mapsto} \overline{*}(ms', x')$.

- Next, we want to show the mapping $\overline{*}(ms, x) \overset{xy}{\leftarrow} \overline{*}(ms', x')$, i.e. $m \overset{xy}{\leftarrow} \overline{*}(ms', x')$, and property $isSyn^*(\overline{*}(ms', x'))$. Here, we distinguish two complementary cases according to the possible structures of $\overline{*}(ms', x')$, x' and of x.

  1. If $syn^{\overline{*}}(\overline{*}(ms', x'), ms', x')$ holds and if x' and x satisfy $\Theta_1^*(x', kb')$, $\Theta_1^{\oplus}(x', kb')$ and $isSyn^{\ominus}(x') \Rightarrow isSyn^{\ominus}(x)$, then we use lemma 125.
     In this case, $\overline{*}(ms', x')$ can be composed by $*$ using left $*$-sub-messages from ms'. It can be composed by $\ominus$ only when this holds for $m = \overline{*}(ms, x)$, too.
  2. Otherwise, i.e. when $syn^{\overline{*}}(\overline{*}(ms', x'), ms', x')$ does not hold or x' and x satisfy $\neg\Theta_1^*(x', kb')$, $\neg\Theta_1^{\oplus}(x', kb')$ or $isSyn^{\ominus}(x') \wedge \neg isSyn^{\ominus}(x)$, the structural condition of $x \overset{xy}{\leftrightarrow} x'$ allows us to link m and $\overline{*}(ms', x')$ to an appropriate pair in xy that permits to use lemma 128. So, we use the following predicate to define how $m_c \in DY(kb)$ and $m'_c \in DY(kb')$ are linked to an appropriate $(x_c, x'_c) \in xy$:

     $$\begin{aligned}&\Theta_2^*(m_c, m'_c, xy, kb, kb') \Leftrightarrow \\ &(\exists ms_c \subset DY(kb), x_c \in DY(kb), ms'_c \subset DY(kb'), x'_c \in DY(kb') : \\ &\quad m_c = \overline{*}(ms_c, x_c) \wedge \wp(ms_c, ms'_c) \subset^{\overset{xy}{\leftrightarrow}} \wedge (x_c, x'_c) \in xy \wedge syn^{\overline{*}}(m'_c, ms'_c, x'_c))\end{aligned}$$

     Note that Θ_2^* is similar but not the same as the Ξ_* predicate. While Ξ_* decomposes m_c until a right $*$-sub-message without available $*$-sub-messages is obtained, Θ_2^* decomposes m'_c until a right $*$-sub-message x'_c mapped by a pair in xy is obtained. So, non-constructor-type applications of $*$ are embedded in Ξ_* on the right-side, but in Θ_2^* on the left-side.

In case (1), $isSyn^*(\overline{*}(ms', x'))$ ensues from $syn^{\overline{*}}(\overline{*}(ms', x'), ms', x')$ and $ms' \neq \emptyset$.

In case (2), where $\neg syn^{\overline{*}}(\overline{*}(ms',x'),ms',x')$, $isSyn^{\ominus}(x') \wedge \neg isSyn^{\ominus}(x)$, $\neg\Theta_1^*(x',kb')$ or $\neg\Theta_1^{\oplus}(x',kb')$ holds, we obtain property $\Theta_2^*(\overline{*}(ms,x),\overline{*}(ms',x'),xy,kb,kb')$ with the help of lemma 126. The application of this lemma reduces the structural differences between $\overline{*}(ms,x)$ (resp. x) and $\overline{*}(ms',x')$ (resp. x') to some pair in xy used to obtain the mapping of these messages. It is based on the structural condition of $x \overset{xy}{\leftrightarrow} x'$, $\Theta_1^*(x,kb)$, $\Theta_1^{\oplus}(x,kb)$ and $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$. Here, $isSyn^*(\overline{*}(ms',x'))$ follows from $\Theta_2^*(\overline{*}(ms,x),\overline{*}(ms',x'),xy,kb,kb')$ and the assumption $m \notin dom(xy)$: The definition of Θ_2^* could not provide $syn^{\overline{*}}(\overline{*}(ms',x'),ms'_c,x'_c)$, $\wp(ms_c,ms'_c) \subset \overset{xy}{\leftrightarrow}$, $(x_c,x'_c) \in xy$ and $\overline{*}(ms,x) = \overline{*}(ms_c,x_c)$ with $ms'_c = \emptyset$, as this implies $ms_c = \emptyset$, $\overline{*}(ms,x) = x_c$ and thus $(\overline{*}(ms,x),x'_c) \in xy$, which refutes $m \notin dom(xy)$. That is, we only obtain $syn^{\overline{*}}(\overline{*}(ms',x'),ms'_c,x'_c)$ with $ms'_c \neq \emptyset$, which implies $isSyn^*(\overline{*}(ms',x'))$.

13.5.1 Decomposition into $*$-Sub-Messages in $DY(kb)$

For syn^*-messages, the following lemma permits to identify the result of a successive decomposition into available $*$-sub-messages.

Lemma[VSE] 123 (Decomposition into $*$-Sub-Messages in $DY(kb)$):
Let m satisfy $syn^*(m,m_0,m_1)$ for m_0 and m_1 in $DY(kb)$. Then, it exists $ms \subset DY(kb)$ and $x \in DY(kb)$ satisfying

$$syn^{\overline{*}}(m,ms,x) \wedge ms \neq \emptyset \wedge \Theta_1^*(x,kb).$$

The proof (by induction on $|m|$) is similar to the proof of lemma 111.

13.5.2 Mapping by $\overset{xy}{\mapsto}$ using the $*$-Sub-Messages in $DY(kb)$

For syn^*-messages that are not mapped in xy, the following lemma provides the mapping by $\overset{xy}{\mapsto}$ relative to the mappings of the available $*$-sub-messages.

Lemma[VSE] 124 (Mapping of syn^*-Messages by $\overset{xy}{\mapsto}$):
Let $ms \subset DY(kb)$, $x \in DY(kb)$, $ms' \subset DY(kb)$ and $x' \in DY(kb)$ satisfy $syn^{\overline{*}}(\overline{*}(ms,x),ms,x)$, $ms \neq \emptyset$, $\Theta_1^*(x,kb)$, $\Theta_1^{\oplus}(x,kb)$, $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$, $x \overset{xy}{\leftrightarrow} x'$ and $\overline{*}(ms,x) \notin dom(xy)$. Let (the induction hypothesis of) lemma 122 hold for all $\hat{m}$ with $|\hat{m}| < |\overline{*}(ms,x)|$. Then, we have

$$\overline{*}(ms,x) \overset{xy}{\mapsto} \overline{*}(ms',x').$$

Since $\overline{*}(ms,x) \notin dom(xy)$ and $ms \neq \emptyset$ hold, we prove $\overline{*}(ms,x) \overset{xy}{\mapsto} \overline{*}(ms',x')$ by showing that for any decomposition of m into $*$-sub-messages $m_0,m_1 \in DY(kb)$ and possibly into a $\ominus$-sub-message m_0 the composition of the corresponding mapped messages (by $*$ and respectively $\ominus$) yields $\overline{*}(ms',x')$.

13.5.2.1 Mapping by Decomposition into $*$-Sub-Messages

We start with decompositions into $*$-sub-messages. That is, we assume $m_0,m_1 \in DY(kb)$ with $syn^*(\overline{*}(ms,x),m_0,m_1)$ and want to show that m'_0 and m'_1 in the mappings $m_0 \overset{xy}{\leftrightarrow} m'_0$ and $m_1 \overset{xy}{\leftrightarrow} m'_1$, which are provided by (the induction hypothesis of) lemma 122, satisfy the equality $*(m'_0,m'_1) = \overline{*}(ms',x')$.

Based on $syn^*(\overline{*}(ms,x),m_0,m_1)$, $syn^{\overline{*}}(\overline{*}(ms,x),ms,x)$, $ms \subset DY(kb)$, $ms \neq \emptyset$, $\Theta_1^*(x,kb)$ and $x,m_0,m_1 \in DY(kb)$, we obtain $m_0 \in ms$, $ms = m_0 \uplus ms_0$ and $syn^{\overline{*}}(m_1,ms_0,x)$. Furthermore, $\wp(ms,ms') \subset^{xy}_{\leftrightarrow}$ means that there is ms'_0 with $ms' = m'_0 \uplus ms'_0$ and $\wp(ms_0,ms'_0) \subset^{xy}_{\leftrightarrow}$. In the following, we want to show that $m'_1 = \overline{*}(ms'_0,x')$ to prove the required equality by

$$\begin{aligned} *(m'_0,m'_1) &= *(m'_0,\overline{*}(ms'_0,x')) \\ &= \overline{*}(m'_0 \uplus ms'_0,x') \\ &= \overline{*}(ms',x'). \end{aligned}$$

- If $ms_0 = \emptyset$, we obtain $m_1 = x$, $m'_1 = x'$, and $ms'_0 = \emptyset$. This permits to show the required equality by $\overline{*}(ms'_0,x') = \overline{*}(\emptyset,x') = x' = m'_1$.
- In the complementary case, i.e. $ms_0 \neq \emptyset$, m_1 is a syn^*-message and this permits to show $m'_1 = \overline{*}(ms'_0,x')$ based on the structural condition for $m_1 \overset{xy}{\leftrightarrow} m'_1$ by the following case distinction:
  1. When $(m_1,m'_1) \in xy$, i.e. $(\overline{*}(ms_0,x),m'_1) \in xy$, we apply condition Ψ^1_* to instantiate m'_1 with $\overline{*}(ms'_0,x')$.
  2. When $\Xi(m_1,m'_1,xy,kb,kb')$, structural properties on m_1 such as $syn^{\overline{*}}(m_1,ms_0,x)$, $ms_0 \neq \emptyset$ and $\Theta_1^{\oplus}(x,kb)$ permit to reduce this to the Ξ_*-case. That is, we get $syn^{\overline{*}}(m_1,ms_1,x_1)$, $ms_1 \neq \emptyset$, $\Theta_1^*(x_1,kb)$, $\Theta_1^{\oplus}(x_1,kb)$, $\wp(ms_1,ms'_1) \subset^{xy}_{\leftrightarrow}$, $x_1 \overset{xy}{\leftrightarrow} x'_1$ and $m'_1 = \overline{*}(ms'_1,x'_1)$. Based on $syn^{\overline{*}}(m_1,ms_0,x)$, $syn^{\overline{*}}(m_1,ms_1,x_1)$, $\Theta_1^*(x,kb)$ and $\Theta_1^*(x_1,kb)$, we obtain $x = x_1$ and $ms_1 = ms_0$. This yields $x'_1 = x'$ and $ms'_1 = ms'_0$, permitting to rewrite $m'_1 = \overline{*}(ms'_1,x'_1)$ to $m'_1 = \overline{*}(ms'_0,x')$, as required. □

13.5.2.2 Mapping by Decomposition into a $\ominus$-Sub-Message

We continue the proof with a possible decomposition into a $\ominus$-sub-message. That is, we assume $m_0 \in DY(kb)$ with $syn^{\ominus}(\overline{*}(ms,x),m_0)$ and want to show that m'_0 in the mapping $m_0 \overset{xy}{\leftrightarrow} m'_0$, which is provided by (the induction hypothesis of) lemma 122, satisfies the equality $\ominus(m'_0) = \overline{*}(ms',x')$.

Based on the structural properties $syn^{\ominus}(\overline{*}(ms,x),m_0)$, $syn^{\overline{*}}(\overline{*}(ms,x),ms,x)$ and $ms \neq \emptyset$, we obtain $syn^{\overline{*}}(m_0,ms,x_0)$ and $syn^{\ominus}(x,x_0)$ for $x_0 \in DY(kb)$. Then, we use $syn^{\ominus}(x,x_0)$, $\Theta_1^*(x,kb)$, $\Theta_1^{\oplus}(x,kb)$ and the structural condition for $x \overset{xy}{\leftrightarrow} x'$ to deduce $x_0 \overset{xy}{\leftrightarrow} \ominus(x')$:

- When $(x,x') \in xy$, we get $(\ominus(x),\ominus(x')) \in xy$, i.e. $(x_0,\ominus(x')) \in xy$, by case distinction and with the help of the necessary conditions $\Psi^1_{\ominus}$, $\Psi^1_{\ominus,*}$ and $\Psi^1_{\ominus,\oplus}$.
- When $\Xi(x,x',xy,kb,kb')$, the structural properties on x permit to reduce this to the $\Xi_{\ominus}$-case. This yields $x_0 \overset{xy}{\leftrightarrow} x'_0$ and $obj^{\ominus}(x',x'_0)$. Hence, we have $x' = \ominus(x'_0)$ and thus $\ominus(x') = x'_0$, permitting to rewrite $x_0 \overset{xy}{\leftrightarrow} x'_0$ to $x_0 \overset{xy}{\leftrightarrow} \ominus(x')$.

Finally, we use $x_0 \overset{xy}{\leftrightarrow} \ominus(x')$ to show $m'_0 = \overline{*}(ms',\ominus(x'))$ and hence $\ominus(m'_0) = \overline{*}(ms',x')$, based on the structural condition for $m_0 \overset{xy}{\leftrightarrow} m'_0$:

- When $(m_0,m'_0) \in xy$, i.e. $(\overline{*}(ms,x_0),m'_0) \in xy$ holds, we employ condition Ψ^1_* to obtain $(x_0,\overline{*}(\overline{inv}(ms'),m'_0)) \in xy$ and use that to deduce $\ominus(x') = \overline{*}(\overline{inv}(ms'),m'_0)$, which transforms obviously to the required equality $\overline{*}(ms',\ominus(x')) = m'_0$.
- When $\Xi(m_0,m'_0,xy,kb,kb')$, the structural properties on m_0 permit to reduce this to the Ξ_*-case. That is, we get the structural properties $syn^{\overline{*}}(m_0,ms_1,x_1)$, $ms_1 \neq \emptyset$, $\Theta_1^*(x_1,kb)$, $\Theta_1^{\oplus}(x_1,kb)$, $\wp(ms_1,ms'_1) \subset^{xy}_{\leftrightarrow}$, $x_1 \overset{xy}{\leftrightarrow} x'_1$ and $m'_0 = \overline{*}(ms'_1,x'_1)$. Based on

$syn^{\overline{*}}(m_0,ms,x_0)$, $syn^{\overline{*}}(m_0,ms_1,x_1)$, $\Theta_1^*(\ominus(x_0),kb)$ and $\Theta_1^*(x_1,kb)$, we obtain $x_0 = x_1$ and $ms_1 = ms$. This yields $x_1' = \ominus(x')$ and $ms_1' = ms'$, permitting to rewrite $m_0' = \overline{*}(ms_1',x_1')$ to $m_0' = \overline{*}(ms',\ominus(x'))$, as required. □

13.5.3 Mapping by $\overset{xy}{\leftarrow\!\!\shortmid}$ using the $*$-Sub-Messages in $DY(kb')$

In this section, we describe the lemmata used in the Ξ_*-case for the proof of the mapping by $\overset{xy}{\leftarrow\!\!\shortmid}$. The first lemma provides the mapping of syn^*-messages for which only available $*$-sub-messages and possibly a $\ominus$-sub-message with known mappings exist.

Lemma[VSE] 125 (Mapping of syn^*-Messages by $\overset{xy}{\leftarrow\!\!\shortmid}$, Case 1):
Let $ms \subset DY(kb)$, $x \in DY(kb)$, $ms' \subset DY(kb')$ and $x' \in DY(kb')$ satisfy $syn^{\overline{*}}(\overline{*}(ms,x),ms,x)$, $ms \neq \emptyset$, $\Theta_1^*(x,kb)$, $\Theta_1^{\oplus}(x,kb)$, $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$ and $x \overset{xy}{\leftrightarrow} x'$. Let (the induction hypothesis of) lemma 122 hold for all $\hat{m}$ with $|\hat{m}| < |\overline{*}(ms,x)|$. Then, we have

$$\begin{aligned}&(syn^{\overline{*}}(\overline{*}(ms',x'),ms',x') \wedge \Theta_1^*(x',kb') \wedge \\ &\Theta_1^{\oplus}(x',kb') \wedge (isSyn^{\ominus}(x') \Rightarrow isSyn^{\ominus}(x))) \\ &\Rightarrow \overline{*}(ms,x) \overset{xy}{\leftarrow\!\!\shortmid} \overline{*}(ms',x').\end{aligned}$$

Proof Details: The proof is simple when $\overline{*}(ms',x') \in codom(xy)$ holds: Having the pair $(m_x,\overline{*}(ms',x')) \in xy$, we use condition Ψ_*^2 to instantiate m_x with $\overline{*}(ms,x)$ and this immediately implies $\overline{*}(ms,x) \overset{xy}{\leftarrow\!\!\shortmid} \overline{*}(ms',x')$.

In the complementary case, i.e. $\overline{*}(ms',x') \notin codom(xy)$, we prove $\overline{*}(ms,x) \overset{xy}{\leftarrow\!\!\shortmid} \overline{*}(ms',x')$ by showing that for any decomposition of $\overline{*}(ms',x')$ into $*$-sub-messages $m_0',m_1' \in DY(kb')$ and possibly into the $\ominus$-sub-message m_0' the composition of the corresponding mapped messages (by $*$ and respectively $\ominus$) yields $\overline{*}(ms,x)$. Recall, $\Theta_1^{\oplus}(x',kb')$ excludes any decomposition of $\overline{*}(ms',x')$ into $\oplus$-parts from $DY(kb')$.

- Based on $syn^{\overline{*}}(\overline{*}(ms',x'),ms',x')$, $\Theta_1^*(x',kb')$ and $syn^*(\overline{*}(ms',x'),m_0',m_1')$, the $*$-sub-messages m_0' and m_1' yield a partition $ms' = m_0' \uplus ms_0'$ with $m_1' = \overline{*}(ms_0',x')$. Using $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$, we identify some $m_0 \in ms$ satisfying $m_0 \overset{xy}{\leftrightarrow} m_0'$, a corresponding partition $m_0 \uplus ms_0 = ms$ and $m_1 = \overline{*}(ms_0,x)$. Then, (the induction hypothesis of) lemma 122 permits to show $m_1 \overset{xy}{\leftrightarrow} \overline{*}(ms_0',x')$, i.e. $m_1 \overset{xy}{\leftrightarrow} m_1'$, as described in Sec. 13.5.2.1. This allows us to prove the required equality by $*(m_0,m_1) = *(m_0,\overline{*}(ms_0,x)) = \overline{*}(ms,x)$.
- Based on $(isSyn^{\ominus}(x') \Rightarrow isSyn^{\ominus}(x))$, the $\ominus$-sub-message m_0' yields x_0' satisfying $syn^{\ominus}(x',x_0')$ with $m_0' = \overline{*}(ms',x_0')$ and x_0 satisfying $syn^{\ominus}(x,x_0)$. Using x_0 and ms, we identify $m_0 = \overline{*}(ms,x_0)$ as the $\ominus$-sub-message of $\overline{*}(ms,x)$. Then, (the induction hypothesis of) lemma 122 permits to show $m_0 \overset{xy}{\leftrightarrow} \overline{*}(ms',x_0')$, i.e. $m_0 \overset{xy}{\leftrightarrow} m_0'$, as described in Sec. 13.5.2.2. This allows us to prove the required equality by $\ominus(m_0) = \ominus(\overline{*}(ms,x_0)) = \overline{*}(ms,\ominus(x_0)) = \overline{*}(ms,x)$. □

When the required structural conditions on the mapped messages by lemma 125 do not hold, we prepare the application of lemma 128 by deducing the Θ_2^* condition. This is carried out by the following lemma.

Lemma[VSE] 126 (Θ_2^* for Mapping of syn^*-Messages by $\overset{xy}{\leftarrow\!\!\shortmid}$):
Let $ms \subset DY(kb)$, $ms' \subset DY(kb')$, $x \in DY(kb)$ and $x' \in DY(kb')$ satisfy $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$,

$x \overset{xy}{\leftrightarrow} x'$ and $(x,x') \in xy$ or $\Xi(x,x',xy,kb,kb')$. Furthermore, let $\neg syn^{\overline{*}}(\overline{*}(ms',x'),ms',x')$, $isSyn^{\ominus}(x') \wedge \neg isSyn^{\ominus}(x)$, $\neg\Theta_1^*(x',kb')$ or $\neg\Theta_1^{\oplus}(x',kb')$ hold. Then, we have

$$\Theta_2^*(\overline{*}(ms,x),\overline{*}(ms',x'),xy,kb,kb').$$

Proof Details: First, $\Theta_1^*(x,kb)$ and $\Theta_1^{\oplus}(x,kb)$ reduce $\Xi(x,x',xy,kb,kb')$ to a canonical case Ξ_f or to the $\Xi_{\ominus}$-case. Then, we want to show that the structural conditions on $\overline{*}(ms',x')$, x' and on x falsify the canonical cases Ξ_f and the $\Xi_{\ominus}$-case:

1. If $syn^{\overline{*}}(\overline{*}(ms',x'),ms',x')$ does not hold, it exists $m'_0 \in ms'$ with $syn^*(x',inv(m'_0),x'_0)$ or $obj^{\oplus}(x',x'_0,x'_1)$, $syn^*(x'_0,inv(m'_0),x'_2)$ and $syn^*(*(m'_0,x'_1),m'_0,x'_1)$. Hence, the structure of x' does not match neither any canonical case nor the $\Xi_{\ominus}$-case.
2. When $isSyn^{\ominus}(x') \wedge \neg isSyn^{\ominus}(x)$ holds, the structures of x and x' do not match neither any canonical case nor the $\Xi_{\ominus}$-case.
3. When $\neg\Theta_1^*(x',kb')$ or $\neg\Theta_1^{\oplus}(x',kb')$ holds, the structure of x' does not match neither any canonical case nor the $\Xi_{\ominus}$-case.

Consequently, $\Xi(x,x',xy,kb,kb')$ does not hold and we have $(x,x') \in xy$.

In case $syn^{\overline{*}}(\overline{*}(ms',x'),ms',x')$, we trivially have $\Theta_2^*(\overline{*}(ms,x),\overline{*}(ms',x'),xy,kb,kb')$ by definition.

In the complementary case, i.e. in case (1), we use $(x,x') \in xy$ and deduce that the pair $(*(m_0,x),*(m'_0,x'_0))$ is in xy, based on Ψ_*^2 or $\Psi_{*,\oplus}^2$.

In case property $syn^*(x',inv(m'_0),x'_0)$ holds, the application of condition Ψ_*^2 yields the pairs $(m_{x1},inv(m'_0)),(*(inv(m_{x1}),x),x'_0) \in xy$. Using $m_0 \overset{xy}{\leftrightarrow} m'_0$ and its structural condition, lemma 127 permits to get $m_{x1} = inv(m_0)$ and then to rewrite $(*(inv(m_{x1}),x),x'_0) \in xy$ to $(*(inv(inv(m_0)),x),x'_0) \in xy$, i.e. $(*(m_0,x),x'_0) \in xy$ for $x'_0 = *(m'_0,x')$. Similarly, we apply condition $\Psi_{*,\oplus}^2$ in the second case, where $obj^{\oplus}(x',x'_0,x'_1)$, $syn^*(x'_0,inv(m'_0),x'_2)$ and $syn^*(*(m'_0,x'_1),m'_0,x'_1)$ hold, to deduce $(*(m_0,x),*(m'_0,x'_0)) \in xy$.

In general, there can be a non-empty $ms'_0 \subseteq ms'$ containing messages that are simplified like m'_0. So, the derivation of $(*(m_0,x),*(m'_0,x'_0)) \in xy$ is generalized (by induction) to the derivation of $(\overline{*}(ms_0,x),\overline{*}(ms'_0,x'_0)) \in xy$. For our proof, we need to identify the non-simplified elements of ms' by providing $ms'_1 \subset ms'$ with $ms' = ms'_1 \uplus ms'_0$ and $syn^{\overline{*}}(\overline{*}(ms',x'),ms'_1,\overline{*}(ms'_0,x'_0))$. Such ms'_1 exists by construction. Hence, we have all we need to show $\Theta_2^*(\overline{*}(ms,x),\overline{*}(ms',x'),xy,kb,kb')$ by definition. □

The next lemma, used in particular in the above proof, permits to transfer a mapping to an application of inv.

Lemma[VSE] 127 ($\overset{xy}{\leftrightarrow}$, inv):
For $m_x,m \in DY(kb)$ and $m' \in DY(kb')$, we have

$$(m \overset{xy}{\leftrightarrow} m' \wedge ((m,m') \in xy \vee \Xi(m,m',xy,kb,kb')))$$
$$\Rightarrow ((m_x,inv(m')) \in xy \Rightarrow m_x = inv(m)).$$

Proof Details: The proof is by case distinction on the structure of $inv(m')$:

- In case $obj^{inv}(inv(m'),m')$ holds, we apply Ψ_{inv}^2 to $(m_x,inv(m')) \in xy$ and obtain $(m_{x1},m') \in xy$ and $m_x = inv(m_{x1})$. This permits to deduce $m_{x1} = m$ and hence $m_x = inv(m)$
- In case property $sel_{inv}^{inv}(inv(m'),m')$ holds, we have $obj^{inv}(m',m'_1)$ and $inv(m') = inv(inv(m'_1)) = m'_1$, which permits to rewrite $(m_x,inv(m')) \in xy$ to $(m_x,m'_1) \in xy$. Then, we proceed by case distinction:

1. When $(m,m') \in xy$, i.e. $(m,inv(m'_1)) \in xy$, we apply Ψ^2_{inv} to $(m,inv(m'_1)) \in xy$ and obtain $(m_{x1},m'_1) \in xy$ and $m = inv(m_{x1})$. Using $(m_x,m'_1) \in xy$, we obtain thus $m_x = m_{x1}$, $m = inv(m_x)$ and hence $m_x = inv(m)$.
2. When $\Xi(m,m',xy,kb,kb')$, the structure of m' reduces this to the Ξ_{inv}-case. That is, $obj^{inv}(m,m_0)$, $m_0 \overset{x}{\leftrightarrow} m'_0$ and $obj^{inv}(m',m'_0)$. Using $obj^{inv}(m',m'_1)$ and $(m_x,m'_1) \in xy$, we obtain $m'_1 = m'_0$, $m_0 = m_x$, $m = inv(m_x)$ and hence the required equality $m_x = inv(m)$. □

The fourth lemma provides the mapping of syn^*-messages by $\overset{xy}{\hookleftarrow}$ in case the mapping for one (possibly nested) right $*$-sub-message is given by a pair in xy.

Lemma[VSE] 128 (Mapping of syn^*-Messages by $\overset{xy}{\hookleftarrow}$, Case 2):
For $m \in DY(kb)$ and $m' \in DY(kb')$, we have

$$\Theta^*_2(m,m',xy,kb,kb') \Rightarrow m \overset{xy}{\hookleftarrow} m'.$$

Proof Details: The proof of this lemma is by induction on $|m'|$.

In the base case, where $|m'| = 0$, we have $m' = c_i$ and $(m,c_i) \in xy$, by the definition of Θ^*_2. This trivially implies $m \overset{xy}{\hookleftarrow} m'$.

In the step case, the definition of Θ^*_2 provides $m = \overline{*}(ms,x)$, $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$, $(x,x') \in xy$ and $syn^{\overline{*}}(m',ms',x')$.

If $ms' = \emptyset$, we have $m' = x'$, $m = x$ and $(m,m') \in xy$, which trivially implies $m \overset{xy}{\hookleftarrow} m'$.

Otherwise, if $m' \in codom(xy)$, we have $(m_x,\overline{*}(ms',x')) \in xy$ and we use Ψ^2_* to instantiate m_x with $\overline{*}(ms,x)$, i.e. m, and this immediately implies $m \overset{xy}{\hookleftarrow} m'$.

In the complementary case, where $ms' \neq \emptyset$ and $m' \notin codom(xy)$ hold, we prove $m \overset{xy}{\hookleftarrow} m'$ by providing *the* mapped messages for the available $*$-sub-messages, $\oplus$-parts and $\ominus$-sub-message of m' and then showing that their composition with $*$, $\oplus$ and respectively $\ominus$ yields $\overline{*}(ms,x)$, i.e. m.

- Let $syn^*(m',m'_0,m'_1)$ hold for two arbitrary available $*$-sub-messages of m'. Then, we distinguish two cases:
  1. When $m'_0 \in ms'$ holds, we have the multiset $ms' = m'_0 \uplus ms'_0$ and $syn^{\overline{*}}(m'_1,ms'_0,x')$. Here, $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$ permits to identify $m_0 \overset{xy}{\leftrightarrow} m'_0$ and $ms = m_0 \uplus ms_0$ with $\wp(ms_0,ms'_0) \subset \overset{xy}{\leftrightarrow}$. Since $inv(m_0)$ and m belong to $DY(kb)$ and $\overline{*}(ms_0,x) = *(inv(m_0),m)$, we have $\overline{*}(ms_0,x) \in DY(kb)$ and $\Theta^*_2(\overline{*}(ms_0,x),m'_1,xy,kb,kb')$, by definition. This permits to obtain $\overline{*}(ms_0,x) \overset{xy}{\hookleftarrow} m'_1$ by the induction hypothesis. Hence, we show the required equality by $*(m_0,\overline{*}(ms_0,x)) = \overline{*}(m_0 \uplus ms_0,x) = \overline{*}(ms,x)$.
  2. When $m'_0 \notin ms'$, we have $syn^*(x',m'_0,x'_0)$ and $syn^{\overline{*}}(m'_1,ms',x'_0)$. First, we use condition Ψ^2_* with $(x,*(m'_0,x'_0)) \in xy$ to get $(m_{x1},m'_0),(*(inv(m_{x1}),x),x'_0) \in xy$. Since $inv(m_{x1})$ and m belong to $DY(kb)$ and $\overline{*}(ms,x) = m$, we are able to have $\overline{*}(ms,*(inv(m_{x1}),x)) \in DY(kb)$ and $\Theta^*_2(\overline{*}(ms,*(inv(m_{x1}),x)),m'_1,xy,kb,kb')$, by definition. This permits to obtain the mapping $\overline{*}(ms,*(inv(m_{x1}),x)) \overset{xy}{\hookleftarrow} m'_1$ by the induction hypothesis. Hence, we show the required equality by

$$*(m_{x1},\overline{*}(ms,*(inv(m_{x1}),x))) = \overline{*}(ms,x).$$

- Let $obj^\oplus(m',m'_0,m'_1)$ hold for two arbitrary available $\oplus$-parts of m'. Based on $syn^{\overline{*}}(m',ms',x')$ and $ms' \subset DY(kb)$, it follows $obj^\oplus(x',x'_0,x'_1)$, $syn^{\overline{*}}(m'_0,ms',x'_0)$ and $syn^{\overline{*}}(m'_1,ms',x'_1)$ for $x'_0,x'_1 \in DY(kb')$.

  First, we want to use $(x,x') \in xy$ to deduce corresponding pairs as mappings for x'_0 and x'_1. This is done with the help of the necessary condition $\Psi^2_\oplus$.

  Using $\Psi^2_\oplus$ for $(x,\oplus(x'_0,x'_1)) \in xy$ yields $(m_{x1},x'_0),(\oplus(\ominus(m_{x1}),x),x'_1) \in xy$. This permits to have $\Theta^*_2(\overline{*}(ms,m_{x1}),m'_0,xy,kb,kb')$ and $\Theta^*_2(\overline{*}(ms,\oplus(\ominus(m_{x1}),x)),m'_1,xy,kb,kb')$ by definition, and then to obtain the mapping $\overline{*}(ms,m_{x1}) \overset{xy}{\hookleftarrow} m'_0$ and respectively $\overline{*}(ms,\oplus(\ominus(m_{x1}),x)) \overset{xy}{\hookleftarrow} m'_1$ by the induction hypothesis. Hence, we show the required equality by

$$\oplus(\overline{*}(ms,m_{x1}),\overline{*}(ms,\oplus(\ominus(m_{x1}),x))) = \overline{*}(ms,\oplus(m_{x1},\oplus(\ominus(m_{x1}),x))) = \overline{*}(ms,x).$$

- Let property $syn^\ominus(m',m'_0)$ hold. Then, $syn^{\overline{*}}(m',ms',x')$ implies $syn^\ominus(x',x'_0)$ and $syn^{\overline{*}}(m'_0,ms',x'_0)$ for some x'_0.

  First, we use $(x,x') \in xy$ and $\Psi^2_\ominus$, $\Psi^2_{\ominus,*}$ or $\Psi^2_{\ominus,\oplus}$ to deduce $(\ominus(x),\ominus(x')) \in xy$, i.e. $(\ominus(x),x'_0) \in xy$. This permits to have $\Theta^*_2(\overline{*}(ms,\ominus(x)),m'_0,xy,kb,kb')$ by definition, and then to get $\overline{*}(ms,\ominus(x)) \overset{xy}{\hookleftarrow} m'_0$ by the induction hypothesis. Hence, we show the required equality by $\ominus(\overline{*}(ms,\ominus(x))) = \overline{*}(ms,x)$. □

13.6 Handling of the non-canonical $\Xi_\oplus$ Case

The $\Xi_\oplus$-case in lemma 122 defines the mapping by decomposition for $\oplus$-objects that have available $\oplus$-parts in $DY(kb)$. Since multiple decomposition alternatives (into $\oplus$-parts) are possible, we use a definition of $\Xi_\oplus$ that covers also nested $\oplus$-parts obtained by successive decomposition. This definition employs the $\Theta^\oplus_1$ predicate to qualify the $\oplus$-parts where the successive decomposition halts.

In addition to $\Theta^\oplus_1$ and above introduced predicates, we use predicate $syn^{\overline{\oplus}}(m,ms)$ introduced in Sec. 6.5.1.1 to identify a successive decomposition of m into $\oplus$-parts in ms. Furthermore, we employ the following predicate as short-cut:

$$\Theta^\oplus_2(ms,ik) \Leftrightarrow \forall m \in ms : \Theta^\oplus_1(m,ik)$$

Accordingly, $syn^{\overline{\oplus}}(m,ms)$ and $\Theta^\oplus_2(ms,kb)$ identify in the following definition of $\Xi_\oplus$ *all* successive decompositions of m into available $\oplus$-parts, i.e. the elements of ms. At the same time, all decompositions by $\oplus$ are given by a proper partition $ms_0 \uplus ms_1$ of ms.

$$\begin{aligned}\Xi_\oplus : \quad & \exists ms \subset DY(kb), ms' \subset DY(kb') : \\ & syn^{\overline{\oplus}}(m,ms) \wedge len(ms) > 1 \wedge \Theta^\oplus_2(ms,kb) \wedge \\ & \wp(ms,ms') \subset^{xy}_{\leftrightarrow} \wedge m' = \overline{\oplus}(ms') \wedge m' \neq \infty\end{aligned}$$

The predicate $\Xi_\oplus$ defines the mapped message m' to a $\oplus$-message m (with available $\oplus$-parts) relative to the *arbitrarily many* available $\oplus$-parts (the elements of ms): m' is the result of successive applications of $\oplus$ using the messages mapped to the elements of ms.

Before we use the predicate $\Xi_\oplus$ in the proof of theorem 108 in TC-AMP (see Sec. 13.7), we describe the handling of $\oplus$-messages, i.e. the $\Xi_\oplus$-case, in the step case of lemma 122:

For $m \in DY(kb)$, $m \notin dom(xy)$ and $\neg\Theta_1^{\oplus}(m,kb)$, we must provide $m' \in DY(kb')$ satisfying $m \overset{xy}{\leftrightarrow} m'$ and $\Xi_{\oplus}(m,m',xy,kb,kb')$.

The proof starts by applying lemma 129 for *a successive decomposition of m into its available $\oplus$-parts in $DY(kb)$*. It provides $ms \subset DY(kb)$ with $syn^{\overline{\oplus}}(m,ms)$, $len(ms) > 1$ and $\Theta_2^{\oplus}(ms,kb)$. Since all elements in ms are smaller than m, we may apply the induction hypothesis to obtain $ms' \subset DY(kb')$ with $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$.

We continue the proof by setting $m' = \overline{\oplus}(ms')$ and showing $m \overset{xy}{\leftrightarrow} m'$ as follows:

- First, we apply lemma 130 *to obtain the mapping of m by $\overset{xy}{\mapsto}$ using the available $\oplus$-parts in $ms \subset DY(kb)$*. It permits to show $\overline{\oplus}(ms) \overset{xy}{\mapsto} \overline{\oplus}(ms')$, i.e. $m \overset{xy}{\mapsto} \overline{\oplus}(ms')$.

- Next, we aim at the mapping of $\overline{\oplus}(ms')$ by $\overset{xy}{\leftarrow}$ using the $\oplus$-parts of $\overline{\oplus}(ms')$ in $DY(kb')$. Since the multiset ms' does not necessarily include only $\oplus$-parts of $\overline{\oplus}(ms')$, we apply lemma 131 to obtain an equivalent multiset with the required property. For $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$, this lemma provides $\wp(ms_1,ms_1') \subset \overset{xy}{\leftrightarrow}$ such that $syn^{\overline{\oplus}}(\overline{\oplus}(ms'),ms_1')$, and $syn^{\overline{\oplus}}(\overline{\oplus}(ms),ms_1)$ hold and all mappings $m_0 \overset{xy}{\leftrightarrow} m_0'$ in $\wp(ms_1,ms_1')$ fulfill the property $\Theta_1^{\oplus}(m_0,kb)$ or $\Theta_2^{*}(m_0,m_0',xy,kb,kb')$. For the mappings $m_0 \overset{xy}{\leftrightarrow} m_0'$ where m_0 possesses non-confidential $\oplus$-parts in $DY(kb)$, i.e. where $\Theta_1^{\oplus}(m_0,kb)$ does not hold, the mapped messages m_0 and m_0' can be linked to a pair in xy used as the basis for their mapping according to the definition of Θ_2^{*}.

  Note that $syn^{\overline{\oplus}}(\overline{\oplus}(ms'),ms_1')$ ensues the required condition $\overline{\oplus}(ms') \neq \infty$.

  After obtaining ms_1 and ms_1' by lemma 131, we check whether $len(ms_1) = 1$ holds. In this case, the proof is trivially closed, as $\wp(ms_1,ms_1') = \{m \overset{xy}{\leftrightarrow} m'\}$.

  If $len(ms_1) > 1$, we prove $m \overset{xy}{\leftarrow} m'$, i.e. $\overline{\oplus}(ms_1) \overset{xy}{\leftarrow} \overline{\oplus}(ms_1')$, with the help of lemma 136, which requires a multiset consisting of all non-confidential nested $\oplus$-parts of $\overline{\oplus}(ms_1')$. Since ms_1' could include $\oplus$-objects with non-confidential $\oplus$-parts, i.e. $\Theta_2^{\oplus}(ms_1',kb')$ could not hold, we need to identify equivalent multisets for ms_1' (and ms_1) according to the following principle:

  1. Define ms_2' to be the greatest subset of ms_1' fulfilling $\Theta_2^{\oplus}(ms_2',kb')$. This permits to identify $ms_2 \subseteq ms_1$ with $\wp(ms_2,ms_2') \subset \overset{xy}{\leftrightarrow}$.

  2. For the remaining mappings $m_u \overset{xy}{\leftrightarrow} m_u'$ in $\wp(ms_1,ms_1') \setminus \wp(ms_2,ms_2')$, we know that $\Theta_1^{\oplus}(m_u',kb')$ does not hold. So, we derive for everyone of these mappings corresponding multisets ms_u and ms_u' that satisfy $\Theta_2^{\oplus}(ms_u',kb')$, $\overline{\oplus}(ms_u) = m_u$, $syn^{\overline{\oplus}}(m_u',ms_u')$, $\wp(ms_u,ms_u') \subset \overset{xy}{\leftarrow}$ and so that all mappings $m_0 \overset{xy}{\leftarrow} m_0'$ in $\wp(ms_u,ms_u')$ fulfill $\Theta_2^{*}(m_0,m_0',xy,kb,kb')$:

     We first use the structural condition of the mapping $m_u \overset{xy}{\leftrightarrow} m_u'$, i.e. $(m_u,m_u') \in xy$ or $\Xi(m_u,m_u',xy,kb,kb')$, in case $\Theta_1^{\oplus}(m_u,kb)$ to deduce $\Theta_2^{*}(m_u,m_u',xy,kb,kb')$, by lemma 133. This allows us to focus on one case, i.e. $\Theta_2^{*}(m_u,m_u',xy,kb,kb')$, which yields $syn^{\overline{*}}(m_u',ms_x',x')$, $\wp(ms_x,ms_x') \subset \overset{xy}{\leftrightarrow}$, a pair $(x,x') \in xy$ and $m_u = \overline{*}(ms_x,x)$. W.l.o.g., let x' satisfy $obj^{\overline{\oplus}}(x',\{x_0',x_1'\})$ with $\Theta_2^{\oplus}(\{x_0',x_1'\},kb')$. Then, we apply condition $\Psi_{\oplus}^2$ to the pair $(x,x') \in xy$ and derive (m_{x0},x_0') and $(\oplus(\ominus(m_{x0}),x),x_1')$ in xy. Using the resulting pairs, we get $\Theta_2^{*}(\overline{*}(ms_x,m_{x0}),\overline{*}(ms_x',x_0'),xy,kb,kb')$ together with $\Theta_2^{*}(\overline{*}(ms_x,\oplus(\ominus(m_{x0}),x)),\overline{*}(ms_x',x_1'),xy,kb,kb')$, by definition, and thus $\overline{*}(ms_x,m_{x0}) \overset{xy}{\leftarrow} \overline{*}(ms_x',x_0')$ and $\overline{*}(ms_x,\oplus(\ominus(m_{x0}),x)) \overset{xy}{\leftarrow} \overline{*}(ms_x',x_1')$, with the help of lemma 128. This permits to set

     (a) $ms_u = \{\overline{*}(ms_x,m_{x0}),\overline{*}(ms_x,\oplus(\ominus(m_{x0}),x))\}$

(b) and $ms'_u = \{\overline{*}(ms'_x, x'_0), \overline{*}(ms'_x, x'_1)\}$.

If x' possesses $n > 2$ available $\oplus$-parts, $(n-1)$ successive applications of $\Psi^2_{\oplus}$ starting as well with $(x, x') \in xy$ permit clearly to gain n corresponding pairs in xy, which we use the same way to construct ms_u and ms'_u.

3. Define ms'_3 and ms_3 to be the union of all multisets ms'_u and respectively ms_u obtained in (2). Obviously, we have $\wp(ms_3, ms'_3) \subset^{xy}_{\leftrightarrow}$, $\overline{\oplus}(ms_1) = \overline{\oplus}(ms_2 \uplus ms_3)$, $syn^{\overline{\oplus}}(\overline{\oplus}(ms'_1), ms'_2 \uplus ms'_3)$ and $\Theta^{\oplus}_2(ms'_2 \uplus ms'_3, kb')$.

Note that the transformation of $\wp(ms_1, ms'_1) \setminus \wp(ms_2, ms'_2)$ to $\wp(ms_3, ms'_3)$ yields to a new proof situation where $syn^{\overline{\oplus}}(\overline{\oplus}(ms), ms_2 \uplus ms_3)$ does not necessarily hold. For that reason, we are not able to use (the induction hypothesis of) lemma 122 at least for decompositions into $\oplus$-parts. But, (the induction hypothesis of) lemma 122 still applies for the common left $*$-sub-messages of $\overline{\oplus}(ms_2 \uplus ms_3)$, since these are also common left $*$-sub-messages of the elements in ms.

Recapitulating, the obtained $ms_2 \uplus ms_3$ and $ms'_2 \uplus ms'_3$ permit to apply lemma 136 *to obtain the mapping of* $\overline{\oplus}(ms'_2 \uplus ms'_3)$*, i.e.* $\overline{\oplus}(ms')$*, by* $\overset{xy}{\leftarrow}$ *using the identified mappings of the nested* $\oplus$*-parts in* $DY(kb')$. It permits to show $\overline{\oplus}(ms_2 \uplus ms_3) \overset{xy}{\leftarrow} \overline{\oplus}(ms'_2 \uplus ms'_3)$, and this means $m \overset{xy}{\leftarrow} \overline{\oplus}(ms')$ because $m = \overline{\oplus}(ms) = \overline{\oplus}(ms_1) = \overline{\oplus}(ms_2 \uplus ms_3)$ and $\overline{\oplus}(ms') = \overline{\oplus}(ms'_1) = \overline{\oplus}(ms'_2 \uplus ms'_3)$.

13.6.1 Decomposition into $\oplus$-Parts in $DY(kb)$

For $\oplus$-objects, the following lemma permits to identify the result of a successive decomposition into available $\oplus$-parts.

Lemma[VSE] 129 (Decomposition into $\oplus$-Parts in $DY(kb)$):
Let m satisfy $obj^{\oplus}(m, m_0, m_1)$ for m_0 and m_1 in $DY(kb)$. Then, it exists $ms \subset DY(kb)$ satisfying

$$syn^{\overline{\oplus}}(m, ms) \wedge len(ms) > 1 \wedge \Theta^{\oplus}_2(ms, kb).$$

The proof (by induction on $|m|$) is similar to the proof of lemma 111.

13.6.2 Mapping by $\overset{xy}{\mapsto}$ using the $\oplus$-Parts in $DY(kb)$

For $\oplus$-objects that are not mapped in xy, the following lemma provides the mapping by $\overset{xy}{\mapsto}$ relative to the mappings of the available $\oplus$-parts.

Lemma[VSE] 130 (Mapping of $\oplus$-Objects by $\overset{xy}{\mapsto}$):
Let $ms \subset DY(kb)$ and $ms' \subset DY(kb)$ satisfy $syn^{\overline{\oplus}}(\overline{\oplus}(ms), ms)$, $len(ms) > 1$, $\Theta^{\oplus}_2(ms, kb)$, $\wp(ms, ms') \subset^{xy}_{\leftrightarrow}$ and $\overline{\oplus}(ms) \notin dom(xy)$. Let (the induction hypothesis of) lemma 122 hold for all $\hat{m}$ with $|\hat{m}| < |\overline{\oplus}(ms)|$. Then, we have

$$\overline{\oplus}(ms) \overset{xy}{\mapsto} \overline{\oplus}(ms').$$

Since $\overline{\oplus}(ms) \notin dom(xy)$ and $len(ms) > 1$ hold, we prove $\overline{\oplus}(ms) \overset{xy}{\mapsto} \overline{\oplus}(ms')$ by showing that for any decomposition of m into $\oplus$-parts, $*$-sub-messages $m_0, m_1 \in DY(kb)$ and possibly into a $\ominus$-sub-message m_0 the composition of the corresponding mapped messages (by $\oplus$, $*$ and respectively $\ominus$) yields $\overline{\oplus}(ms')$.

13.6.2.1 Mapping by Decomposition into $\oplus$-Parts

We start with decompositions into $\oplus$-parts. That is, we assume $m_0, m_1 \in DY(kb)$ with $obj^{\oplus}(\overline{\oplus}(ms), m_0, m_1)$ and want to show that m_0' and m_1' in the mappings $m_0 \overset{xy}{\leftrightarrow} m_0'$ and $m_1 \overset{xy}{\leftrightarrow} m_1'$, which are provided by (the induction hypothesis of) lemma 122, satisfy the equality $\oplus(m_0', m_1') = \overline{\oplus}(ms')$.

Based on $obj^{\oplus}(\overline{\oplus}(ms), m_0, m_1)$, $syn^{\overline{\oplus}}(\overline{\oplus}(ms), ms)$, $ms \subset DY(kb)$, $len(ms) > 1$, $\Theta_2^{\oplus}(ms, kb)$ and $m_0, m_1 \in DY(kb)$, we obtain $ms = ms_0 \uplus ms_1$, $\Theta_2^{\oplus}(ms_0, kb)$, $\Theta_2^{\oplus}(ms_1, kb)$, $syn^{\overline{\oplus}}(m_0, ms_0)$ and $syn^{\overline{\oplus}}(m_1, ms_1)$. Furthermore, $\wp(ms, ms') \subset^{xy}_{\leftrightarrow}$ means that there is a partition $ms' = ms_0' \uplus ms_1'$ with $\wp(ms_0, ms_0'), \wp(ms_1, ms_1') \subset^{xy}_{\leftrightarrow}$. In the following, we want to show that $m_0' = \overline{\oplus}(ms_0')$ and $m_1' = \overline{\oplus}(ms_1')$ to prove the required equality by

$$\begin{aligned} \oplus(m_0', m_1') &= \oplus(\overline{\oplus}(ms_0'), \overline{\oplus}(ms_1')) \\ &= \overline{\oplus}(ms_0' \uplus ms_1') \\ &= \overline{\oplus}(ms'). \end{aligned}$$

1. If $ms_0 = \{m_2\}$ and $ms_1 = \{m_3\}$, we obtain $m_0 = m_2$, $m_1 = m_3$, $ms_0' = \{m_2'\}$ and $ms_1' = \{m_3'\}$ for $m_2 \overset{xy}{\leftrightarrow} m_2'$ and $m_3 \overset{xy}{\leftrightarrow} m_3'$. This permits to deduce $m_0' = m_2'$ and $m_1' = m_3'$ and then to show the required equalities by $\overline{\oplus}(ms_0') = \overline{\oplus}(\{m_2'\}) = m_2' = m_0'$ and by $\overline{\oplus}(ms_1') = \overline{\oplus}(\{m_3'\}) = m_3' = m_1'$.

2. If $ms_0 = \{m_2\}$ and $len(ms_1) > 1$, we first obtain $m_0 = m_2$, $ms_0' = \{m_2'\}$ for $m_2 \overset{xy}{\leftrightarrow} m_2'$, permitting to show $\overline{\oplus}(ms_0') = m_0'$ as in (1).

   $len(ms_1) > 1$, $syn^{\overline{\oplus}}(m_1, ms_1)$, $\wp(ms_1, ms_1') \subset^{xy}_{\leftrightarrow}$ and $ms_1 \subset DY(kb)$ permit to show $\overline{\oplus}(ms_1') = m_1'$ based on the structural condition for $m_1 \overset{xy}{\leftrightarrow} m_1'$ by the following case distinction:

   (a) Case $(m_1, m_1') \in xy$, i.e. $(\overline{\oplus}(ms_1), m_1') \in xy$, is handled based on condition $\Psi^1_{\oplus}$: For $ms_1 = m_2 \uplus ms_2$, we successively apply condition $\Psi^1_{\oplus}$ to obtain the pair $(m_2, \oplus(\ominus(\overline{\oplus}(ms_2'))), m_1')) \in xy$ for $ms_1' = m_2' \uplus ms_2'$, and then the equality $m_2' = \oplus(\ominus(\overline{\oplus}(ms_2')), m_1')$. This permits to show $m_1' = \oplus(m_2', \overline{\oplus}(ms_2')) = \overline{\oplus}(m_2' \uplus ms_2') = \overline{\oplus}(ms_1')$, as required.

   (b) When $\Xi(m_1, m_1', xy, kb, kb')$, structural properties on m_1 such as $syn^{\overline{\oplus}}(m_1, ms_1)$, and $len(ms_1) > 1$ permit to reduce this to the $\Xi_{\oplus}$-case. That is, we get $syn^{\overline{\oplus}}(m_1, ms_2)$, $\Theta_2^{\oplus}(ms_2, kb)$, $\wp(ms_2, ms_2') \subset^{xy}_{\leftrightarrow}$ and $m_1' = \overline{\oplus}(ms_2')$. Based on $syn^{\overline{\oplus}}(m_1, ms_1)$, $syn^{\overline{\oplus}}(m_1, ms_2)$, $\Theta_2^{\oplus}(ms_1, kb)$ and $\Theta_2^{\oplus}(ms_2, kb)$, we obtain $ms_1 = ms_2$. This yields $ms_1' = ms_2'$, permitting to rewrite $m_1' = \overline{\oplus}(ms_2')$ to $m_1' = \overline{\oplus}(ms_1')$, as required.

3. The case where $ms_1 = \{m_2\}$ and $len(ms_0) > 1$ is similar to (2).

4. If $len(ms_0), len(ms_1) > 1$, the second proof part in (2) applies. □

13.6.2.2 Mapping by Decomposition into $*$-Sub-Messages

Next, we consider arbitrary decompositions into $*$-sub-messages. That is, we assume $m_0, m_1 \in DY(kb)$ with $syn^{*}(\overline{\oplus}(ms), m_0, m_1)$ and want to show that m_0' and m_1' in the mappings $m_0 \overset{xy}{\leftrightarrow} m_0'$ and $m_1 \overset{xy}{\leftrightarrow} m_1'$, which are provided by (the induction hypothesis of) lemma 122, satisfy the equality $*(m_0', m_1') = \overline{\oplus}(ms')$.

- First, $syn^{*}(\overline{\oplus}(ms), m_0, m_1)$ implies that all messages $m_2 \in ms$ satisfy $syn^{*}(m_2, m_0, m_3)$ for $m_3 \in ms_1$ and $syn^{\overline{\oplus}}(m_1, ms_1)$.

- Then, we show (by induction on the length of ms_1) that there is $\wp(ms_1, ms_1') \subset^{xy}_{\leftrightarrow}$ such that $\overline{\oplus}(ms') = *(m_0', \overline{\oplus}(ms_1'))$ and the mappings $m_2 \overset{xy}{\leftrightarrow} m_2'$ in $\wp(ms, ms')$ with $syn^*(m_2, m_0, m_3)$ transfer to mappings $m_3 \overset{xy}{\leftrightarrow} *(inv(m_0'), m_2')$ in $\wp(ms_1, ms_1')$. The latter is based on the structural condition of $m_2 \overset{xy}{\leftrightarrow} m_2'$:
  - When $(m_2, m_2') \in xy$, i.e. $(*(m_0, m_3), m_2') \in xy$, we apply condition Ψ^1_* to deduce $(m_3, *(inv(m_0'), m_2')) \in xy$ and thus $m_3 \overset{xy}{\leftrightarrow} *(inv(m_0'), m_2')$.
  - In case $\Xi(m_2, m_2', xy, kb, kb')$, we may focus on the Ξ_*-case due to the structure of m_2, and this provides $syn^{\overline{*}}(m_2, ms_2, x_2)$, $\wp(ms_2, ms_2') \subset^{xy}_{\leftrightarrow}$, $x_2 \overset{xy}{\leftrightarrow} x_2'$ and $m_2' = \overline{*}(ms_2', x_2')$. Since $syn^*(m_2, m_0, m_3)$ holds, we have $ms_2 = \{m_0\}$ and $x_2 = m_3$ or the complementary case with $ms_2 = m_0 \uplus ms_3$ and $syn^{\overline{*}}(m_3, ms_3, x_2)$ for $ms_3 \neq \emptyset$. In the former case, $m_3 = x_2$ and $m_2' = \overline{*}(\{m_0'\}, x_2') = *(m_0', x_2')$ permit to rewrite $x_2 \overset{xy}{\leftrightarrow} x_2'$ to $m_3 \overset{xy}{\leftrightarrow} *(inv(m_0'), m_2')$. In the latter case, the structural condition of the mapping $m_3 \overset{xy}{\leftrightarrow} m_3'$, i.e. $\overline{*}(ms_3, x_2) \overset{xy}{\leftrightarrow} m_3'$, permits to show $m_3' = \overline{*}(ms_3', x_2')$ with $ms_2' = m_0' \uplus ms_3'$ and thus $*(m_0', m_3') = *(m_0', \overline{*}(ms_3', x_2')) = m_2'$, which implies $m_3 \overset{xy}{\leftrightarrow} *(inv(m_0'), m_2')$:
    - If $(\overline{*}(ms_3, x_2), m_3') \in xy$, we apply Ψ^1_* to obtain $(x_2, \overline{*}(\overline{inv}(ms_3'), m_3')) \in xy$ and then deduce $x_2' = \overline{*}(\overline{inv}(ms_3'), m_3')$ and thus $m_3' = \overline{*}(ms_3', x_2')$.
    - Otherwise, $\Xi(\overline{*}(ms_3, x_2), m_3', xy, kb, kb')$ is traced back to the Ξ_*-case, which permits to show $m_3' = \overline{*}(ms_3', x_2')$.

- Next, $syn^{\overline{\oplus}}(m_1, ms_1)$, $len(ms_1) > 1$ and $\wp(ms_1, ms_1') \subset^{xy}_{\leftrightarrow}$ permit to obtain the mapping $m_1 \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_1')$ as described in Sec. 13.6.2.1, case (2).

- Hence, we show the required equality by $*(m_0', m_1') = *(m_0', \overline{\oplus}(ms_1')) = \overline{\oplus}(ms')$. □

13.6.2.3 Mapping by Decomposition into a $\ominus$-Sub-Message

We continue the proof with a possible decomposition into a $\ominus$-sub-message. That is, we assume $m_0 \in DY(kb)$ with $syn^{\ominus}(\overline{\oplus}(ms), m_0)$ and want to show that m_0' in the mapping $m_0 \overset{xy}{\leftrightarrow} m_0'$, which is provided by (the induction hypothesis of) lemma 122, satisfies the equality $\ominus(m_0') = \overline{\oplus}(ms')$.

- First, $syn^{\ominus}(\overline{\oplus}(ms), m_0)$ implies that all messages $m_1 \in ms$ satisfy $syn^{\ominus}(m_1, m_2)$ for $m_2 \in ms_0$ and $syn^{\overline{\oplus}}(m_0, ms_0)$.

- Then, we show (by induction on the length of ms_0) that there is $\wp(ms_0, ms_0') \subset^{xy}_{\leftrightarrow}$ such that $\overline{\oplus}(ms') = \ominus(\overline{\oplus}(ms_0'))$ and the mappings $m_1 \overset{xy}{\leftrightarrow} m_1'$ in $\wp(ms, ms')$ with $syn^{\ominus}(m_1, m_2)$ transfer to mappings $m_2 \overset{xy}{\leftrightarrow} \ominus(m_1')$ in $\wp(ms_0, ms_0')$. The latter is based on the structural condition of $m_1 \overset{xy}{\leftrightarrow} m_1'$:
  - When $(m_1, m_1') \in xy$, i.e. $(\ominus(m_2), m_1') \in xy$, we deduce $(\ominus(\ominus(m_2)), \ominus(m_1')) \in xy$, i.e. $(m_2, \ominus(m_1')) \in xy$, and thus $m_2 \overset{xy}{\leftrightarrow} \ominus(m_1')$, with the help of $\Psi^1_{\ominus}$, $\Psi^1_{\ominus,*}$ or $\Psi^1_{\ominus,\oplus}$.
  - In case $\Xi(m_1, m_1', xy, kb, kb')$, the $syn^{\ominus}$-structure of m_1 and $\Theta^{\oplus}_1(m_1, kb)$ (ensued from $\Theta^{\oplus}_2(ms, kb)$) reduces this to the $\Xi_{\ominus}$- or the Ξ_*-case.

    In the $\Xi_{\ominus}$-case, we get $obj^{\ominus}(m_1, m_3)$, $m_3 \overset{xy}{\leftrightarrow} m_3'$ and $obj^{\ominus}(m_1', m_3')$, i.e. $m_1' = \ominus(m_3')$. Based on $syn^{\ominus}(m_1, m_2)$, we deduce $m_2 = m_3$ and $m_2 \overset{xy}{\leftrightarrow} \ominus(m_1')$.

In the Ξ_*-case, we get the corresponding structural properties $syn^{\overline{*}}(m_1, ms_1, x_1)$, $\wp(ms_1, ms_1') \subset^{xy}_{\leftrightarrow}$, $x_1 \overset{xy}{\leftrightarrow} x_1'$ and $m_1' = \overline{*}(ms_1', x_1')$. Since $syn^{\ominus}(m_1, m_2)$ holds, we obtain $syn^{\ominus}(x_1, x_2)$ and $syn^{\overline{*}}(m_2, ms_1, x_2)$. Then, we expand the structural condition of $x_1 \overset{xy}{\leftrightarrow} x_1'$ to obtain $x_2 \overset{xy}{\leftrightarrow} \ominus(x_1')$. After that, the structural condition of $m_2 \overset{xy}{\leftrightarrow} m_2'$, i.e. $\overline{*}(ms_1, x_2) \overset{xy}{\leftrightarrow} m_2'$, permits to show $m_2' = \overline{*}(ms_1', \ominus(x_1'))$ and thus $m_2' = \ominus(\overline{*}(ms_1', x_1')) = \ominus(m_1')$, which implies $m_2 \overset{xy}{\leftrightarrow} \ominus(m_1')$:

* If $(\overline{*}(ms_1, x_2), m_2') \in xy$, we apply Ψ^1_* to obtain $(x_2, \overline{*}(\overline{inv}(ms_1'), m_2')) \in xy$ and then deduce $\ominus(x_1') = \overline{*}(\overline{inv}(ms_1'), m_2')$ and thus $m_2' = \overline{*}(ms_1', \ominus(x_1'))$.
* Otherwise, $\Xi(\overline{*}(ms_1, x_2), m_2', xy, kb, kb')$ is traced back to the Ξ_*-case, which permits to show $m_2' = \overline{*}(ms_1', \ominus(x_1'))$.

- Next, $syn^{\overline{\oplus}}(m_0, ms_0)$, $len(ms_0) > 1$ and $\wp(ms_0, ms_0') \subset^{xy}_{\leftrightarrow}$ permit to obtain the mapping $m_0 \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_0')$ as described in Sec. 13.6.2.1, case (2).
- Hence, we show the required equality by $\ominus(m_0') = \ominus(\overline{\oplus}(ms_0')) = \overline{\oplus}(ms')$. □

13.6.3 Transformation into $\oplus$-Parts in $DY(kb')$

Given $\wp(ms, ms') \subset^{xy}_{\leftrightarrow}$ with $syn^{\overline{\oplus}}(\overline{\oplus}(ms), ms)$, the following lemma permits to check whether ms' fulfills $syn^{\overline{\oplus}}(\overline{\oplus}(ms'), ms')$. Otherwise, it permits to successively transform mapping pairs $m_0 \overset{xy}{\leftrightarrow} m_0'$ and $m_1 \overset{xy}{\leftrightarrow} m_1'$ where $obj^{\oplus}(\oplus(m_0', m_1'), m_0', m_1')$ does not hold to composed mappings $\oplus(m_0, m_1) \overset{xy}{\leftrightarrow} \oplus(m_0', m_1')$ satisfying the Θ^*_2 condition. Recall that Θ^*_2 links the mapping to a pair in the basis relation xy, which is useful to derive further related mappings if required in the rest of the (main) proof.

We use lemma 131 in the main proof to transform $\wp(ms, ms') \subset^{xy}_{\leftrightarrow}$ into $\wp(ms_1, ms_1') \subset^{xy}_{\leftrightarrow}$ such that $syn^{\overline{\oplus}}(\overline{\oplus}(ms'), ms_1')$, and $syn^{\overline{\oplus}}(\overline{\oplus}(ms), ms_1)$ hold and all mappings $m_0 \overset{xy}{\leftrightarrow} m_0'$ in $\wp(ms_1, ms_1')$ fulfill $\Theta^{\oplus}_1(m_0, kb)$ or $\Theta^*_2(m_0, m_0', xy, kb, kb')$.

As reflected in the induction proof of the lemma, the transformation procedure is recursive, where the very first transformed mappings $m_0 \overset{xy}{\leftrightarrow} m_0'$ and $m_1 \overset{xy}{\leftrightarrow} m_1'$ satisfy $\Theta^{\oplus}_1(m_0, kb)$ and $\Theta^{\oplus}_1(m_1, kb)$. In subsequent transformations, used mappings could result from previous transformations and could for that reason satisfy the Θ^*_2-predicate.

Lemma$^{\text{VSE}}$ 131 (Transformation into $\oplus$-Parts in $DY(kb')$):
Let $ms \subset DY(kb)$ and $ms' \subset DY(kb')$ satisfy $len(ms) > 1$, $syn^{\overline{\oplus}}(\overline{\oplus}(ms), ms)$, $\wp(ms, ms') \subset^{xy}_{\leftrightarrow}$ and $\overline{\oplus}(ms) \overset{xy}{\mapsto} \overline{\oplus}(ms')$. Let every mapping $m \overset{xy}{\leftrightarrow} m'$ in $\wp(ms, ms')$ fulfill $\Theta^{\oplus}_1(m, kb)$ or $\Theta^*_2(m, m', xy, kb, kb')$. And let (the induction hypothesis of) lemma 122 hold for all $m \in ms$ and for all $\hat{m}$ with $|\hat{m}| < |\overline{\oplus}(ms)|$. Then, there exists $ms_1 \subset DY(kb)$ and $ms_1' \subset DY(kb')$ such that

$$\wp(ms_1, ms_1') \subset^{xy}_{\leftrightarrow} \wedge syn^{\overline{\oplus}}(\overline{\oplus}(ms), ms_1) \wedge syn^{\overline{\oplus}}(\overline{\oplus}(ms'), ms_1') \wedge$$
$$(\forall m \overset{xy}{\leftrightarrow} m' \in \wp(ms_1, ms_1') : \Theta^{\oplus}_1(m, kb) \vee \Theta^*_2(m, m', xy, kb, kb')).$$

Proof: The proof is by induction on $len(ms)$.

Base Case: For $ms = \{m_0, m_1\}$, we obtain $ms' = \{m_0', m_1'\}$ with $m_0 \overset{xy}{\leftrightarrow} m_0'$ and $m_1 \overset{xy}{\leftrightarrow} m_1'$. If $syn^{\overline{\oplus}}(\overline{\oplus}(ms'), ms')$ holds, we conclude by setting $ms_1 = ms$ and $ms_1' = ms'$.

Otherwise, we have $\neg obj^{\oplus}(\oplus(m'_0,m'_1),m'_0,m'_1)$ and $\oplus(m'_0,m'_1) \neq \infty$. The latter holds, because $\oplus(m'_0,m'_1) = \infty$ permits to refute $obj^{\oplus}(\oplus(m_0,m_1),m_0,m_1)$, i.e. $syn^{\overline{\oplus}}(\overline{\oplus}(ms),ms)$: Assuming $\oplus(m'_0,m'_1) = \infty$ yields $m'_1 = \ominus(m'_0)$ and thus $m_1 \overset{xy}{\leftrightarrow} \ominus(m'_0)$, which permits to get $\ominus(m_1) \overset{xy}{\leftrightarrow} m'_0$ with the help of lemma 132 and then to deduce $m_0 = \ominus(m_1)$.

That is, $\neg obj^{\oplus}(\oplus(m'_0,m'_1),m'_0,m'_1)$ reduces to two cases:

1. $obj^{\oplus}(m'_0,m'_2,m')$ and $m'_1 = \ominus(m')$: We first use the structural condition of the mapping $m_0 \overset{xy}{\leftrightarrow} \oplus(m'_2,m')$ in case $\Theta_1^{\oplus}(m_0,kb)$ to deduce $\Theta_2^*(m_0,\oplus(m'_2,m'),xy,kb,kb')$, by lemma 133. This permits to focus on one case, i.e. $\Theta_2^*(m_0,\oplus(m'_2,m'),xy,kb,kb')$. After that, we first apply lemma 132 to deduce $\ominus(m_1) \overset{xy}{\leftrightarrow} m'$ from $m_1 \overset{xy}{\leftrightarrow} \ominus(m')$, i.e. $m_1 \overset{xy}{\leftrightarrow} m'_1$, and then employ lemma 134 to obtain $\Theta_2^*(\oplus(m_0,m_1),m'_2,xy,kb,kb')$.

2. $obj^{\oplus}(m'_0,m'_2,m')$ and $obj^{\oplus}(m'_1,m'_3,\ominus(m'))$: We first use the structural conditions of the mapping $m_0 \overset{xy}{\leftrightarrow} \oplus(m'_2,m')$ in case $\Theta_1^{\oplus}(m_0,kb)$ and then of the mapping $m_1 \overset{xy}{\leftrightarrow} \oplus(m'_3,\ominus(m'))$ in case $\Theta_1^{\oplus}(m_1,kb)$ to deduce $\Theta_2^*(m_0,\oplus(m'_2,m'),xy,kb,kb')$ and respectively $\Theta_2^*(m_1,\oplus(m'_3,\ominus(m')),xy,kb,kb')$, by lemma 133. This permits to focus on one case, i.e. $\Theta_2^*(m_0,\oplus(m'_2,m'),xy,kb,kb')$ and $\Theta_2^*(m_1,\oplus(m'_3,\ominus(m')),xy,kb,kb')$, and to apply lemma 135 to obtain $\Theta_2^*(\oplus(m_0,m_1),\oplus(m'_2,m'_3),xy,kb,kb')$.

In both cases, we apply lemma 128 using the obtained Θ_2^* property to get a corresponding mapping by $\overset{xy}{\leftarrow}$, i.e. $\oplus(m_0,m_1) \overset{xy}{\leftarrow} m'_2$ in case (1) and $\oplus(m_0,m_1) \overset{xy}{\leftarrow} \oplus(m'_2,m'_3)$ in case (2). These, together with $\overline{\oplus}(ms) \overset{xy}{\mapsto} \overline{\oplus}(ms')$ yield $\oplus(m_0,m_1) \overset{xy}{\leftrightarrow} m'_2$ and respectively $\oplus(m_0,m_1) \overset{xy}{\leftrightarrow} \oplus(m'_2,m'_3)$.

Hence, we conclude by setting $ms_1 = \{\oplus(m_0,m_1)\}$ together with $ms'_1 = \{m'_2\}$ in case (1) and $ms'_1 = \{\oplus(m'_2,m'_3)\}$ in case (2). Here, the Θ_2^* property for the element of ms'_1 implies $syn^{\overline{\oplus}}(\overline{\oplus}(ms'),ms'_1)$.

Step Case: For multiset ms that satisfies $len(ms) > 2$, we proceed as in the Base Case if $syn^{\overline{\oplus}}(\overline{\oplus}(ms'),ms')$ holds.

Otherwise, we have $\neg obj^{\oplus}(\oplus(m'_0,m'_1),m'_0,m'_1)$ for $m'_0,m'_1 \in ms'$ and $m_0,m_1 \in ms$ with $m_0 \overset{xy}{\leftrightarrow} m'_0$ and $m_1 \overset{xy}{\leftrightarrow} m'_1$. Since $\oplus(m_0,m_1)$ is smaller than $\overline{\oplus}(ms)$, we apply (the induction hypothesis of) lemma 122 to get $\oplus(m_0,m_1) \overset{xy}{\leftrightarrow} \oplus(m'_0,m'_1)$. The properties about the structure of $\oplus(m'_0,m'_1)$ are shown similar to the Base Case: That is, $\oplus(m'_0,m'_1) \neq \infty$ holds and we distinguish two cases:

1. In case $obj^{\oplus}(m'_0,m'_2,m')$ and $m'_1 = \ominus(m')$ hold, we obtain the equality $\oplus(m'_0,m'_1) = m'_2$ and property $\Theta_2^*(\oplus(m_0,m_1),m'_2,xy,kb,kb')$.

2. In case $obj^{\oplus}(m'_0,m'_2,m')$ and $obj^{\oplus}(m'_1,m'_3,\ominus(m'))$, we obtain $\oplus(m'_0,m'_1) = \oplus(m'_2,m'_3)$ and $\Theta_2^*(\oplus(m_0,m_1),\oplus(m'_2,m'_3),xy,kb,kb')$.

In both cases, we use ms_{01} and ms'_{01} satisfying $ms = \{m_0,m_1\} \uplus ms_{01}$ and $ms' = \{m'_0,m'_1\} \uplus ms'_{01}$ to obtain $\wp(\oplus(m_0,m_1) \uplus ms_{01},\oplus(m'_0,m'_1) \uplus ms'_{01}) \subset \overset{xy}{\leftrightarrow}$ where $len(\oplus(m_0,m_1) \uplus ms_{01}) < len(ms)$ and in particular $syn^{\overline{\oplus}}(\overline{\oplus}(ms),\oplus(m_0,m_1) \uplus ms_{01})$ allow us to conclude by the induction hypothesis. □

13.6.3.1 Lemma on $\ominus$ Application in Mappings

The following lemma lifts the inclusion principle by $\Psi^2_{\ominus}$ to mappings where the right-side results by an application of $\ominus$: Having $m \overset{xy}{\leftrightarrow} \ominus(m')$, we also have $\ominus(m) \overset{xy}{\leftrightarrow} m'$.

This lemma is employed in particular in the proof of lemma 131.

Lemma[VSE] 132 ($\overset{xy}{\leftrightarrow}$ and $\ominus$):
Let m be in $DY(kb) \setminus \{\infty\}$ and m' in $DY(kb') \setminus \{\infty\}$ and let (the induction hypothesis of) lemma 122 hold for all $\hat{m}$ with $|\hat{m}| \leq |m|, |\ominus(m)|$. Then, we have

$$m \overset{xy}{\leftrightarrow} \ominus(m') \Rightarrow \ominus(m) \overset{xy}{\leftrightarrow} m'.$$

Proof: The proof is based on lemma 122, which provides the structural conditions for the assumed mapping $m \overset{xy}{\leftrightarrow} \ominus(m')$ and for a mapping $\ominus(m) \overset{xy}{\leftrightarrow} m'_\ominus$ that we use to show $m'_\ominus = m'$:

1. In case $(\ominus(m), m'_\ominus) \in xy$, we proceed by the following case distinction:
   (a) If $isObj^\ominus(\ominus(m))$, $isObj^*(\ominus(m))$ or $isObj^\oplus(\ominus(m))$ holds, we use $\Psi^1_\ominus$, $\Psi^1_{\ominus,*}$ or respectively $\Psi^1_{\ominus,\oplus}$ to obtain $(\ominus(\ominus(m)), \ominus(m'_\ominus)) \in xy$, i.e. $(m, \ominus(m'_\ominus)) \in xy$, and then deduce $\ominus(m'_\ominus) = \ominus(m')$, which implies $m'_\ominus = m'$, as required.
   (b) Otherwise, $m \neq \infty$ implies $isObj^\ominus(m)$ and the structural condition of $m \overset{xy}{\leftrightarrow} \ominus(m')$ yields three cases:
      i. In case $(m, \ominus(m')) \in xy$, we use $\Psi^1_\ominus$ to obtain $(\ominus(m), \ominus(\ominus(m'))) \in xy$, i.e. $(\ominus(m), m') \in xy$, which implies $m' = m'_\ominus$, as required.
      ii. In the $\Xi_\ominus$-case, we have $obj^\ominus(m, m_0)$, $m_0 \overset{xy}{\leftrightarrow} m'_0$ and $obj^\ominus(\ominus(m'), m'_0)$. This implies $m_0 = \ominus(m)$, $m'_0 = m'_\ominus$ and $\ominus(m') = \ominus(m'_\ominus)$, which rewrites to the required equality $m' = m'_\ominus$.
      iii. In the Ξ_*-case, we have $syn^{\overline{*}}(m, ms, x)$, $\Theta^*_1(x, kb)$, $\Theta^\oplus_1(x, kb)$, $\wp(ms, ms') \subset \overset{xy}{\leftrightarrow}$, $x \overset{xy}{\leftrightarrow} x'$ and $\ominus(m') = \overline{*}(ms', x')$. Furthermore, $isObj^\ominus(m)$ implies $isObj^\ominus(x)$, which we use to deduce $obj^\ominus(x, x_0)$, $x_0 \overset{xy}{\leftrightarrow} x'_0$ and $x' = \ominus(x'_0)$. Based on $syn^{\overline{*}}(m, ms, x)$ and $obj^\ominus(x, x_0)$, we get $syn^{\overline{*}}(\ominus(m), ms, x_0)$. After that, we want to deduce $m'_\ominus = \overline{*}(ms', x'_0)$, which permits to show $m'_\ominus = m'$ by $\overline{*}(ms', x'_0) = \overline{*}(ms', \ominus(x')) = \ominus(\overline{*}(ms', x')) = \ominus(\ominus(m')) = m'$:
         - When $(\ominus(m), m'_\ominus) \in xy$, i.e. $(\overline{*}(ms, x_0), m'_\ominus) \in xy$, we use Ψ^1_* to instantiate $m'_\ominus$ with $\overline{*}(ms', x'_0)$, as required.
         - Otherwise, the Ξ_*-case applies, which yields the structural properties $syn^{\overline{*}}(\ominus(m), ms_\ominus, x_\ominus)$, $\Theta^*_1(x_\ominus, kb)$, $\Theta^\oplus_1(x_\ominus, kb)$, $\wp(ms_\ominus, ms'_\ominus) \subset \overset{xy}{\leftrightarrow}$, $x_\ominus \overset{xy}{\leftrightarrow} x'_\ominus$ and $m'_\ominus = \overline{*}(ms'_\ominus, x'_\ominus)$. This implies $ms_\ominus = ms$ and $x_\ominus = x_0$, based on $syn^{\overline{*}}(\ominus(m), ms, x_0)$, $obj^\ominus(x, x_0)$ and $\Theta^*_1(x, kb)$. That is, $ms'_\ominus = ms'$, $x'_\ominus = x'_0$ and thus $m'_\ominus = \overline{*}(ms', x'_0)$, as required.

2. When any Ξ_f-case applies for $\ominus(m) \overset{xy}{\leftrightarrow} m'_\ominus$ with $f \notin \{\ominus, *, \oplus\}$, we have $isObj^\ominus(m)$. This permits to show $m' = m'_\ominus$ as in case (1-b).

3. When the $\Xi_\ominus$-case applies for $\ominus(m) \overset{xy}{\leftrightarrow} m'_\ominus$, we have $obj^\ominus(\ominus(m), m_0)$, $m_0 \overset{xy}{\leftrightarrow} m'_0$ and $obj^\ominus(m'_\ominus, m'_0)$. Since m_0 must equal m, we deduce $m'_0 = \ominus(m')$ and thus $m'_\ominus = \ominus(m'_0) = \ominus(\ominus(m')) = m'$, as required.

4. When the Ξ_*-case applies for $\ominus(m) \overset{xy}{\leftrightarrow} m'_\ominus$, we have $syn^{\overline{*}}(\ominus(m), ms_\ominus, x_\ominus)$, $\Theta^*_1(x_\ominus, kb)$, $\Theta^\oplus_1(x_\ominus, kb)$, $\wp(ms_\ominus, ms'_\ominus) \subset \overset{xy}{\leftrightarrow}$, $x_\ominus \overset{xy}{\leftrightarrow} x'_\ominus$, $m'_\ominus = \overline{*}(ms'_\ominus, x'_\ominus)$ and $isSyn^*(m'_\ominus)$. This implies $x'_\ominus \neq \infty$, which permits to provide $u \in DY(kb') \setminus \{\infty\}$ such that $x'_\ominus = \ominus(u)$ and then to rewrite $x_\ominus \overset{xy}{\leftrightarrow} x'_\ominus$ to $x_\ominus \overset{xy}{\leftrightarrow} \ominus(u)$. Based on $\Theta^*_1(x_\ominus, kb)$ and $\Theta^\oplus_1(x_\ominus, kb)$, we may ensue $\ominus(x_\ominus) \overset{xy}{\leftrightarrow} u$ from $x_\ominus \overset{xy}{\leftrightarrow} \ominus(u)$, replaying the proofs in (1)–(3), because the Ξ_*- and the $\Xi_\oplus$-case do not apply for $\ominus(x_\ominus) \overset{xy}{\leftrightarrow} u$.

Based on $syn^{\overline{*}}(\ominus(m),ms_\ominus,x_\ominus)$, we have $syn^{\overline{*}}(m,ms_\ominus,\ominus(x_\ominus))$. We want to deduce $\ominus(m') = \overline{*}(ms'_\ominus,u)$, which permits to show the required equality $m'_\ominus = m'$ by $m'_\ominus = \overline{*}(ms'_\ominus,x'_\ominus) = \overline{*}(ms'_\ominus,\ominus(u)) = \ominus(\overline{*}(ms'_\ominus,u)) = \ominus(\ominus(m')) = m'$:

- When $(m,\ominus(m')) \in xy$, i.e. $(\overline{*}(ms_\ominus,\ominus(x_\ominus)),\ominus(m')) \in xy$, we use Ψ^1_* to instantiate $\ominus(m')$ with $\overline{*}(ms'_\ominus,u)$, as required.
- Otherwise, the Ξ_*-case applies, where we use the corresponding structural properties $syn^{\overline{*}}(m,ms_\ominus,\ominus(x_\ominus))$, $\wp(ms_\ominus,ms'_\ominus) \subset \overset{xy}{\leftrightarrow}$ and $\ominus(x_\ominus) \overset{xy}{\leftrightarrow} u$ to deduce $\ominus(m') = \overline{*}(ms'_\ominus,u)$, as in case (1-(b)-iii).

5. When the $\Xi_\oplus$-case applies for $\ominus(m) \overset{xy}{\leftrightarrow} m'_\ominus$, we have $syn^{\overline{\oplus}}(\ominus(m),ms)$, $\Theta^\oplus_2(ms,kb)$, $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$ and $m'_\ominus = \overline{\oplus}(ms')$. Based on $syn^{\overline{\oplus}}(\ominus(m),ms)$, we get $syn^{\overline{\oplus}}(m,\overline{\ominus}(ms))$ and all mappings $m_x \overset{xy}{\leftrightarrow} m'_x$ in $\wp(ms,ms')$ satisfy $m_x \neq \infty$ and $m'_x \neq \infty$ and associate to mappings $\ominus(m_x) \overset{xy}{\leftrightarrow} \ominus(m'_x)$ in $\wp(\overline{\ominus}(ms),\overline{\ominus}(ms'))$: For a given $m_x \overset{xy}{\leftrightarrow} m'_x$, it exists $m'_u \in DY(kb') \setminus \{\infty\}$ such that $m'_x = \ominus(m'_u)$ and $m_x \overset{xy}{\leftrightarrow} m'_x$ rewrites to $m_x \overset{xy}{\leftrightarrow} \ominus(m'_u)$. Based on $\Theta^\oplus_2(ms,kb)$, which implies $\Theta^\oplus_1(m_x,kb)$, we may ensue $\ominus(m_x) \overset{xy}{\leftrightarrow} m'_u$ from $m_x \overset{xy}{\leftrightarrow} \ominus(m'_u)$, replaying the proofs in (1)–(4), because the $\Xi_\oplus$-case does not apply for $\ominus(m_x) \overset{xy}{\leftrightarrow} m'_u$. The set $\wp(\overline{\ominus}(ms),\overline{\ominus}(ms'))$ consists obviously of the ensued mappings $\ominus(m_x) \overset{xy}{\leftrightarrow} m'_u$, where $m'_u = \ominus(m'_x)$ follows from $m'_x = \ominus(m'_u)$.
Now, we want to deduce $\ominus(m') = \overline{\oplus}(\overline{\ominus}(ms'))$, which permits to show the required equality $m'_\ominus = m'$ by $m'_\ominus = \overline{\oplus}(ms') = \ominus(\overline{\oplus}(\overline{\ominus}(ms'))) = \ominus(\ominus(m')) = m'$:

- When $(m,\ominus(m')) \in xy$, i.e. $(\overline{\oplus}(\overline{\ominus}(ms)),\ominus(m')) \in xy$, we use $\Psi^1_\oplus$ to instantiate $\ominus(m')$ with $\overline{\oplus}(\overline{\ominus}(ms'))$, as required.
- Otherwise, the $\Xi_\oplus$-case applies, which yields the corresponding structural properties $syn^{\overline{\oplus}}(m,ms_1)$, $\Theta^\oplus_2(ms_1,kb)$, $\wp(ms_1,ms'_1) \subset \overset{xy}{\leftrightarrow}$ and $\ominus(m') = \overline{\oplus}(ms'_1)$. This implies $ms_1 = \overline{\ominus}(ms)$, based on $syn^{\overline{\oplus}}(m,\overline{\ominus}(ms))$ and $\Theta^\oplus_2(ms,kb)$. That is, $ms'_1 = \overline{\ominus}(ms')$ and thus $\ominus(m') = \overline{\oplus}(\overline{\ominus}(ms'))$, as required. □

13.6.3.2 Derivations of Θ^*_2-Conditions

The proof of lemma 131 mainly transforms pairs of mappings where the application of $\oplus$ is not constructor-type to single mappings. The transformation is based on derivations for the Θ^*_2-condition, using lemmata described in this section.

The first lemma permits to ensue the Θ^*_2-condition for mappings where the left side satisfies the $\Theta^\oplus_1$-condition and the right side is a $\oplus$-object.

Lemma$^{\text{VSE}}$ 133 ($\Theta^\oplus_1$ and $isObj^\oplus$ implies Θ^*_2):
Let $m \in DY(kb)$ and $m' \in DY(kb')$ satisfy $m \overset{xy}{\leftrightarrow} m'$, $\Theta^\oplus_1(m,kb)$ and $isObj^\oplus(m')$. And let (the induction hypothesis of) lemma 122 hold for all $\hat{m}$ with $|\hat{m}| \leq |m|$. Then, we have

$$\Theta^*_2(m,m',xy,kb,kb').$$

Proof: The proof is by case distinction:

- If $(m,m') \in xy$, we just need to have $syn^{\overline{*}}(m',\emptyset,m')$, which holds due to $isObj^\oplus(m')$.
- Otherwise, $\Xi(m,m',xy,kb,kb')$ reduces to the Ξ_*-case, based on $\Theta^\oplus_1(m,kb)$ and $isObj^\oplus(m')$. This provides $syn^{\overline{*}}(m,ms,x)$, $\Theta^*_1(x,kb)$, $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$, $x \overset{xy}{\leftrightarrow} x'$ and $m' = \overline{*}(ms',x')$. Based on $isObj^\oplus(m')$, we obtain $isObj^\oplus(x')$ and this permits to reduce

the structural condition of $x \overset{xy}{\leftrightarrow} x'$ to $(x,x') \in xy$. After that, we are able to provide a (not necessarily proper) partition $ms' = ms'_x \uplus ms'_y$ satisfying $syn^{\overline{*}}(m', ms'_x, \overline{*}(ms'_y, x'))$ and $(\overline{*}(ms_y, x), \overline{*}(ms'_y, x')) \in xy$ (cp. the proof of lemma 126). Hence, we have all we need to show $\Theta^*_2(m, m', xy, kb, kb')$ by definition. □

The next lemma permits to derive the Θ^*_2-condition for a transformation by a decrypt-type simplification.

Lemma$^{\text{VSE}}$ **134 (Θ^*_2 for $dec^{\oplus}_{\oplus}$):**
Let xy be a finite set of message pairs representing a relation between $DY(kb)$ and $DY(kb')$ that satisfies Ψ. For $m_0, m_1 \in DY(kb)$ and $m'_0, m' \in DY(kb')$, we have

$$(\Theta^*_2(m_0, \oplus(m'_0, m'), xy, kb, kb') \wedge obj^{\oplus}(\oplus(m'_0, m'), m'_0, m') \wedge \ominus(m_1) \overset{xy}{\leftrightarrow} m')$$
$$\Rightarrow \Theta^*_2(\oplus(m_0, m_1), m'_0, xy, kb, kb').$$

Proof: The proof starts by expanding the above definition of $\Theta^*_2(m_0, \oplus(m'_0, m'), xy, kb, kb')$ to obtain $syn^{\overline{*}}(\oplus(m'_0, m'), ms', \oplus(x'_0, x'))$, $\wp(ms, ms') \subset \overset{xy}{\leftrightarrow}$, $(x_0, \oplus(x'_0, x')) \in xy$ and $m_0 = \overline{*}(ms, x_0)$. Then, we employ condition $\Psi^2_{\oplus}$ to deduce $(m_{x1}, x'_0), (\oplus(x_0, \ominus(m_{x1})), x') \in xy$ from $(x_0, \oplus(x'_0, x')) \in xy$. Since $syn^{\overline{*}}(\oplus(m'_0, m'), ms', \oplus(x'_0, x'))$ implies $syn^{\overline{*}}(m', ms', x')$, we can show $\Theta^*_2(\overline{*}(ms, \oplus(x_0, \ominus(m_{x1}))), m', xy, kb, kb')$, by definition. This permits to obtain $\overline{*}(ms, \oplus(x_0, \ominus(m_{x1}))) \overset{xy}{\leftarrow} m'$ with the help of lemma 128. After that, we use $\ominus(m_1) \overset{xy}{\leftrightarrow} m'$ to deduce $\overline{*}(ms, \oplus(x_0, \ominus(m_{x1}))) = \ominus(m_1)$ and thus $m_{x1} = \oplus(x_0, \overline{*}(\overline{inv}(ms), m_1))$. This permits to rewrite $(m_{x1}, x'_0) \in xy$ to $(\oplus(x_0, \overline{*}(\overline{inv}(ms), m_1)), x'_0) \in xy$. Since $\oplus(m_0, m_1) = \overline{*}(ms, \oplus(x_0, \overline{*}(\overline{inv}(ms), m_1)))$, $m'_0 = \overline{*}(ms', x'_0)$ and $syn^{\overline{*}}(\overline{*}(ms', x'_0), ms', x'_0)$ ensues from $syn^{\overline{*}}(\oplus(m'_0, m'), ms', \oplus(x'_0, x'))$, we have $\Theta^*_2(\oplus(m_0, m_1), m'_0, xy, kb, kb')$, by definition. □

The last lemma permits to derive the Θ^*_2-condition for a transformation by a merge simplification.

Lemma$^{\text{VSE}}$ **135 (Θ^*_2 for $mrg^{\oplus}$):**
Let xy be a finite set of message pairs representing a relation between $DY(kb)$ and $DY(kb')$ that satisfies Ψ. For $m_0, m_1 \in DY(kb)$ and $\oplus(m'_0, m'), \oplus(m'_1, \ominus(m')) \in DY(kb')$, we have

$$(\Theta^*_2(m_0, \oplus(m'_0, m'), xy, kb, kb') \wedge obj^{\oplus}(\oplus(m'_0, m'), m'_0, m') \wedge$$
$$\Theta^*_2(m_1, \oplus(m'_1, \ominus(m')), xy, kb, kb') \wedge obj^{\oplus}(\oplus(m'_1, \ominus(m')), m'_1, \ominus(m')))$$
$$\Rightarrow \Theta^*_2(\oplus(m_0, m_1), \oplus(m'_0, m'_1), xy, kb, kb').$$

Proof: The proof starts by expanding the above definition of $\Theta^*_2(m_0, \oplus(m'_0, m'), xy, kb, kb')$ to obtain these structural properties $syn^{\overline{*}}(\oplus(m'_0, m'), ms'_0, \oplus(x'_0, x'_2))$, $\wp(ms_0, ms'_0) \subset \overset{xy}{\leftrightarrow}$, $(x_0, \oplus(x'_0, x'_2)) \in xy$ and $m_0 = \overline{*}(ms_0, x_0)$. Next, we expand $\Theta^*_2(m_1, \oplus(m'_1, \ominus(m')), xy, kb, kb')$ to get $syn^{\overline{*}}(\oplus(m'_1, \ominus(m')), ms'_1, \oplus(x'_1, \ominus(x'_3)))$, $\wp(ms_1, ms'_1) \subset \overset{xy}{\leftrightarrow}$, $(x_1, \oplus(x'_1, \ominus(x'_3))) \in xy$ and the equality $m_1 = \overline{*}(ms_1, x_1)$.

W.l.o.g., we assume $ms'_0 = ms'_{01} \uplus ms'_2$, $ms'_1 = ms'_{01} \uplus ms'_3$, $x'_2 = \overline{*}(ms'_3, x'_{23})$ and $x'_3 = \overline{*}(ms'_2, x'_{23})$. If $ms'_3 \neq \emptyset$ holds, we deduce $(\overline{*}(\overline{inv}(ms_3), x_0), \oplus(\overline{*}(\overline{inv}(ms'_3), x'_0), x'_{23})) \in xy$ from $(x_0, \oplus(x'_0, x'_2)) \in xy$, i.e. $(x_0, \oplus(x'_0, \overline{*}(ms'_3, x'_{23}))) \in xy$, based on Ψ^2_* and/or $\Psi^2_{*,\oplus}$. Similarly, if $ms'_2 \neq \emptyset$, we deduce $(\overline{*}(\overline{inv}(ms_2), x_1), \oplus(\overline{*}(\overline{inv}(ms'_2), x'_1), \ominus(x'_{23}))) \in xy$ from $(x_1, \oplus(x'_1, \ominus(x'_3))) \in xy$, i.e. $(x_1, \oplus(x'_1, \overline{*}(ms'_2, \ominus(x'_{23})))) \in xy$, based on Ψ^2_* and/or $\Psi^2_{*,\oplus}$. Then, we use the obtained pairs and apply the necessary condition $\Psi^2_{\oplus,mrg}$.

Applying this condition for the pair $(\overline{*}(\overline{inv}(ms_3),x_0),\oplus(\overline{*}(\overline{inv}(ms'_3),x'_0),x'_{23})) \in xy$ together with the pair $(\overline{*}(\overline{inv}(ms_2),x_1),\oplus(\overline{*}(\overline{inv}(ms'_2),x'_1),\ominus(x'_{23}))) \in xy$, we obtain the pair $(\oplus(\overline{*}(\overline{inv}(ms_3),x_0),\overline{*}(\overline{inv}(ms_2),x_1)),\oplus(\overline{*}(\overline{inv}(ms'_3),x'_0),\overline{*}(\overline{inv}(ms'_2),x'_1))) \in xy$. After that, we deduce the pair $(\oplus(\overline{*}(ms_2,x_0),\overline{*}(ms_3,x_1)),\oplus(\overline{*}(ms'_2,x'_0),\overline{*}(ms'_3,x'_1))) \in xy$ with the help of the condition Ψ^2_* and/or $\Psi^2_{*,\oplus}$. Based on $syn^{\overline{*}}(\oplus(m'_0,m'),ms'_0,\oplus(x'_0,x'_2))$, $ms'_0 = ms'_{01} \uplus ms'_2$, $syn^{\overline{*}}(\oplus(m'_1,\ominus(m')),ms'_1,\oplus(x'_1,\ominus(x'_3)))$ and $ms'_1 = ms'_{01} \uplus ms'_3$, we obtain $syn^{\overline{*}}(\oplus(m'_0,m'_1),ms'_{01},\oplus(\overline{*}(ms'_2,x'_0),\overline{*}(ms'_3,x'_1)))$. Furthermore, the equalities $m_0 = \overline{*}(ms_0,x_0)$ and $m_1 = \overline{*}(ms_1,x_1)$, together with the mappings in $\wp(ms_0,ms'_0)$ and $\wp(ms_1,ms'_1)$, permit to deduce $\oplus(m_0,m_1) = \overline{*}(ms_{01},\oplus(\overline{*}(ms_2,x_0),\overline{*}(ms_3,x_1)))$. Using these structures of $\oplus(m_0,m_1)$ and $\oplus(m'_0,m'_1)$ together with the last derived pair in xy, it is easy to show $\Theta^*_2(\oplus(m_0,m_1),m'_0,xy,kb,kb')$, by definition. □

13.6.4 Mapping by $\overset{xy}{\hookleftarrow}$ using the $\oplus$-Parts in $DY(kb')$

In this section, we describe the lemma employed in the $\Xi_\oplus$-case for the proof of the mapping by $\overset{xy}{\hookleftarrow}$, whose application is prepared by the derivation of appropriate $\oplus$-parts in $DY(kb')$. This lemma is applied in the last proof step of the $\Xi_\oplus$-case to deduce the mapping $\overline{\oplus}(ms_2 \uplus ms_3) \overset{xy}{\hookleftarrow} \overline{\oplus}(ms'_2 \uplus ms'_3)$ after having obtained the mappings in $\wp(ms_2,ms'_2) \subset \overset{xy}{\leftrightarrow}$ and $\wp(ms_3,ms'_3) \subset \overset{xy}{\hookleftarrow}$ with the required properties.

Lemma[VSE] 136 (Mapping of $\oplus$-Objects by $\overset{xy}{\hookleftarrow}$):
Let $ms = ms_2 \uplus ms_3 \subset DY(kb)$ and $ms' = ms'_2 \uplus ms'_3 \in DY(kb')$ satisfy $len(ms) > 1$, $syn^{\overline{\oplus}}(\overline{\oplus}(ms'),ms')$, $\wp(ms_2,ms'_2) \subset \overset{xy}{\leftrightarrow}$ and $\wp(ms_3,ms'_3) \subset \overset{xy}{\hookleftarrow}$. Additionally, let all mappings $m \overset{xy}{\leftrightarrow} m'$ in $\wp(ms_2,ms'_2)$ fulfill $\Theta^\oplus_1(m,kb)$ or $\Theta^*_2(m,m',xy,kb,kb')$ and all mappings $m \overset{xy}{\hookleftarrow} m'$ in $\wp(ms_3,ms'_3)$ fulfill $\Theta^*_2(m,m',xy,kb,kb')$. Then, we have

$$\Theta^\oplus_2(ms',kb') \Rightarrow \overline{\oplus}(ms) \overset{xy}{\hookleftarrow} \overline{\oplus}(ms').$$

Proof: The proof of this lemma is by induction on $|\overline{\oplus}(ms')|$.

In the base case, we have $ms' = \{c_i,c_j\}$ for atomic messages c_i and c_j different from ∞. Then, $\wp(ms_2,ms'_2)$ and $\wp(ms_3,ms'_3)$ provide $m_0 \overset{xy}{\leftrightarrow} c_i$ or $m_0 \overset{xy}{\hookleftarrow} c_i$ and $m_1 \overset{xy}{\leftrightarrow} c_i$ or $m_1 \overset{xy}{\hookleftarrow} c_i$ and we prove $\overline{\oplus}(ms) \overset{xy}{\hookleftarrow} \overline{\oplus}(ms')$ by the following case distinction:

- If $\overline{\oplus}(ms') \in codom(xy)$, i.e. $(m_x,\oplus(c_i,c_j)) \in xy$, holds, we use $\Psi^2_\oplus$ to instantiate m_x with $\oplus(m_0,m_1)$, i.e. $\overline{\oplus}(ms)$, and this immediately implies $\overline{\oplus}(ms) \overset{xy}{\hookleftarrow} \overline{\oplus}(ms')$.
- Otherwise, the proof of $\overline{\oplus}(ms) \overset{xy}{\hookleftarrow} \overline{\oplus}(ms')$ is trivial: Since $\overline{\oplus}(ms)$ can be decomposed only to the $\oplus$-parts c_i and c_j (modulo commutativity of $\oplus$), which are mapped to m_0 and m_1, the definition of $\widetilde{rec}$ provides $\oplus(m_0,m_1) \overset{xy}{\hookleftarrow} \oplus(c_i,c_j)$.

In the step case, the proof is analog by a similar case distinction. In particular, case $\overline{\oplus}(ms') \in codom(xy)$ is handled as in the base case with the help of $\Psi^2_\oplus$, which is applied $(len(ms)-1)$-times.

In case $\overline{\oplus}(ms') \notin codom(xy)$, we prove $\overline{\oplus}(ms) \overset{xy}{\hookleftarrow} \overline{\oplus}(ms')$ by providing *the* mapped messages for the available $\oplus$-parts, $*$-sub-messages and $\ominus$-sub-message of $\overline{\oplus}(ms')$ and then showing that their composition with $\oplus$, $*$ and respectively $\ominus$ yields $\overline{\oplus}(ms)$.

- Let $obj^\oplus(\overline{\oplus}(ms'),m'_0,m'_1)$ and $m'_0,m'_1 \in DY(kb')$ hold for two arbitrary $\oplus$-parts of $\overline{\oplus}(ms')$. Based on $syn^{\overline{\oplus}}(\overline{\oplus}(ms'),ms')$ and $\Theta^\oplus_2(ms',kb')$, we get a proper partition

$ms' = ms'_0 \uplus ms'_1$ satisfying $m'_0 = \overline{\oplus}(ms'_0)$ and $m'_1 = \overline{\oplus}(ms'_1)$. Furthermore, $\wp(ms_2, ms'_2)$ and $\wp(ms_3, ms'_3)$ are also partitioned into $\wp(ms_{02}, ms'_{02})$, $\wp(ms_{03}, ms'_{03})$, $\wp(ms_{12}, ms'_{12})$ and $\wp(ms_{13}, ms'_{13})$ for $ms_2 = ms_{02} \uplus ms_{12}$ and $ms_3 = ms_{03} \uplus ms_{13}$. We emphasize that $ms_0 = ms_{02} \uplus ms_{03}$, $ms'_0 = ms'_{02} \uplus ms'_{03}$, $ms_1 = ms_{12} \uplus ms_{13}$ and $ms'_1 = ms'_{12} \uplus ms'_{13}$ are not empty.

If $len(ms'_0) = 1$ or $len(ms'_1) = 1$, we deduce $\overline{\oplus}(ms_0) \overset{xy}{\leftrightarrow} m'_0$ and respectively $\overline{\oplus}(ms_1) \overset{xy}{\leftrightarrow} m'_1$ using the corresponding sole mapping (in $\wp(ms_{02}, ms'_{02})$ or $\wp(ms_{03}, ms'_{03})$ and respectively $\wp(ms_{12}, ms'_{12})$ or $\wp(ms_{13}, ms'_{13})$).

When $len(ms'_0) > 1$ or $len(ms'_1) > 1$, we deduce $\overline{\oplus}(ms_0) \overset{xy}{\leftarrow} m'_0$ and respectively $\overline{\oplus}(ms_1) \overset{xy}{\leftarrow} m'_1$ by the induction hypothesis.

In the four possible combinations, we show the required equality by

$$\oplus(\overline{\oplus}(ms_0), \overline{\oplus}(ms_1)) = \overline{\oplus}(ms_0 \uplus ms_1) = \overline{\oplus}(ms).$$

- Let $syn^*(\overline{\oplus}(ms'), m'_0, m'_1)$ and $m'_0, m'_1 \in DY(kb')$ hold for two arbitrary $*$-sub-messages of $\overline{\oplus}(ms')$. First, this means that for all $m' \in ms'$ we have some $m'_* \in DY(kb')$ that satisfies $syn^*(m', m'_0, m'_*)$. Then, we use that for every mapping $m \overset{xy}{\leftrightarrow} m'$ in $\wp(ms_2, ms'_2)$ to obtain in both cases, i.e. $\Theta_1^{\oplus}(m, kb)$ and $\Theta_2^*(m, m', xy, kb, kb')$, the condition $\Theta_2^*(*(inv(m_0), m), m'_*, xy, kb, kb')$ with the (same) mapping $m_0 \overset{xy}{\leftrightarrow} m'_0$. We also use $syn^*(m', m'_0, m'_*)$ for every mapping $m \overset{xy}{\leftarrow} m'$ in $\wp(ms_3, ms'_3)$ together with $\Theta_2^*(m, m', xy, kb, kb')$ to deduce the condition $\Theta_2^*(*(inv(m_0), m), m'_*, xy, kb, kb')$ with the (same) mapping $m_0 \overset{xy}{\leftrightarrow} m'_0$. This permits to define $ms'_{*2} = \{m'_* \mid m' \in ms'_2\}$, $ms'_{*3} = \{m'_* \mid m' \in ms'_3\}$, $ms'_* = ms'_{*2} \uplus ms'_{*3}$, $ms_{*2} = \{*(inv(m_0), m) \mid m \in ms_2\}$, $ms_{*3} = \{*(inv(m_0), m) \mid m \in ms_3\}$ and $ms_* = ms_{*2} \uplus ms_{*3}$ satisfying $m'_1 = \overline{\oplus}(ms'_*)$ and $\overline{\oplus}(ms_*) = *(inv(m_0), \overline{\oplus}(ms))$. Thus, we have all we need to apply the induction hypothesis and get $\overline{\oplus}(ms_*) \overset{xy}{\leftarrow} \overline{\oplus}(ms'_*)$, i.e. $\overline{\oplus}(ms_*) \overset{xy}{\leftarrow} m'_1$. This permits to show the required equality by $*(m_0, \overline{\oplus}(ms_*)) = *(m_0, *(inv(m_0), \overline{\oplus}(ms))) = \overline{\oplus}(ms)$.

- Let $syn^{\ominus}(\overline{\oplus}(ms'), m'_0)$ hold for the $\ominus$-sub-message of $\overline{\oplus}(ms')$. First, this means that for all $m' \in ms'$ we have some $m'_{\ominus} \in DY(kb')$ that satisfies $syn^{\ominus}(m', m'_{\ominus})$. Then, we use that for every mapping $m \overset{xy}{\leftrightarrow} m'$ in $\wp(ms_2, ms'_2)$ to obtain in both cases, i.e. $\Theta_1^{\oplus}(m, kb)$ and $\Theta_2^*(m, m', xy, kb, kb')$, the condition $\Theta_2^*(\ominus(m), m'_{\ominus}, xy, kb, kb')$. We also use $syn^{\ominus}(m', m'_{\ominus})$ for every mapping $m \overset{xy}{\leftarrow} m'$ in $\wp(ms_3, ms'_3)$ together with $\Theta_2^*(m, m', xy, kb, kb')$ to deduce the condition $\Theta_2^*(\ominus(m), m'_{\ominus}, xy, kb, kb')$. This permits to define $ms'_{\ominus 2} = \{m'_{\ominus} \mid m' \in ms'_2\}$, $ms'_{\ominus 3} = \{m'_{\ominus} \mid m' \in ms'_3\}$, $ms'_{\ominus} = ms'_{\ominus 2} \uplus ms'_{\ominus 3}$, $ms_{\ominus 2} = \{\ominus(m) \mid m \in ms_2\}$, $ms_{\ominus 3} = \{\ominus(m) \mid m \in ms_3\}$ and $ms_{\ominus} = ms_{\ominus 2} \uplus ms_{\ominus 3}$ satisfying $m'_0 = \overline{\oplus}(ms'_{\ominus})$ and $\overline{\oplus}(ms_{\ominus}) = \ominus(\overline{\oplus}(ms))$. Thus, we have all we need to apply the induction hypothesis and get $\overline{\oplus}(ms_{\ominus}) \overset{xy}{\leftarrow} \overline{\oplus}(ms'_{\ominus})$, i.e. $\overline{\oplus}(ms_{\ominus}) \overset{xy}{\leftarrow} m'_0$. This permits to show the required equality by $\ominus(\overline{\oplus}(ms_{\ominus})) = \ominus(\ominus(\overline{\oplus}(ms))) = \overline{\oplus}(ms)$. □

13.7 Proof of the Central Indistinguishability Theorem

The central indistinguishability theorem 108 in TC-AMP is shown the same way as in Sec. 11.4. In particular, the base case is trivial (see Sec. 11.4.1). The step case is handled according to the same proof plan as described in Sec. 11.4.2. In the following, we describe the concrete cases in the TC-AMP algebra.

13.7.1 Handling of the Canonical Cases:

Cases $f = fst$, $f = snd$ and $f = pair$ are handled as described in Sec. 11.4.2.2. That is, it remains to replay the same proof plan for cases $f = inv$, $f = h_1$ and $f = h_2$.

13.7.1.1 obj$^{\mathbf{f}}$-Case:

According to the proof plan in Sec. 11.4.2.2.1, the necessary conditions Ψ^1_{inv}, $\Psi^1_{h_1}$ and $\Psi^1_{h_2}$ permit to achieve the proof task in cases $f = inv$, $f = h_1$ and respectively $f = h_2$ based on lemma 122.

13.7.1.2 Complementary Case:

According to the proof plan in Sec. 11.4.2.2.2, we just need to consider case $f = inv$.

Proof Details: In case $f = inv$, the non-constructor-type application of inv in $m_l = inv(m_0)$ is selector-type. It implies $obj^{inv}(m_0, m_l)$ and the structural condition for $m_0 \overset{xy}{\leftrightarrow} m'_0$ is expanded in case $(m_0, m'_0) \in xy$ with the help of Ψ^1_{inv}. This permits to obtain an inv-structure for m'_0 and a mapped message to m_l that is a pendant of $inv(m'_0)$.

The complementary case, i.e. $(m_0, m'_0) \notin xy$, is handled similarly based on the canonical case for inv-objects in Ξ. □

13.7.2 Handling the Case for $f = \ominus$:

In the corresponding proof situation, we have $m_l = \ominus(m_0)$, $m_0 \overset{xy}{\leftrightarrow} m'_0$ and $m'_l = \ominus(m'_0)$, for $m_l, m_0 \in DY(kb)$ and $m'_l, m'_0 \in DY(kb')$. The proof consists then in providing m' that satisfies $m_l \overset{xy}{\leftrightarrow} m'$ and $m' = m'_l$, i.e. $m' = \ominus(m'_0)$.

Proof Details: To employ lemma 132, we distinguish two cases:

1. In case $m_0 = \infty$ and $\ominus(m_0) = \infty$, we set $m' = \infty$, as $m_l \overset{xy}{\leftrightarrow} m'$ ensues trivially from $(\infty, \infty) \in xy$. We prove the required equality $\ominus(m'_0) = \infty$ using Ψ^1_∞:

   First, the structural condition for $m_0 \overset{xy}{\leftrightarrow} m'_0$, i.e. $\infty \overset{xy}{\leftrightarrow} m'_0$, reduces to $(\infty, m'_0) \in xy$, as $\Xi(\infty, m', xy, kb, kb')$ does not hold by definition. Then, the application of Ψ^1_∞ to $(\infty, m'_0) \in xy$ yields $m'_0 = \infty$ and thus $\ominus(m'_0) = \infty$.

2. In the complementary case, i.e. $m_0 \neq \infty$ and $\ominus(m_0) \neq \infty$, we know that for every m'_0 there exists m'_1 satisfying $m'_0 = \ominus(m'_1)$ and $m'_1 = \ominus(m'_0)$. This permits to rewrite $m_0 \overset{xy}{\leftrightarrow} m'_0$ to $m_0 \overset{xy}{\leftrightarrow} \ominus(m'_1)$ and then to deduce $\ominus(m_0) \overset{xy}{\leftrightarrow} m'_1$ by lemma 132. Hence, we have all we need to conclude by setting $m' = m'_1$. □

13.7.3 Handling the Case for $f = *$:

In the corresponding proof situation, we have $m_l = *(m_0, m_1)$, $m_0 \overset{xy}{\leftrightarrow} m'_0$, $m_1 \overset{xy}{\leftrightarrow} m'_1$ and $m'_l = *(m'_0, m'_1)$, for $m_l, m_0, m_1 \in DY(kb)$ and $m'_l, m'_0, m'_1 \in DY(kb')$. The proof consists then in providing m' that satisfies $m_l \overset{xy}{\leftrightarrow} m'$ and $m' = m'_l$, i.e. $m' = *(m'_0, m'_1)$. We proceed by a case distinction, according to the alternative structures of $*(m_0, m_1)$ given by the operations of $\ominus$, $*$ and $\oplus$:

13.7.3.1 syn*-Case:

In case $syn^*(*(m_0,m_1),m_0,m_1)$, lemma 122 provides $*(m_0,m_1) \overset{xy}{\leftrightarrow} m'$, i.e. $m_l \overset{xy}{\leftrightarrow} m'$, for some $m' \in DY(kb')$, where (m_l,m') occurs in xy or it satisfies the Ξ_*- or the $\Xi_\oplus$-case. For our proof, it remains to show $m' = *(m'_0,m'_1)$.

1. In case $(*(m_0,m_1),m') \in xy$, we use Ψ^1_* to get $m' = *(m'_0,m'_1)$, as required.

2. In case $\Xi_*(m_l,m',xy,kb,kb')$, we get $syn^{\overline{*}}(m_l,ms,x)$, $\Theta^*_1(x,kb)$, $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$, $x \overset{xy}{\leftrightarrow} x'$ and $m' = \overline{*}(ms',x')$. Using $syn^*(m_l,m_0,m_1)$, the mappings $m_0 \overset{xy}{\leftrightarrow} m'_0$ and $m_1 \overset{xy}{\leftrightarrow} m'_1$ together with the structural condition of the latter, we prove $*(m'_0,m'_1) = \overline{*}(ms',x')$, i.e. $*(m'_0,m'_1) = m'$, in a similar way as in Sec. 13.5.2.1.

3. In case $\Xi_\oplus(m_l,m',xy,kb,kb')$, we get $syn^{\overline{\oplus}}(m_l,ms)$, $\Theta^\oplus_2(ms,kb)$, $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$ and $m' = \overline{\oplus}(ms')$. Using $syn^*(m_l,m_0,m_1)$, the mappings $m_0 \overset{xy}{\leftrightarrow} m'_0$ and $m_1 \overset{xy}{\leftrightarrow} m'_1$ together with their structural conditions, we prove $*(m'_0,m'_1) = \overline{\oplus}(ms',x')$, i.e. $*(m'_0,m'_1) = m'$, in a similar way as in Sec. 13.6.2.2. □

13.7.3.2 Complementary Case:

When $syn^*(*(m_0,m_1),m_0,m_1)$ does not hold, we distinguish mainly the following three cases:

1. In case $m_1 = \infty$ and $*(m_0,m_1) = \infty$, we set $m' = \infty$ and we prove the required equality $*(m'_0,m'_1) = \infty$ by showing $m'_1 = \infty$: The structural condition for $m_1 \overset{xy}{\leftrightarrow} m'_1$ reduces to $(\infty,m'_1) \in xy$, because $\Xi(\infty,m'_1,xy,kb,kb')$ does not hold. This permits to apply Ψ^1_∞ and obtain $m'_1 = \infty$, as required.

2. In case $sel^*_*(*(m_0,m_1),m_0,m_1)$, $sel^*_\ominus(*(m_0,m_1),m_0,m_1)$ or $sel^*_\oplus(*(m_0,m_1),m_0,m_1)$, we have $syn^*(m_1,inv(m_0),m_2)$ with $m_2 = *(m_0,m_1)$. lemma 122 provides $m_2 \overset{xy}{\leftrightarrow} m'_2$, permitting to set $m' = m'_2$. It remains to show $m'_2 = *(m'_0,m'_1)$, based on the structural condition for $m_1 \overset{xy}{\leftrightarrow} m'_1$, which provides the following three cases:

   (a) When $(m_1,m'_1) \in xy$ holds, we first apply condition Ψ^1_* to obtain that the pairs $(inv(m_0),m_{y1}),(m_2,*(inv(m_{y1}),m'_1))$ are in xy. Then, we employ a pendant of lemma 127 to deduce $m_{y1} = inv(m'_0)$, $(m_2,*(m'_0,m'_1)) \in xy$ and thus $m'_2 = *(m'_0,m'_1)$, as required.

   (b) In case $\Xi_*(m_1,m'_1,xy,kb,kb')$, we get the structural properties $syn^{\overline{*}}(m_1,ms_1,x)$, $\Theta^*_1(x,kb)$, $\wp(ms_1,ms'_1) \subset \overset{xy}{\leftrightarrow}$, $x \overset{xy}{\leftrightarrow} x'$ and $m'_1 = \overline{*}(ms'_1,x')$. Based on property $syn^*(m_1,inv(m_0),m_2)$, we obtain $ms_1 = inv(m_0) \uplus ms_2$ and $syn^{\overline{*}}(m_2,ms_2,x)$. Furthermore, it is obvious to show that $m_0 \overset{xy}{\leftrightarrow} m'_0$ implies $inv(m_0) \overset{xy}{\leftrightarrow} inv(m'_0)$. That is, we have $ms'_1 = inv(m'_0) \uplus ms'_2$ with $\wp(ms_2,ms'_2) \subset \overset{xy}{\leftrightarrow}$.
   We want to show $m'_2 = \overline{*}(ms'_2,x')$, in order to prove the required equality by $*(m'_0,m'_1) = *(m'_0,\overline{*}(ms'_1,x')) = *(m'_0,\overline{*}(inv(m'_0) \uplus ms'_2,x')) = \overline{*}(ms'_2,x') = m'_2$.
   If $ms_2 = \emptyset$, we have $m_2 = x$, $ms'_2 = \emptyset$ and $m'_2 = x' = \overline{*}(ms'_2,x')$, as required.

   Otherwise, the structural condition of $m_2 \overset{xy}{\leftrightarrow} m'_2$ reduces to two cases:

   - When $(m_2,m'_2) \in xy$, i.e. $(\overline{*}(ms_2,x),m'_2) \in xy$, we use Ψ^1_* to get $m'_2 = \overline{*}(ms'_2,x')$, as required.
   - When $\Xi_*(m_2,m'_2,xy,kb,kb')$, we use in particular $syn^{\overline{*}}(m_2,ms_2,x)$ to show $m'_2 = \overline{*}(ms'_2,x')$, as required.

(c) In case $\Xi_{\oplus}(m_1, m_1', xy, kb, kb')$, we get $syn^{\overline{\oplus}}(m_1, ms_1)$, $len(ms_1) > 1$, $\Theta_2^{\oplus}(ms_1, kb)$, $\wp(ms_1, ms_1') \subset^{xy}_{\leftrightarrow}$ and $m_1' = \overline{\oplus}(ms_1')$. Based on $syn^*(m_1, inv(m_0), m_2)$, there is ms_2 such that all messages m in ms_1 satisfy $syn^*(m, inv(m_0), m_3)$ for $m_3 \in ms_2$ and such that $syn^{\overline{\oplus}}(m_2, ms_2)$ holds. Then, we show (by induction on the length of ms_2, as described in Sec. 13.6.2.2) that there is $\wp(ms_2, ms_2') \subset^{xy}_{\leftrightarrow}$ such that $\overline{\oplus}(ms_1') = *(inv(m_0'), \overline{\oplus}(ms_2'))$ and the mappings $m \overset{xy}{\leftrightarrow} m'$ in $\wp(ms_1, ms_1')$ with $syn^*(m, inv(m_0), m_3)$ transfer to mappings $m_3 \overset{xy}{\leftrightarrow} *(m_0', m')$ in $\wp(ms_2, ms_2')$. Next, $syn^{\overline{\oplus}}(m_2, ms_2)$, $len(ms_2) > 1$ and $\wp(ms_2, ms_2') \subset^{xy}_{\leftrightarrow}$ permit to obtain the mapping $m_2 \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_2')$ as described in Sec. 13.6.2.1, case (2). Hence, we prove the required equality by $m_2' = \overline{\oplus}(ms_2') = *(m_0', *(inv(m_0'), \overline{\oplus}(ms_2'))) = *(m_0', \overline{\oplus}(ms_1')) = *(m_0', m_1')$.

3. In case $trs^*_{\oplus}(*(m_0, m_1), m_0, m_1)$, we have $obj^{\oplus}(*(m_0, m_1), m_2, m_3)$, $syn^*(m_2, m_0, m_4)$, $syn^*(*(inv(m_0), m_3), inv(m_0), m_3)$ and $obj^{\oplus}(m_1, m_4, *(inv(m_0), m_3))$. lemma 122 provides $*(m_0, m_1) \overset{xy}{\leftrightarrow} m'$ and it remains to show $m' = *(m_0', m_1')$, based on the structural condition of this mapping, which provides the following three cases:

   (a) When $(*(m_0, m_1), m') \in xy$, i.e. $(\oplus(*(m_0, m_4), m_3), m') \in xy$, we employ condition $\Psi^1_{*,\oplus}$.
   The application of condition $\Psi^1_{*,\oplus}$ to $(\oplus(*(m_0, m_4), m_3), m') \in xy$ permits to obtain $(m_0, m_{y1}), (\oplus(m_4, *(inv(m_0), m_3)), *(inv(m_y 1), m')) \in xy$ and then to deduce $m_{y1} = m_0'$ and $m_1' = *(inv(m_0'), m')$. Using the latter equality, we show $*(m_0', m_1') = m'$, as required.

   (b) In the Ξ_*-case, the definition of predicate Ξ_* provides the structural properties $syn^{\overline{*}}(*(m_0, m_1), ms, \oplus(*(m_0, x_1), x_2))$, $\wp(ms, ms') \subset^{xy}_{\leftrightarrow}$, $\oplus(*(m_0, x_1), x_2) \overset{xy}{\leftrightarrow} x'$ and $m' = \overline{*}(ms', x')$. This permits to obtain $syn^{\overline{*}}(m_1, ms, \oplus(x_1, *(inv(m_0), x_2)))$.
   First, we focus on the mappings $\oplus(*(m_0, x_1), x_2) \overset{xy}{\leftrightarrow} x'$ and $m_0 \overset{xy}{\leftrightarrow} m_0'$ to deduce $\oplus(x_1, *(inv(m_0), x_2)) \overset{xy}{\leftrightarrow} *(inv(m_0'), x')$. Then, we use the structural condition for $m_1 \overset{xy}{\leftrightarrow} m_1'$ and $syn^{\overline{*}}(m_1, ms, \oplus(x_1, *(inv(m_0), x_2)))$ to show that m_1' equals $\overline{*}(ms', *(inv(m_0'), x'))$. Finally, we prove the required equality by
   $$*(m_0', m_1') = *(m_0', \overline{*}(ms', *(inv(m_0'), x'))) = \overline{*}(ms', x') = m'.$$

   (c) In the $\Xi_{\oplus}$-case, the definition of $\Xi_{\oplus}$ provides $syn^{\overline{\oplus}}(*(m_0, m_1), ms)$, $len(ms) > 1$, $\Theta_2^{\oplus}(ms, kb)$, $\wp(ms, ms') \subset^{xy}_{\leftrightarrow}$ and $m' = \overline{\oplus}(ms')$. First, we provide a multiset ms_1 such that all messages m in the multiset ms transfer to messages $*(inv(m_0), m)$ in ms_1 and such that $syn^{\overline{\oplus}}(m_1, ms_1)$ holds. Then, we show (by induction on the length of ms) that there is $\wp(ms_1, ms_1') \subset^{xy}_{\leftrightarrow}$ such that $\overline{\oplus}(ms_1') = *(inv(m_0'), \overline{\oplus}(ms'))$ and the mappings $m \overset{xy}{\leftrightarrow} m'$ in $\wp(ms, ms')$ transfer to corresponding mappings $*(inv(m_0), m) \overset{xy}{\leftrightarrow} *(inv(m_0'), m')$ in $\wp(ms_1, ms_1')$. After that, $syn^{\overline{\oplus}}(m_1, ms_1)$, $len(ms_1) > 1$ and $\wp(ms_1, ms_1') \subset^{xy}_{\leftrightarrow}$ permit to obtain the mapping $m_1 \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_1')$ as described in Sec. 13.6.2.1, case (2). Hence, we prove the required equality by $*(m_0', m_1') = *(m_0', \overline{\oplus}(ms_1')) = *(m_0', *(inv(m_0'), \overline{\oplus}(ms'))) = \overline{\oplus}(ms') = m'$. □

13.7.4 Handling the Case for $f = \oplus$:

In the corresponding proof situation, we have $m_l = \oplus(m_0, m_1)$, $m_0 \overset{xy}{\leftrightarrow} m_0'$, $m_1 \overset{xy}{\leftrightarrow} m_1'$ and $m_l' = \oplus(m_0', m_1')$, for $m_l, m_0, m_1 \in DY(kb)$ and $m_l', m_0', m_1' \in DY(kb')$. The proof consists then

in providing $m_l \overset{xy}{\leftrightarrow} m'$ and showing $m' = m'_l$, i.e. $m' = \oplus(m'_0, m'_1)$. We proceed by a case distinction, according to the alternative structures of $\oplus(m_0, m_1)$ given by the operations of $\oplus$:

13.7.4.1 obj$^{\oplus}$-Case:

In case $obj^{\oplus}(\oplus(m_0,m_1), m_0, m_1)$, lemma 122 provides $\oplus(m_0,m_1) \overset{xy}{\leftrightarrow} m'$, i.e. $m_l \overset{xy}{\leftrightarrow} m'$, for some $m' \in DY(kb')$, where (m_l, m') occurs in xy or it satisfies the $\Xi_{\oplus}$-case. For our proof, it remains to show $m' = \oplus(m'_0, m'_1)$.

1. In case $(\oplus(m_0,m_1), m') \in xy$, we use $\Psi^1_{\oplus}$ to get $m' = \oplus(m'_0, m'_1)$, as required.

2. In case $\Xi_{\oplus}(m_l, m', xy, kb, kb')$, we get $syn^{\overline{\oplus}}(m_l, ms)$, $\Theta^{\oplus}_2(ms, kb)$, $\wp(ms, ms') \subset \overset{xy}{\leftrightarrow}$ and $m' = \overline{\oplus}(ms')$. Using $obj^{\oplus}(m_l, m_0, m_1)$, the mappings $m_0 \overset{xy}{\leftrightarrow} m'_0$ and $m_1 \overset{xy}{\leftrightarrow} m'_1$ together with their structural conditions, we prove $\oplus(m'_0, m'_1) = \overline{\oplus}(ms')$, i.e. $\oplus(m'_0, m'_1) = m'$, in a similar way as in Sec. 13.6.2.1. □

13.7.4.2 Complementary Case:

When $obj^{\oplus}(\oplus(m_0,m_1), m_0, m_1)$ does not hold, we apply lemma 122 to obtain $\oplus(m_0,m_1) \overset{xy}{\leftrightarrow} m'$, i.e. $m_l \overset{xy}{\leftrightarrow} m'$, and prove $m' = \oplus(m'_0, m'_1)$ using the following lemma:

Lemma$^{\text{VSE}}$ 137 ($\neg obj^{\oplus}(.,.,.)$ and $\overset{xy}{\leftrightarrow}$):
Let $m_0, m_1 \in DY(kb)$ and $m'_0, m'_1 \in DY(kb')$ satisfy $\neg obj^{\oplus}(\oplus(m_0,m_1), m_0, m_1)$, $m_0 \overset{xy}{\leftrightarrow} m'_0$ and $m_1 \overset{xy}{\leftrightarrow} m'_1$. Then, we have

$$\oplus(m_0,m_1) \overset{xy}{\leftrightarrow} \oplus(m'_0, m'_1).$$

Proof: The proof is by induction on $|m_0| + |m_1|$.

Base Case: For $|m_0| + |m_1| = 0$ and $\neg obj^{\oplus}(\oplus(m_0,m_1), m_0, m_1)$, we distinguish two cases:

1. $m_0 = m_1 = \infty$: Here, we have $\oplus(m_0,m_1) = \infty$ and thus $\oplus(m_0,m_1) \overset{xy}{\leftrightarrow} \infty$. That is, it remains to prove $\oplus(m'_0, m'_1) = \infty$.

   Since $m_0 = m_1 = \infty$, we have $m_0 \overset{xy}{\leftrightarrow} \infty$ and $m_1 \overset{xy}{\leftrightarrow} \infty$, which permits to deduce $m'_0 = \infty$ and $m'_1 = \infty$. This allows us to show $\oplus(m'_0, m'_1) = \infty$, as required.

2. W.l.o.g., $m_0 = \infty$ and $m_1 = c_i$ for $c_i \in At$: Here, we have $\oplus(m_0,m_1) = c_i = m_1$ and thus $\oplus(m_0,m_1) \overset{xy}{\leftrightarrow} m'_1$. That is, it remains to prove $\oplus(m'_0, m'_1) = m'_1$.

   Since $m_0 = \infty$, we have $m_0 \overset{xy}{\leftrightarrow} \infty$, which permits to deduce $m'_0 = \infty$ and then $\oplus(m'_0, m'_1) = \oplus(\infty, m'_1) = m'_1$, as required.

Step Case: For $|m_0| + |m_1| > 0$ and $\neg obj^{\oplus}(\oplus(m_0,m_1), m_0, m_1)$, we distinguish mainly four cases:

1. W.l.o.g., $m_0 = \infty$ and $|m_1| > 0$: Here, we proceed as in (2) from the Base Case.

2. $|m_0|,|m_1| > 0$ and $m_1 = \ominus(m_0)$: Here, we have $\oplus(m_0,m_1) = \infty$ and consequently $\oplus(m_0,m_1) \overset{xy}{\leftrightarrow} \infty$. That is, it remains to prove $\oplus(m_0',m_1') = \infty$.

   Based on the proof in Sec. 13.7.2, we know that $m_0 \overset{xy}{\leftrightarrow} m_0'$ implies $\ominus(m_0) \overset{xy}{\leftrightarrow} \ominus(m_0')$. Furthermore, $m_1 = \ominus(m_0)$ permits to rewrite $m_1 \overset{xy}{\leftrightarrow} m_1'$ to $\ominus(m_0) \overset{xy}{\leftrightarrow} m_1'$. This allows us to deduce $m_1' = \ominus(m_0')$ and then $\oplus(m_0',m_1') = \oplus(m_0',\ominus(m_0')) = \infty$, as required.

3. W.l.o.g., $obj^{\oplus}(m_1,m_2,\ominus(m_0))$: Here, we have $\oplus(m_0,m_1) = m_2$ and the structural condition of $m_1 \overset{xy}{\leftrightarrow} m_1'$ reduces to two cases:

   (a) In case $(m_1,m_1') \in xy$, i.e. $(\oplus(m_2,\ominus(m_0)),m_1') \in xy$, we use condition $\Psi^1_{\oplus}$ to deduce $(m_2,m_{y1}),(\ominus(m_0),\oplus(\ominus(m_{y1}),m_1')) \in xy$. That is, we have $m_2 \overset{xy}{\leftrightarrow} m_{y1}$, i.e. $\oplus(m_0,m_1) \overset{xy}{\leftrightarrow} m_{y1}$, and it remains to prove $\oplus(m_0',m_1') = m_{y1}$.

   As explained in (2), $m_0 \overset{xy}{\leftrightarrow} m_0'$ implies $\ominus(m_0) \overset{xy}{\leftrightarrow} \ominus(m_0')$ and this allows us to deduce $\oplus(\ominus(m_{y1}),m_1') = \ominus(m_0')$, then $\ominus(m_{y1}) = \oplus(\ominus(m_0'),\ominus(m_1'))$ and finally $m_{y1} = \oplus(m_0',m_1')$, as required.

   (b) In the $\Xi_{\oplus}$-case, the definition of $\Xi_{\oplus}$ provides $syn^{\overline{\oplus}}(m_1,ms)$, $len(ms) > 1$, $\Theta^{\oplus}_2(ms,kb)$, $\wp(ms,ms') \subset \overset{xy}{\leftrightarrow}$ and $m_1' = \overline{\oplus}(ms')$. Based on $obj^{\oplus}(m_1,m_2,\ominus(m_0))$ and $m_0 \in DY(kb)$, we provide a proper partition $ms_2 \uplus ms_{\ominus} = ms$ such that $syn^{\overline{\oplus}}(m_2,ms_2)$ and $syn^{\overline{\oplus}}(\ominus(m_0),ms_{\ominus})$ hold. This yields equally to a proper partition $ms_2' \uplus ms_{\ominus}' = ms'$ such that $\wp(ms_2,ms_2'),\wp(ms_{\ominus},ms_{\ominus}') \subset \overset{xy}{\leftrightarrow}$ holds. After that, $syn^{\overline{\oplus}}(m_2,ms_2)$ and $\wp(ms_2,ms_2') \subset \overset{xy}{\leftrightarrow}$ permit to obtain the mapping $m_2 \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_2')$ as described in Sec. 13.6.2.1. That is, we have $\oplus(m_0,m_1) \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_2')$, and it remains to prove $\oplus(m_0',m_1') = \overline{\oplus}(ms_2')$.

   First, $syn^{\overline{\oplus}}(\ominus(m_0),ms_{\ominus})$ permits to obtain a non-empty multiset ms_0 satisfying $syn^{\overline{\oplus}}(m_0,ms_0)$ and $m \in ms_{\ominus} \Leftrightarrow \ominus(m) \in ms_0$. Then, we show (by induction on the length of $ms_{\ominus}$) that there is $\wp(ms_0,ms_0') \subset \overset{xy}{\leftrightarrow}$ such that $\overline{\oplus}(ms_0') = \ominus(\overline{\oplus}(ms_{\ominus}'))$ and the mappings $m \overset{xy}{\leftrightarrow} m'$ in $\wp(ms_{\ominus},ms_{\ominus}')$ transfer to corresponding mappings $\ominus(m) \overset{xy}{\leftrightarrow} \ominus(m')$ in $\wp(ms_0,ms_0')$. After that, $syn^{\overline{\oplus}}(m_0,ms_0)$ and $\wp(ms_0,ms_0') \subset \overset{xy}{\leftrightarrow}$ permit to obtain the mapping $m_0 \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_0')$ as described in Sec. 13.6.2.1. This allows us to deduce $m_0' = \overline{\oplus}(ms_0')$ and then to show the required equality by $\oplus(m_0',m_1') = \oplus(\overline{\oplus}(ms_0'),\overline{\oplus}(ms')) = \oplus(\ominus(\overline{\oplus}(ms_{\ominus}')),\overline{\oplus}(ms_2' \uplus ms_{\ominus}')) = \overline{\oplus}(ms_2')$.

4. W.l.o.g., $obj^{\oplus}(m_0,m_2,m)$, $obj^{\oplus}(m_1,m_3,\ominus(m))$ and $obj^{\oplus}(\oplus(m_2,m_3),m_2,m_3)$: Here, we have $\oplus(m_0,m_1) = \oplus(m_2,m_3)$ and we need to proceed by a case distinction (mainly six cases) induced by combining the structural conditions of $m_0 \overset{xy}{\leftrightarrow} m_0'$ and $m_1 \overset{xy}{\leftrightarrow} m_1'$:

   (a) In case $(m_0,m_0'),(m_1,m_1') \in xy$, i.e. $(\oplus(m_2,m),m_0'),(\oplus(m_1,\ominus(m)),m_1') \in xy$, we apply condition $\Psi^1_{\oplus,mrg}$ to obtain the pair $(\oplus(m_2,m_3),\oplus(m_0',m_1')) \in xy$ and thus $\oplus(m_0,m_1) \overset{xy}{\leftrightarrow} \oplus(m_0',m_1')$, as required.

   (b) In case $(m_0,m_0') \in xy$ and $\Xi_*(m_1,m_1',xy,kb,kb')$ hold, we want to proceed as in the proof of lemma 131 (cp. case (2) in the Base Case), to derive the mapping $\oplus(m_0,m_1) \overset{xy}{\leftrightarrow} \oplus(m_0',m_1')$ with the help of the following predicate (as a pendant of Θ^*_2):

$$\begin{aligned}&\Theta^*_3(m_c,m_c',xy,kb,kb') \Leftrightarrow\\ &(\exists ms_c \subset DY(kb), x_c \in DY(kb), ms_c' \subset DY(kb'), x_c' \in DY(kb') :\\ &\quad syn^{\overline{*}}(m_c,ms_c,x_c) \wedge \wp(ms_c,ms_c') \subset \overset{xy}{\leftrightarrow} \wedge (x_c,x_c') \in xy \wedge m_c' = \overline{*}(ms_c',x_c'))\end{aligned}$$

Based on the properties $(m_0, m_0') \in xy$, $isObj^{\oplus}(m_0)$ and $\Xi_*(m_1, m_1', xy, kb, kb')$, we trivially have $\Theta_3^*(m_0, m_0', xy, kb, kb')$ and $\Theta_3^*(m_1, m_1', xy, kb, kb')$. Then, we use in particular $obj^{\oplus}(m_0, m_2, m)$ and $obj^{\oplus}(m_1, m_3, \ominus(m))$ to obtain

$$\Theta_3^*(\oplus(m_2, m_3), \oplus(m_0', m_1'), xy, kb, kb')$$

with the help of a lemma that is a pendant of lemma 135. Afterwards, we derive $\oplus(m_2, m_3) \overset{xy}{\leftrightarrow} \oplus(m_0', m_1')$ and hence $\oplus(m_0, m_1) \overset{xy}{\leftrightarrow} \oplus(m_0', m_1')$ with the help of a lemma that can be shown according to the proof principle of lemma 128.

(c) In case $(m_0, m_0') \in xy$ and $\Xi_{\oplus}(m_1, m_1', xy, kb, kb')$ hold, we have $syn^{\overline{\oplus}}(m_1, ms)$, $len(ms) > 1$, $\wp(ms, ms') \subset \overset{xy}{\leftrightarrow}$ and $m_1' = \overline{\oplus}(ms')$. We consider a proper partition $ms_4 \uplus ms_5 = ms$ and a corresponding proper partition $ms_4' \uplus ms_5' = ms'$ such that $\wp(ms_4, ms_4'), \wp(ms_5, ms_5') \subset \overset{xy}{\leftrightarrow}$ holds. Then, we derive the mappings $\overline{\oplus}(ms_4) \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_4')$ and $\overline{\oplus}(ms_5) \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_5')$, as described in Sec. 13.6.2.1. Afterwards, $obj^{\oplus}(\oplus(m_3, \ominus(m)), \overline{\oplus}(ms_4), \overline{\oplus}(ms_5))$ provides the following cases:

- $\overline{\oplus}(ms_4) = \ominus(m)$ and $\overline{\oplus}(ms_5) = m_3$: Focusing on the mappings $m_0 \overset{xy}{\leftrightarrow} m_0'$, i.e. $\oplus(m_2, m) \overset{xy}{\leftrightarrow} m_0'$, and $\overline{\oplus}(ms_4) \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_4')$, i.e. $\ominus(m) \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_4')$, we have $\neg obj^{\oplus}(\oplus(m_0, \ominus(m)), m_0, \ominus(m))$ and this permits to apply the induction hypothesis and obtain $\oplus(m_0, \ominus(m)) \overset{xy}{\leftrightarrow} \oplus(m_0', \overline{\oplus}(ms_4'))$, i.e. $m_2 \overset{xy}{\leftrightarrow} \oplus(m_0', \overline{\oplus}(ms_4'))$. After that, we use $m_2 \overset{xy}{\leftrightarrow} \oplus(m_0', \overline{\oplus}(ms_4'))$ and $\overline{\oplus}(ms_5) \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_5')$, i.e. $m_3 \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_5')$, together with $obj^{\oplus}(\oplus(m_2, m_3), m_2, m_3)$, to prove $\oplus(m_2, m_3) \overset{xy}{\leftrightarrow} \oplus(\oplus(m_0', \overline{\oplus}(ms_4')), \overline{\oplus}(ms_5'))$, as described in Sec. 13.7.4.1. Since

  $$\oplus(\oplus(m_0', \overline{\oplus}(ms_4')), \overline{\oplus}(ms_5')) = \oplus(m_0', \overline{\oplus}(ms')) = \oplus(m_0', m_1'),$$

  the latter mapping rewrites to $\oplus(m_2, m_3) \overset{xy}{\leftrightarrow} \oplus(m_0', m_1')$, i.e. $\oplus(m_0, m_1) \overset{xy}{\leftrightarrow} \oplus(m_0', m_1')$, as required.
- $obj^{\oplus}(\ominus(m), \overline{\oplus}(ms_4), m_5)$ and $obj^{\oplus}(\overline{\oplus}(ms_5), m_3, m_5)$: Here, $\oplus(m_2, m)$ rewrites to $\oplus(m_2, \ominus(m_5), \ominus(\overline{\oplus}(ms_4)))$ and this permits to apply the induction hypothesis for $\oplus(m_2, m) \overset{xy}{\leftrightarrow} m_0'$ and $\overline{\oplus}(ms_5) \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_5')$ and deduce

  $$\oplus(m_2, \oplus(m_3, \ominus(\overline{\oplus}(ms_4)))) \overset{xy}{\leftrightarrow} \oplus(m_0', \overline{\oplus}(ms_5')).$$

  After that, we use the obtained mapping and $\overline{\oplus}(ms_4) \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_4')$ to apply the induction hypothesis and deduce the following mapping

  $$\oplus(\oplus(m_2, \oplus(m_3, \ominus(\overline{\oplus}(ms_4)))), \overline{\oplus}(ms_4)) \overset{xy}{\leftrightarrow} \oplus(\oplus(m_0', \overline{\oplus}(ms_5')), \overline{\oplus}(ms_4')),$$

  i.e. $\oplus(m_2, m_3) \overset{xy}{\leftrightarrow} \oplus(m_0', m_1')$, as required.
- $obj^{\oplus}(\overline{\oplus}(ms_4), \ominus(m), m_{3,4})$ and $obj^{\oplus}(m_3, \overline{\oplus}(ms_5), m_{3,4})$: Here, we first deduce $\oplus(m_2, m_{3,4}) \overset{xy}{\leftrightarrow} \oplus(m_0', \overline{\oplus}(ms_4'))$ with the help of the induction hypothesis. Then, we use the obtained mapping together with $\overline{\oplus}(ms_5) \overset{xy}{\leftrightarrow} \overline{\oplus}(ms_5')$ to prove $\oplus(\oplus(m_2, m_{3,4}), \overline{\oplus}(ms_5)) \overset{xy}{\leftrightarrow} \oplus(\oplus(m_0', \overline{\oplus}(ms_4')), \overline{\oplus}(ms_5'))$, as described in Sec. 13.7.4.1. The latter mapping rewrites to $\oplus(m_2, m_3) \overset{xy}{\leftrightarrow} \oplus(m_0', m_1')$, as required.
- $obj^{\oplus}(\overline{\oplus}(ms_4), m_4, m_{3,4})$, $obj^{\oplus}(m_3, m_{3,4}, m_{3,5})$ and $obj^{\oplus}(\overline{\oplus}(ms_5), m_5, m_{3,5})$: Here, we first deduce

  $$\oplus(\oplus(m_2, \ominus(m_5)), m_{3,4}) \overset{xy}{\leftrightarrow} \oplus(m_0', \overline{\oplus}(ms_4'))$$

with the help of the induction hypothesis. Afterwards, we use the obtained mapping together with $\overline{\oplus}(ms_5) \overset{xy}{\leftrightarrow} \overline{\oplus}(ms'_5)$ to get

$$\oplus(\oplus(\oplus(m_2, \ominus(m_5)), m_{3,4}), \oplus(m_5, m_{3,5})) \overset{xy}{\leftrightarrow} \oplus(\oplus(m'_0, \overline{\oplus}(ms'_4)), \overline{\oplus}(ms'_5))$$

with the help of the induction hypothesis. The latter mapping rewrites to $\oplus(m_2, m_3) \overset{xy}{\leftrightarrow} \oplus(m'_0, m'_1)$, as required.

(d) In case $\Xi_*(m_0, m'_0, xy, kb, kb')$ and $\Xi_*(m_1, m'_1, xy, kb, kb')$, the proof is performed mainly as in (b).

(e) In case $\Xi_*(m_0, m'_0, xy, kb, kb')$ and $\Xi_\oplus(m_1, m'_1, xy, kb, kb')$, the proof is performed mainly as in (c).

(f) In case $\Xi_\oplus(m_0, m'_0, xy, kb, kb')$ and $\Xi_\oplus(m_1, m'_1, xy, kb, kb')$, the proof is performed mainly as in (c). □

Chapter 14

Resistance Proof of TC-AMP

In Chap. 13, we described the proof of the central indistinguishability theorem in the TC-AMP algebra, identifying the necessary and sufficient conditions. In this chapter, we apply the proof technique introduced in Chap. 11 to show the resistance of TC-AMP against offline password testing: After the formalization of this property and the definition of the basis relations (in Sec. 14.1), we present the required regularity properties (in Sec. 14.2), then we describe the proof of the basis simulation relation lemma (in Sec. 14.3). Finally, we show (in Sec. 14.4) the proof obligations resulting from the conditions of the central indistinguishability theorem 108, which are identified in Chap. 13.

14.1 Formalization and Basis Relations

The resistance of TC-AMP against offline password testing is formalized by a simple adaptation of PACE's property 105. As described in Sec. 11.1, we need just to adapt the set of traces (TCAMP instead of PACE) and the regular messages with an occurrence of a protected password π. In case of TC-AMP, a protected password π occurs in a first or a second TC-AMP message. Thus, we obtain the following formalization of the resistance property:

Property[VSE] 138 (Resistance against Offline Password Guessing; TC-AMP):

$$
\begin{aligned}
&(tr \in \text{TCAMP} \wedge \pi \notin DY(spies(tr)) \wedge (\forall m \in spies(tr) : \neg uses(m, \pi')) \wedge \\
&\quad ((\exists g_1, g_2, nc_1 : \oplus(*(nc_1, g_1), \ominus(*(\pi, g_2))) \in spies(tr)) \vee \\
&\quad\ (\exists g_1, nc_2, m_{2,2} : pair(*(nc_2, *(\pi, g_1)), m_{2,2}) \in spies(tr)))) \\
&\Rightarrow \\
&\forall i, j : \overline{DY}(\pi \bullet \overline{spies(tr)}, i) = \overline{DY}(\pi \bullet \overline{spies(tr)}, j) \Leftrightarrow \\
&\qquad \overline{DY}(\pi' \bullet \overline{spies(tr)}, i) = \overline{DY}(\pi' \bullet \overline{spies(tr)}, j)
\end{aligned}
$$

According to the proof plan in Sec. 11.1, the first proof task consists in identifying the contents of the sets xy that are used as basis relations. These sets are defined relative to ik and to the parameters π, π'. As described in Sec. 11.1.1.2, we define the sets xy to be the smallest sets of message pairs that satisfy the (protocol- and property-specific) inclusion rules Φ_0–Φ_8, below. With these rules, we aim at basis relations xy that satisfy in particular Ψ^c and Ψ^a.

Φ_0 If $(m, m') \in xy$ and m or m' is a $\ominus$-, $*$- or a $\oplus$-object, then xy includes $(\ominus(m), \ominus(m'))$

Φ_1 If there is g_1, g_2 and nc_1 with $\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2))) \in ik$ or there is g_1, nc_2 and m_2 with $pair(*(nc_2,*(\pi,g_1)),m_2) \in ik$, then xy includes (π,π') and (∞,∞).

Φ_2 If there exists a password $pw(i)$ and two numerical atomic messages $num(j)$ and $num(k)$ with $pair(ag(i),pair(pw(i),pair(num(j),num(k)))) \in ik$ holds, then xy includes $(ag(i),ag(i))$.

If there is $ag(j)$ and $pw(j)$ with $pair(ag(j),pair(pw(j),(pair(num(i),num(k))))) \in ik$ or there is $pw(j)$ with $pair(pw(j),(pair(num(i),num(k)))) \in ik$, then xy includes $(num(i),num(i))$ and $(num(k),num(k))$.

If there is $ag(i)$, $num(j)$ and $num(k)$ with $pair(ag(i),pair(pw(i),pair(num(j),num(k)))) \in ik$ or there is $num(j)$ and $num(k)$ with $pair(pw(i),pair(num(j),num(k))) \in ik$, then xy includes $(pw(i),pw(i))$.

Φ_3 If $nc(i) \in ik$, then xy includes $(nc(i),nc(i))$.

Φ_4 `% regular msg1-1: --> (4.1)`
If $\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))) \in ik$ with $nc_1 \notin DY(ik)$ and $pw,g_2 \in DY(ik)$, then xy includes $(*(nc_1,g_1),*(nc_1,g_1))$.

`% regular msg1-2: --> (4.2)`
If $\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))) \in ik$ with $nc_1,pw \notin DY(ik)$, then xy includes $(\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))))$.

`% regular msg1-3: (4.2) & pi [*,plu]--> (pi,pi') & (4.3)`
If $\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2))) \in ik$, then xy includes $(\oplus(*(inv(\pi),*(nc_1,g_1)),\ominus(g_2)),\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(\pi,g_2)))))$ and $(inv(\pi),inv(\pi'))$.

`% regular msg1-4: (4.2) & mns(*(pi,g2)) [plu]--> (4.6) & (4.8)`
If $\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2))) \in ik$ with $g_2 \in DY(ik)$, then xy includes $(\ominus(g_2),\ominus(g_2))$, $(\ominus(*(\pi,g_2)),\ominus(*(\pi',g_2)))$, $(*(inv(\pi),*(nc_1,g_1)),\oplus(\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(\pi,g_2)))),g_2))$ and $(*(nc_1,g_1),\oplus(\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2))),*(\pi',g_2)))$.

Φ_5 `% regular msg2-1:`
If $pair(*(pw,*(nc_2,g_1)),h_1(m_1,*(pw,*(nc_2,g_1)),m_3)) \in ik$ with $nc_2 \notin DY(ik)$ and $pw \in DY(ik)$, then xy includes $(*(nc_2,g_1),*(nc_2,g_1))$.

`% regular msg2-2:`
If $pair(*(pw,*(nc_2,g_1)),h_1(m_1,*(pw,*(nc_2,g_1)),m_3)) \in ik$ such that $nc_2,pw \notin DY(ik)$ holds, then xy includes $(*(pw,*(nc_2,g_1)),*(pw,*(nc_2,g_1)))$.

`% regular msg2-3:`
If $pair(*(pw,*(nc_2,g_1)),h_1(m_1,*(pw,*(nc_2,g_1)),m_3)) \in ik$ with $m_3 \notin DY(ik)$, then xy includes $(h_1(m_1,*(pw,*(nc_2,g_1)),m_3),h_1(m_1,*(pw,*(nc_2,g_1)),m_3))$.

`% regular msg2-4:`
If we have a message $pair(*(\pi,*(nc_2,g_1)),h_1(m_1,*(\pi,*(nc_2,g_1)),m_3)) \in ik$, then xy includes $(*(nc_2,g_1),*(inv(\pi'),*(\pi,*(nc_2,g_1))))$ and $(inv(\pi),inv(\pi'))$.

Φ_6 `% regular msg3:`
If we have a third TC-AMP message $h_2(m_1,m_2,m_3) \in ik$ such that $m_3 \notin DY(ik)$ holds, then xy includes $(h_2(m_1,m_2,m_3),h_2(m_1,m_2,m_3))$.

Φ_7 `% regular oops-1: --> (7.1)`
If we have the oops message $\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[g_2,pw],g_1))) \in ik$ such that $m[g_2,pw] = \oplus(*(nc_1,g_1),\ominus(*(pw,g_2)))$, $nc_1,nc_2 \notin DY(ik)$ and $pw \in DY(ik)$ hold, then xy includes $(*(nc_2,*(nc_1,g_1)),*(nc_2,*(nc_1,g_1)))$.

`% regular oops-2: --> (7.2),(7.3),(7.4)`
If we have the oops message $\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[g_2,pw],g_1))) \in ik$ such that $m[g_2,pw] = \oplus(*(nc_1,g_1),\ominus(*(pw,g_2)))$ and $nc_1,nc_2,pw \notin DY(ik)$ hold, then xy includes
$(inv(m[g_2,pw]),inv(m[g_2,pw]))$,
$(\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[g_2,pw],g_1)))$,
$\quad \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[g_2,pw],g_1))))$ and
$(\oplus(*(inv(m[g_2,pw]),*(nc_2,*(nc_1,g_1))),*(nc_2,g_1))$,
$\quad \oplus(*(inv(m[g_2,pw]),*(nc_2,*(nc_1,g_1))),*(nc_2,g_1)))$.

`% regular oops-3:`
`% (7.3) & plu-part2 [plu]--> (7.5) & (7.6)`
`% (7.6) & inv(pi') [*]--> (inv(pi),inv(pi')) & (7.7)`
`% (7.5) & m[pi] [*,plu]--> (m[pi],m[pi]) & (7.8)`
`% (7.5) & inv(pi') [*,plu]--> (inv(pi),inv(pi')) & (7.9)`
`% (7.8) & inv(pi') [*,plu]--> (inv(pi),inv(pi')) & (7.10)`
`% (7.9) & plu-part [plu]--> (7.7) & (7.11)`
`% (7.10) & plu-part [plu]--> (reg-msg2) & (7.12)`
If we have the oops message $\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[g_2,\pi],g_1))) \in ik$ with $m[g_2,\pi] = \oplus(*(nc_1,g_1),\ominus(*(\pi,g_2)))$, then xy includes
$(*(nc_2,*(nc_1,g_1)),\oplus(\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[g_2,\pi],g_1))),$
$\quad \ominus(*(m[g_2,\pi],*(inv(\pi'),*(\pi,*(nc_2,g_1))))))),$
$(*(m[g_2,\pi],*(nc_2,g_1)),*(m[g_2,\pi],*(inv(\pi'),*(\pi,*(nc_2,g_1))))),$
`% % % %`
$(*(m[g_2,\pi],*(\pi,*(nc_2,g_1))),*(m[g_2,\pi],*(\pi,*(nc_2,g_1)))),$
`% % % %`
$(*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1))),$
$\quad \oplus(\oplus(*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1))),*(nc_2,g_1)),$
$\quad\quad \ominus(*(inv(\pi'),*(\pi,*(nc_2,g_1))))),$
`% % % %`
$(*(\pi,*(nc_2,*(nc_1,g_1))),\oplus(\oplus(*(\pi',*(nc_2,*(nc_1,g_1))),*(\pi',*(nc_2,*(m[g_2,\pi],g_1)))),$
$\quad \ominus(*(m[g_2,\pi],*(\pi,*(nc_2,g_1))))),$
`% % % %`
$(*(\pi,*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),$
$\quad \oplus(\oplus(*(\pi',*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),*(\pi',*(nc_2,g_1))),$
$\quad\quad \ominus(*(\pi,*(nc_2,g_1))))),$
`% % % %`
$(\oplus(*(\pi,*(nc_2,*(nc_1,g_1))),*(m[g_2,\pi],*(\pi,*(nc_2,g_1)))),$
$\quad \oplus(*(\pi',*(nc_2,*(nc_1,g_1))),*(\pi',*(nc_2,*(m[g_2,\pi],g_1))))))$ and
`% % % %`
$(\oplus(*(\pi,*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),*(\pi,*(nc_2,g_1)))$
$\quad \oplus(*(\pi',*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),*(\pi',*(nc_2,g_1))))$.

Φ_8 `% merge-1: --> (8.1)`
If we have $\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),\oplus(*(nc_3,g_1),\ominus(*(pw,g_2))) \in ik$ such that $nc_1 \neq nc_3$, $nc_1,nc_3,pw \notin DY(ik)$ hold, then xy includes
$(\oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))),\oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))))$.

`% merge-2: --> (8.2)`
If $\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2))),\oplus(*(nc_3,g_1),\ominus(*(\pi,g_2))) \in ik$ with $nc_1,nc_3 \notin DY(ik)$ and $nc_1 \neq nc_3$, then xy includes
$(\oplus(*(inv(\pi),*(nc_1,g_1)),\ominus(*(inv(\pi),*(nc_3,g_1)))),$
$\quad \oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(nc_3,g_1)))))$.

The inclusion rule Φ_0 is used to fulfill $\Psi^1_{\ominus}$, $\Psi^2_{\ominus}$, $\Psi^1_{\ominus,\oplus}$, $\Psi^2_{\ominus,\oplus}$, $\Psi^1_{\ominus,*}$ and $\Psi^2_{\ominus,*}$.

According to Φ_1, the presence of a first or a second TC-AMP message with an occurrence of π necessitates to include the pair (π, π') together with the pair (∞, ∞), which permits to fulfill Ψ^1_∞ and Ψ^2_∞.

Φ_2 states when pairs (c_i, c_i) are included for public atomic messages c_i. Φ_3 is similar but focuses on the public nonces.

Φ_4 covers the pairs that can be included when a first TC-AMP message is generated by a non-compromised agent (excluding derivations by merging). The following cases are distinguished:

- `msg1-1`: The sender has used a compromised password pw, which is accessible together with g_2 and g_1.
- `msg1-2`: The sender has used a non-compromised password pw, which could equal π.
- `msg1-3`: `msg1-2` yields to a pair that includes a $\oplus$-object with π as a public left $*$-part.
- `msg1-4`: `msg1-2` yields to a pair that includes a $\oplus$-object with $\ominus(*(\pi, g_2))$ as a public $\oplus$-part, if g_2 is public.

Φ_5 covers the pairs that can be included when a second TC-AMP message is generated by a non-compromised agent. The following cases are distinguished:

- `msg2-1`: The sender has used a compromised password pw.
- `msg2-2`: The sender has used a non-compromised password pw, which could equal π.
- `msg2-3`: The third h_1-part in the second message part is not public.
- `msg2-4`: `msg2-2` yields to a pair that includes a $*$-object with π as a public left $*$-part.

According to Φ_6, corresponding pairs are included when the third h_2-part in the third TC-AMP message is not public.

Φ_7 covers the pairs that can be included in the oops case of TC-AMP where the nonces are protected. The following cases are distinguished:

- `oops-1`: The password pw is public.
- `oops-2`: The password pw is not compromised and could equal π.
- `oops-3`: Eight further pairs are included because of the following reasons:
  - `(7-5), (7-6)`: `oops-2` yields to a pair that includes some $\oplus$-object possessing the $*$-object $*(m[g_2, \pi], *(nc_2, g_1))$ as a public $\oplus$-part.
  - `(7-7)`: The pair `(7-6)` includes a $*$-object with $inv(\pi')$ as a public left $*$-part.
  - `(7-8)`: The pair `(7-5)` includes a $\oplus$-object with $m[g_2, \pi]$ as a public left $*$-part.
  - `(7-9)`: The pair `(7-5)` includes a $\oplus$-object with $inv(\pi')$ as a public left $*$-part.
  - `(7-10)`: The pair `(7-8)` includes a $\oplus$-object with $inv(\pi')$ as a public left $*$-part.
  - `(7-11)`: The pair `(7-9)` includes a $\oplus$-object with $*(m[g_2, \pi], *(\pi, *(nc_2, g_1)))$ as a public $\oplus$-part.
  - `(7-12)`: The pair `(7-10)` includes a $\oplus$-object with $\ominus(*(\pi, *(nc_2, g_1)))$ as a public $\oplus$-part.

Φ_8 covers the pairs that can be included because of merging derivations. The following cases are distinguished:

- `merge-1`: First messages generated using a non-compromised password pw are subject to merging because of the $\oplus$-part $*(pw, g_2)$.
- `mserge-2`: First messages generated using π yield to pairs with $\oplus$-objects extended with $inv(\pi)$ and respectively $inv(\pi')$ that are subject to merging because of the $\oplus$-part g_2 (and other common $\oplus$-parts).

In the following, we use $\Phi(xy, ik, \pi, \pi')$ to abbreviate the definition of xy being the smallest set that satisfies the inclusion rules Φ_0–Φ_8.

14.2 Employed Regularity Properties

We formalize the required regularity properties and explain how they are shown by trace induction.

14.2.1 Derivable Atomic Messages

In this section, we provide the employed regularity properties about the derivable atomic messages.

Property$^{\text{VSE}}$ 139 (Derivable Atomic Messages):

1. $(tr \in \text{TCAMP} \wedge ag(i) \in DY(pw(k).spies(tr))) \Rightarrow$
   $(\exists j, j' : pair(ag(i), pair(pw(i), pair(num(j), num(j')))) \in spies(tr))$

2. $(tr \in \text{TCAMP} \wedge num(i) \in DY(pw(k).spies(tr))) \Rightarrow$
   $(\exists j, j' : pair(ag(j), pair(pw(j), pair(num(i), num(j')))) \in spies(tr) \vee$
   $pair(ag(j), pair(pw(j), pair(num(j'), num(i)))) \in spies(tr) \vee$
   $pair(pw(j), pair(num(i), num(j'))) \in spies(tr) \vee$
   $pair(pw(j), pair(num(j'), num(i))) \in spies(tr))$

3. $(tr \in \text{TCAMP} \wedge pw(i) \in DY(pw(k).spies(tr))) \Rightarrow$
   $(pw(k) = pw(i) \vee$
   $(\exists j, j' : (pair(ag(i), pair(pw(i), pair(num(j), num(j')))) \in spies(tr) \vee$
   $pair(pw(i), pair(num(j), num(j'))) \in spies(tr))$

4. $(tr \in \text{TCAMP} \wedge nc(i) \in DY(pw(k).spies(tr))) \Rightarrow nc(i) \in spies(tr)$

Properties 1–4 are shown similar to properties 1–3 in Sec. 12.2.1.

14.2.2 Derivable $*$-Messages

In this section, we describe the employed regularity property about the derivable syn^*-messages that are not $\oplus$-objects (simple syn^*-messages). Simple syn^*-messages having only confidential $*$-sub-messages are linked to regular TC-AMP messages, where the different cases are distinguished with the help of the following predicates:

1. TCAMP_3$msg_{1,1}$ and TCAMP_$msg_{1,1}$ link a simple syn^*-message to a first TC-AMP message:

$$\begin{aligned}
&\text{TCAMP_3}msg_{1,1}(m,ik,pw(i)) \Leftrightarrow\\
&(\exists nc_1,g_1,g_2,pw(j) : \oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))) \in ik \wedge nc_1 \notin DY(pw(i)\bullet ik) \wedge\\
&\quad pw(j),g_2 \in DY(pw(i)\bullet ik) \wedge (m = *(nc_1,g_1) \vee m = \ominus(*(nc_1,g_1))))
\end{aligned}$$

$$\begin{aligned}
&\text{TCAMP_}msg_{1,1}(m,ik) \Leftrightarrow\\
&(\exists nc_1,g_1,g_2,pw(j) : \oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))) \in ik \wedge nc_1 \notin DY(ik) \wedge\\
&\quad pw(j),g_2 \in DY(ik) \wedge (m = *(nc_1,g_1) \vee m = \ominus(*(nc_1,g_1)))).
\end{aligned}$$

2. TCAMP_3$msg_{2,1}$ and TCAMP_$msg_{2,1}$ link a simple syn^*-message to a second TC-AMP message:

$$\begin{aligned}
&\text{TCAMP_3}msg_{2,1}(m,ik,pw(i)) \Leftrightarrow\\
&(\exists nc_2,g_1,pw(j),m_1 : pair(*(nc_2,*(pw(j),g_1)),m_1) \in ik \wedge nc_2 \notin DY(pw(i)\bullet ik) \wedge\\
&\quad ((pw(j) \in DY(pw(i)\bullet ik) \wedge (m = *(nc_2,g_1) \vee m = \ominus(*(nc_2,g_1)))) \vee\\
&\quad (pw(j) \notin DY(pw(i)\bullet ik) \wedge (m = *(nc_2,*(pw(j),g_1)) \vee m = \ominus(*(nc_2,*(pw(j),g_1)))))))
\end{aligned}$$

$$\begin{aligned}
&\text{TCAMP_}msg_{2,1}(m,ik) \Leftrightarrow\\
&(\exists nc_2,g_1,pw(j),m_1 : pair(*(nc_2,*(pw(j),g_1)),m_1) \in ik \wedge nc_2 \notin DY(ik) \wedge\\
&\quad ((pw(j) \in DY(ik) \wedge (m = *(nc_2,g_1) \vee m = \ominus(*(nc_2,g_1)))) \vee\\
&\quad (pw(j) \notin DY(ik) \wedge (m = *(nc_2,*(pw(j),g_1)) \vee m = \ominus(*(nc_2,*(pw(j),g_1)))))))).
\end{aligned}$$

3. TCAMP_3$oops_1$ and TCAMP_$oops_1$ link a simple syn^*-message to a lost session key of some TC-AMP run:

$$\begin{aligned}
&\text{TCAMP_3}oops_1(m,ik,pw(i)) \Leftrightarrow\\
&(\exists nc_2,nc_1,g_1,g_2,pw(j) : nc_1,nc_2 \notin DY(pw(i)\bullet ik) \wedge pw(j) \in DY(pw(i)\bullet ik) \wedge\\
&\quad \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))),g_1))) \in ik \wedge\\
&\quad (m = *(nc_2,*(nc_1,g_1)) \vee m = \ominus(*(nc_2,*(nc_1,g_1)))))
\end{aligned}$$

$$\begin{aligned}
&\text{TCAMP_}oops_1(m,ik) \Leftrightarrow\\
&(\exists nc_2,nc_1,g_1,g_2,pw(j) : nc_1,nc_2 \notin DY(ik) \wedge pw(j) \in DY(ik) \wedge\\
&\quad \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))),g_1))) \in ik \wedge\\
&\quad (m = *(nc_2,*(nc_1,g_1)) \vee m = \ominus(*(nc_2,*(nc_1,g_1))))).
\end{aligned}$$

In contrast to TCAMP_$msg_{1,1}$, TCAMP_$msg_{2,1}$ and TCAMP_$oops_1$, where the derivability is defined relative to the immediately observable messages ik only, the derivability in TCAMP_3$msg_{1,1}$, TCAMP_3$msg_{2,1}$ and TCAMP_3$oops_1$ is defined relative to ik extended with some password as needed for the resistance proof.

Using the above predicates, we obtain the following regularity properties:

Property$^{\text{VSE}}$ 140 (Derivable $*$-Messages):

1. $$\begin{aligned}
&(tr \in \text{TCAMP} \wedge isSyn^*(m) \wedge \neg isObj^{\oplus}(m) \wedge m \in DY(pw(i)\bullet spies(tr))) \Rightarrow\\
&((\exists m_1,m_2 : syn^*(m,m_1,m_2) \wedge m_1 \in DY(pw(i)\bullet spies(tr))) \vee\\
&\text{TCAMP_3}msg_{1,1}(m,spies(tr),pw(i)) \vee \text{TCAMP_3}msg_{2,1}(m,spies(tr),pw(i)) \vee\\
&\text{TCAMP_3}oops_1(m,spies(tr),pw(i)))
\end{aligned}$$

2. $(tr \in \text{TCAMP} \wedge isSyn^*(m) \wedge \neg isObj^{\oplus}(m) \wedge m \in DY(spies(tr))) \Rightarrow$
$((\exists m_1, m_2 : syn^*(m, m_1, m_2) \wedge m_1 \in DY(spies(tr))) \vee$
$\text{TCAMP\_}msg_{1,1}(m, spies(tr)) \vee \text{TCAMP\_}msg_{2,1}(m, spies(tr)) \vee$
$\text{TCAMP\_}oops_1(m, spies(tr)))$

The proof of both regularity properties is similar. It is by contradiction, as described in the following for property 140-(1): Using the corresponding negated conclusion, we aim at a refutation of the assumption $m \in DY(pw(i)\bullet spies(tr))$.

The assumed negated conclusion provides:

1. $\forall m_1, m_2 : syn^*(m, m_1, m_2) \Rightarrow m_1 \notin DY(pw(i)\bullet ik)$
2. For all messages $\oplus(*(nc_1, g_1), \ominus(*(pw(j), g_2))) \in ik$ with $nc_1 \notin DY(pw(i)\bullet ik)$ and with $pw(j), g_2 \in DY(pw(i)\bullet ik)$, we have $m \neq *(nc_1, g_1)$ and $m \neq \ominus(*(nc_1, g_1))$.
3. For all $pair(*(nc_2, *(pw(j), g_1)), m_1) \in ik$ where nc_2 is not in $DY(pw(i)\bullet ik)$ and $pw(j)$ in $DY(pw(i)\bullet ik)$, we have $m \neq *(nc_2, g_1)$ and $m \neq \ominus(*(nc_2, g_1))$.
4. For all $pair(*(nc_2, *(pw(j), g_1)), m_1) \in ik$ with $nc_2, pw(j) \notin DY(pw(i)\bullet ik)$, we have $m \neq *(nc_2, *(pw(j), g_1))$ and $m \neq \ominus(*(nc_2, *(pw(j), g_1)))$.
5. For all
$$\oplus(*(nc_2, *(nc_1, g_1)), *(nc_2, *(\oplus(*(nc_1, g_1), \ominus(*(pw(j), g_2))), g_1))) \in ik$$
such that $nc_2, nc_1 \notin DY(pw(i)\bullet ik)$ and $pw(j) \in DY(pw(i)\bullet ik)$ hold, we have $m \neq *(nc_2, *(nc_1, g_1))$ and $m \neq \ominus(*(nc_2, *(nc_1, g_1)))$.

To simplify matters, we focus w.l.o.g. on the case where m is a $*$-object. In case m is a $\ominus$-object with a $*$-object as a $\ominus$-part, $m \in DY(pw(i)\bullet ik)$ yields $\ominus(m) \in DY(pw(i)\bullet ik)$ with $\ominus(m)$ is a $*$-object.

For a $*$-object m, the above assumptions are simplified to:

1. $\forall m_1, m_2 : obj^*(m, m_1, m_2) \Rightarrow m_1 \notin DY(pw(i)\bullet ik)$
2. For all messages $\oplus(*(nc_1, g_1), \ominus(*(pw(j), g_2))) \in ik$ with $nc_1 \notin DY(pw(i)\bullet ik)$ and with $pw(j), g_2 \in DY(pw(i)\bullet ik)$, we have $m \neq *(nc_1, g_1)$.
3. For all $pair(*(nc_2, *(pw(j), g_1)), m_1) \in ik$ where nc_2 is not in $DY(pw(i)\bullet ik)$ and $pw(j)$ in $DY(pw(i)\bullet ik)$, we have $m \neq *(nc_2, g_1)$.
4. For all $pair(*(nc_2, *(pw(j), g_1)), m_1) \in ik$ with $nc_2, pw(j) \notin DY(pw(i)\bullet ik)$, we have $m \neq *(nc_2, *(pw(j), g_1))$.
5. For all
$$\oplus(*(nc_2, *(nc_1, g_1)), *(nc_2, *(\oplus(*(nc_1, g_1), \ominus(*(pw(j), g_2))), g_1))) \in ik$$
such that $nc_2, nc_1 \notin DY(pw(i)\bullet ik)$ and $pw(j) \in DY(pw(i)\bullet ik)$ hold, we have $m \neq *(nc_2, *(nc_1, g_1))$.

The above assumptions (1), (2) and $m \in DY(pw(i)\bullet ik)$ allow us to deduce for all $\oplus(*(nc_1, g_1), \ominus(*(pw(j), g_2))) \in ik$ that m differs from $*(nc_1, g_1)$ and $*(pw(j), g_2)$:

- In case this first TC-AMP message is sent by a compromised agent, nc_1 and $pw(j)$ belong to $DY(ik)$. If m were equal $*(nc_1, g_1)$ or $*(pw(j), g_2)$, assumption (1) would be refuted.

- Otherwise, we have $nc_1 \notin DY(pw(i).ik)$ according to the basic confidentiality property 90-(3). In case $pw(j) \in DY(pw(i).ik)$, the equality $m = *(pw(j),g_2)$ permits to refute assumption (1). Regarding $m \neq *(nc_1,g_1)$, we distinguish two cases: If $g_2 \in DY(pw(i).ik)$ holds, $m \neq *(nc_1,g_1)$ follows from assumption (2). Otherwise, $g_2 \notin DY(pw(i).ik)$ can be used to prove $*(nc_1,g_1),nc_1,g_2 \notin DY(pw(i).ik)$ in a similar way as it is done for the basic confidentiality property 90-(1). That is, the equality $m = *(nc_1,g_1)$ permits to refute $m \in DY(pw(i).ik)$.

The above assumptions (1), (3), (4) and $m \in DY(pw(i).ik)$ allow us to deduce for all $pair(*(nc_2,*(pw(j),g_1)),m_1) \in ik$ that m differs from $*(nc_2,*(pw(j),g_1))$, $*(nc_2,g_1)$ and $*(pw(j),g_1)$:

- In case this second TC-AMP message is sent by a compromised agent, nc_2 and $pw(j)$ belong to $DY(ik)$. If m were equal $*(nc_2,*(pw(j),g_1))$, $*(nc_2,g_1)$ or $*(pw(j),g_1)$, assumption (1) would be refuted.

- Otherwise, we have $nc_2 \notin DY(pw(i).ik)$ according to the basic confidentiality property 90-(4). In case $pw(j)$ is in $DY(pw(i).ik)$, the equalities $m = *(nc_2,*(pw(j),g_1))$ and $m = *(pw(j),g_1)$ permit to refute assumption (1). The negation of the equality $m = *(nc_2,g_1)$ follows from assumption (3). In case $pw(j) \notin DY(pw(i).ik)$, we prove $*(nc_2,g_1),*(pw(j),g_1),nc_2,pw(j) \notin DY(pw(i).ik)$ in a similar way as it is done for the basic confidentiality property 90-(2). That is, the equalities $m = *(nc_2,g_1)$ and $m = *(pw(j),g_1)$ permit to refute $m \in DY(pw(i).ik)$. The negation of the equality $m = *(nc_2,*(pw(j),g_1))$ follows from assumption (4).

The above assumptions (1)–(5) and $m \in DY(pw(i).ik)$ allow us to deduce for all $\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))),g_1))) \in ik$ that m differs from $*(nc_2,*(nc_1,g_1))$, $*(nc_2,g_1)$, $*(nc_1,g_1)$ and $*(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))),g_1))$:

- In the oops-case of TC-AMP, $*(nc_1,g_1)$ occurs in a first TC-AMP message and $*(nc_2,g_1)$ in a second TC-AMP message. That is, we can derive $m \neq *(nc_1,g_1)$, $nc_1,nc_2 \notin DY(pw(i).ik)$, and $m \neq *(nc_2,g_1)$ as explained above.

- If the password $pw(j)$ belongs to $DY(pw(i).ik)$, the negation of $m = *(nc_2,*(nc_1,g_1))$ follows from assumption (5). Otherwise, $pw(j) \notin DY(pw(i).ik)$ allows us to prove $*(nc_2,*(nc_1,g_1)),*(nc_2,g_1),*(nc_1,g_1),*(pw(j),g_1),nc_2,nc_1,pw(j) \notin DY(pw(i).ik)$ in a similar way as it is done for the basic confidentiality property 90-(2). That is, the equality $m = *(nc_2,*(nc_1,g_1))$ permits to refute $m \in DY(pw(i).ik)$.

- The equality $m = *(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(pw(i),g_2))),g_1))$ permits to refute assumption (1), as $*(\oplus(*(nc_1,g_1),\ominus(*(pw(i),g_2))))$ is in ik.

Recapitulating, the above derived dis-equalities of m allow us to refute the assumption $m \in DY(pw(i).spies(tr))$ employing the following property together with the correctness theorem 56:

$$
\begin{aligned}
&(tr \in \text{TC}_{\text{AMP}} \wedge isObj^*(m) \wedge \\
&\quad (\forall m_1,m_2 : obj^*(m,m_1,m_2) \Rightarrow m_1 \notin DY(pw(i).spies(tr))) \wedge \\
&\quad (\forall nc_1,pw(j),g_1,g_2 : \oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))) \in spies(tr) \Rightarrow \\
&\qquad (m \neq *(pw(j),g_2) \wedge m \neq *(nc_1,g_1))) \wedge \\
&\quad (\forall nc_2,pw(j),g_1,m_1 : pair(*(nc_2,*(pw(j),g_1)),m_1) \in spies(tr) \Rightarrow \\
&\qquad (m \neq *(nc_2,*(pw(j),g_1)) \wedge m \neq *(nc_2,g_1) \wedge m \neq *(pw(j),g_1))) \wedge \\
&\quad (\forall nc_2,nc_1,g_1,g_2,pw(j) : \\
&\qquad \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))),g_1))) \in spies(tr) \\
&\qquad \Rightarrow (m \neq *(nc_2,*(nc_1,g_1)) \wedge m \neq *(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))),g_1)))))) \\
&\Rightarrow (pw(i).spies(tr)) \cap ccl_2(m) = \emptyset
\end{aligned}
$$

14.2.3 Derivable $\oplus$-Messages

In this section, we describe the employed regularity property about the derivable $\oplus$-objects.

The $\oplus$-objects with confidential $\oplus$-parts are linked to regular TC-AMP messages, where the different cases are distinguished with the help of dedicated predicates. We use TCAMP_msg_1, TCAMP_$oops$ and TCAMP_mrg, defined in Sec. 9.5.1, when the derivability is defined relative to immediately observable messages ik only. For the regularity property where derivability is defined relative to ik extended with some password, we use the following predicates. We require that the password $pw(i)$ added to ik either does not occur in $DY(ik)$ or within any message in ik, to avoid the handling of cases that are obsolete due to the Ω assumption of the resistance property.

1. TCAMP_$3msg_1$ links a $\oplus$-object to a first TC-AMP message:

$$
\begin{aligned}
&\text{TCAMP\_}3msg_1(m,ik,pw(i)) \Leftrightarrow\\
&((\exists nc_1,g_1,g_2 : \oplus(*(nc_1,g_1),\ominus(*(pw(i),g_2))) \in ik \wedge\\
&\quad nc_1,g_2 \notin DY(pw(i)\bullet ik) \wedge\\
&\quad (m = \oplus(*(nc_1,g_1),\ominus(*(pw(i),g_2))) \vee m = \oplus(*(inv(pw(i)),*(nc_1,g_1)),\ominus(g_2)) \vee\\
&\quad\ m = \oplus(\ominus(*(nc_1,g_1)),*(pw(i),g_2)) \vee m = \oplus(\ominus(*(inv(pw(i))),*(nc_1,g_1)),g_2))) \vee\\
&(\exists nc_1,pw(j),g_1,g_2 : \oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))) \in ik \wedge nc_1,pw(j) \notin DY(pw(i)\bullet ik) \wedge\\
&\quad (m = \oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))) \vee m = \oplus(\ominus(*(nc_1,g_1)),*(pw(j),g_2))))).
\end{aligned}
$$

2. TCAMP_$3mrg$ links a $\oplus$-object to two merged first TC-AMP messages:

$$
\begin{aligned}
&\text{TCAMP\_}3mrg(m,ik,pw(i)) \Leftrightarrow\\
&(\exists nc_1,nc_3,g_1,g_2,pw(j) : nc_1 \neq nc_3 \wedge nc_1,nc_3 \notin DY(pw(i)\bullet ik) \wedge\\
&\quad \oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))),\oplus(*(nc_3,g_1),\ominus(*(pw(j),g_2))) \in ik \wedge\\
&\quad ((pw(j) = pw(i) \wedge g_2 \notin DY(pw(i)\bullet ik)) \vee pw(j) \notin DY(pw(i)\bullet ik)) \wedge\\
&\quad m = \oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1)))).
\end{aligned}
$$

3. TCAMP_$3oops$ links a $\oplus$-object to a lost session key of some TC-AMP run:

$$
\begin{aligned}
&\text{TCAMP\_}3oops(m,ik,pw(i)) \Leftrightarrow\\
&(\exists nc_2,nc_1,g_1,g_2,pw(j) : nc_1,nc_2 \notin DY(pw(i)\bullet ik) \wedge pw(j) \notin DY(pw(i)\bullet ik) \wedge\\
&\quad \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))),g_1))) \in ik \wedge\\
&\quad (m = \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))),g_1))) \vee\\
&\quad\ m = \oplus(*(inv(\oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2)))),*(nc_2,*(nc_1,g_1))),*(nc_2,g_1)) \vee\\
&\quad\ m = \oplus(\ominus(*(nc_2,*(nc_1,g_1))),\ominus(*(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2))),g_1)))) \vee\\
&\quad\ m = \oplus(\ominus(*(inv(\oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2)))),*(nc_2,*(nc_1,g_1)))),\ominus(*(nc_2,g_1))))).
\end{aligned}
$$

Using the above predicates, we obtain the following regularity properties:

Property$^{\text{VSE}}$ 141 (Derivable $\oplus$-Messages):

1. $$
\begin{aligned}
&(tr \in \text{TCAMP} \wedge isObj^{\oplus}(m) \wedge m \in DY(pw(i)\bullet spies(tr)) \wedge pw(i) \notin DY(spies(tr))) \Rightarrow\\
&((\exists m_1,m_2 : (obj^{\oplus}(m,m_1,m_2) \vee syn^{*}(m,m_1,m_2)) \wedge m_1,m_2 \in DY(pw(i)\bullet spies(tr))) \vee\\
&\ \text{TCAMP_}3msg_1(m,spies(tr),pw(i)) \vee \text{TCAMP_}3mrg(m,spies(tr),pw(i)) \vee\\
&\ \text{TCAMP_}3oops(m,spies(tr),pw(i)))
\end{aligned}
$$

2. $(tr \in \text{TCAMP} \wedge isObj^{\oplus}(m) \wedge m \in DY(spies(tr))) \Rightarrow$
$((\exists m_1, m_2 : (obj^{\oplus}(m, m_1, m_2) \vee syn^*(m, m_1, m_2)) \wedge m_1, m_2 \in DY(spies(tr))) \vee$
$\text{TCAMP\_}msg_1(m, spies(tr)) \vee \text{TCAMP\_}mrg(m, spies(tr)) \vee$
$\text{TCAMP\_}oops(m, spies(tr)))$

The proofs of these regularity properties are similar and are based on two similar invariants about the derivable $\oplus$-objects. For the proof of property (2) we use the invariant 94, which is adapted to obtain the following invariant employed for the proof of property (1).

Property[VSE] 142 (Derivable $\oplus$-Objects ; TC-AMP):

$(tr \in \text{TCAMP} \wedge isObj^{\oplus}(\hat{m}) \wedge pw(i) \notin DY(spies(tr)) \wedge$
$DY(pw(i).spies(tr)) \cap ccl_2^{\oplus}(\hat{m}) \neq \emptyset) \Rightarrow$
$(\exists ms_b, ms_p, n : syn^{\overline{\oplus}}(\hat{m}, ms_b \uplus ms_p) \wedge$
$(\forall m \in ms_b : \neg isObj^{\oplus}(m) \wedge m \in DYl(pw(i).spies(tr), n)) \wedge$
$(\forall m \in ms_p : (\exists m', ms :$
$m' \in DYl(pw(i).spies(tr), n) \wedge ms \subseteq DYl(pw(i).spies(tr), n) \wedge$
$syn^{\overline{*}}(m, ms, m') \wedge (\text{TCAMP\_}3msg_1(m', spies(tr), pw(i)) \vee$
$\text{TCAMP\_}3mrg(m', spies(tr), pw(i)) \vee \text{TCAMP\_}3oops(m', spies(tr), pw(i))))))$

Before we discuss the proof of this invariant, we describe how it permits to prove property 141-(1).

- For m in $DY(pw(i).spies(tr))$, the invariant provides two sets ms_b and ms_p where $ms_b \uplus ms_p$ represents a decomposition of m into non-confidential $\oplus$-parts if any.
- If $len(ms_b \uplus ms_p) > 1$, the elements of $ms_b \uplus ms_p$ allow us to obtain $obj^{\oplus}(m, m_1, m_2)$ with $m_1, m_2 \in DY(pw(i).spies(tr))$.
- Otherwise, we necessarily have $ms_b = \emptyset$ and $ms_p = \{m\}$. Additionally, we obtain $syn^{\overline{*}}(m, ms, m_3)$ for $ms \subseteq DYl(pw(i).spies(tr), n)$ and for $m_3 \in DYl(pw(i).ik, n)$ fulfilling case $\text{TCAMP\_}3msg_1(m_3, spies(tr), pw(i))$, $\text{TCAMP\_}3mrg(m_3, spies(tr), pw(i))$ or $\text{TCAMP\_}3oops(m_3, spies(tr), pw(i))$.

  If the multiset ms is not empty, i.e. $ms = m_4 \uplus ms_1$, we obviously have property $syn^*(m, m_4, \overline{*}(ms_1, m_3))$ with $m_4, \overline{*}(ms_1, m_3) \in DY(pw(i).spies(tr))$. Otherwise, we clearly have $m_3 = m$ and this permits to obtain $\text{TCAMP\_}3msg_1(m, spies(tr), pw(i))$, $\text{TCAMP\_}3mrg(m, spies(tr), pw(i))$ or $\text{TCAMP\_}3oops(m, spies(tr), pw(i))$, as required.

Finally, the used invariant is shown similar to the invariant 94, as described in Sec. 6.5.1.2. The same proof schema applies, because the protocol-specific predicates $\text{TCAMP\_}3msg_1$, $\text{TCAMP\_}3mrg$ and $\text{TCAMP\_}3oops$ fulfill the requirements $\aleph_1$ and $\aleph_2$.

Requirement $\aleph_1$ is about the non-confidential left $*$-parts occurring in the $\oplus$-objects that originate from regular messages according to the given predicates. It is fulfilled as described in the following:

- The $\oplus$-objects m that satisfy $\text{TCAMP\_}3mrg(m, ik, pw(i))$ include only confidential left $*$-parts.
- The $\oplus$-objects m that satisfy predicate $\text{TCAMP\_}3msg_1(m, ik, pw(i))$ have just one non-confidential left $*$-part, which corresponds to the employed password $pw(i)$ or to

its inverse $inv(pw(i))$. Here, the definition of TCAMP_3$msg_1(m,ik,pw(i))$ satisfies obviously requirement $\aleph_1$. Take for instance $m = \oplus(*(nc_1,g_1),\ominus(*(pw(i),g_2)))$, requirement $\aleph_1$ necessitates that $*(inv(pw(i)),m)$ is covered by one of the given predicates. This is clearly the case, as TCAMP_3$msg_1(*(inv(pw(i)),m),ik,pw(i))$ ensues from TCAMP_3$msg_1(m,ik,pw(i))$.

- Similarly, the $\oplus$-objects m that satisfy predicate TCAMP_3$oops(m,ik,pw(i))$ have just one non-confidential left $*$-part, which corresponds to a first TC-AMP message $m_1 = \oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2)))$ or to $inv(m_1)$. Here, the definition of TCAMP_3$oops(m,ik,pw(i))$ satisfies obviously requirement $\aleph_1$. Take for instance $m = \oplus(m_{12},*(nc_2,*(m_1,g_1)))$ for $m_{12} = *(nc_1,*(nc_2,g_1))$, requirement $\aleph_1$ necessitates that $*(inv(m_1),m)$ is covered by one of the given predicates. This is clearly the case, as TCAMP_3$oops(*(inv(m_1),m),ik,pw(i))$ ensues from TCAMP_3$oops(m,ik,pw(i))$.

Requirement $\aleph_2$ ensures that all $\oplus$-objects (m_{lr}) resulting by merging other $\oplus$-objects (m_l, m_r) linked to regular protocol messages are covered by the protocol-specific predicates according to this principle: After the simplification of the public common left $*$-submessages of m_{lr} (if any), the resulting m'_{lr} must be covered by the protocol-specific predicates, too. According to the used predicates, we distinguish the following merge-sides:

- $m_1 = \oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2)))$, $m_3 = \oplus(\ominus(*(nc_3,g_1)),*(pw(j),g_2))$ and such that TCAMP_3$msg_1(m_1,ik,pw(i))$ and TCAMP_3$msg_1(m_3,ik,pw(i))$ hold: Here, we have $\oplus(m_1,m_3) = \oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1)))$ and this message is covered by predicate TCAMP_3$mrg(\oplus(m_1,m_3),ik,pw(i))$. It does not possess any common left $*$-submessage.

- $m_1 = \oplus(*(inv(pw(i)),*(nc_1,g_1)),\ominus(g_2))$, $m_3 = \oplus(\ominus(*(inv(pw(i)),*(nc_3,g_1))),g_2)$ and such that TCAMP_3$msg_1(m_1,ik,pw(i))$ and TCAMP_3$msg_1(m_3,ik,pw(i))$ hold: Here, we have $\oplus(m_1,m_3) = \oplus(*(inv(pw(i)),*(nc_1,g_1)),\ominus(*(inv(pw(i)),*(nc_3,g_1))))$ and this message possesses $inv(pw(i))$ as non-confidential common left $*$-submessage and $m'_{13} = \oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1)))$ as the corresponding right $*$-submessage. m'_{13} is covered by TCAMP_3$mrg(\oplus(m_1,m_3),ik,pw(i))$ and does not possess any common left $*$-sub-message.

- Messages $m_{12} = \oplus(*(nc_1,g_1),\ominus(*(nc_2,g_1)))$ and $m_2 = \oplus(*(nc_2,g_1),\ominus(*(pw(j),g_2)))$ satisfying TCAMP_3$mrg(m_{12},ik,pw(i))$ and TCAMP_3$msg_1(m_2,ik,pw(i))$: Here, we have $\oplus(m_{12},m_2) = \oplus(*(nc_1,g_1),\ominus(*(pw(j),g_2)))$ and this message is covered by TCAMP_3$msg_1(\oplus(m_{12},m_2),ik,pw(i))$. It does not possess any common left $*$-submessage.

- Messages $m_{12} = \oplus(*(nc_1,g_1),\ominus(*(nc_2,g_1)))$ and $m_{23} = \oplus(*(nc_2,g_1),\ominus(*(nc_3,g_1)))$ satisfying TCAMP_3$mrg(m_{12},ik,pw(i))$ and TCAMP_3$mrg(m_{23},ik,pw(i))$: Here, we have $\oplus(m_{12},m_{23}) = \oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1)))$ and this message is covered by TCAMP_3$mrg(\oplus(m_{12},m_{23}),ik,pw(i))$. It does not possess any common left $*$-submessage.

14.2.4 Derivable h_1- and h_2-Messages

In this section, we describe the employed regularity properties about the derivable h_1- and h_2-messages.

Property$^{\text{VSE}}$ 143 (Derivable h_1- and h_2-Messages):

1. $(tr \in \text{TCAMP} \wedge h_1(m_1,m_2,m_3) \in DY(pw(i).spies(tr))) \Rightarrow$
   $((m_1,m_2,m_3 \in DY(pw(i).spies(tr))) \vee h_1(m_1,m_2,m_3) \in DY(spies(tr)))$

2. $(tr \in \text{TCAMP} \wedge h_1(m_1,m_2,m_3) \in DY(spies(tr))) \Rightarrow$
$((m_1,m_2,m_3 \in DY(spies(tr))) \vee$
$(pair(m_2,h_1(m_1,m_2,m_3)) \in spies(tr) \wedge m_3 \notin DY(pw(i)\bullet spies(tr))))$

3. $(tr \in \text{TCAMP} \wedge h_2(m_1,m_2,m_3) \in DY(pw(i)\bullet spies(tr))) \Rightarrow$
$((m_1,m_2,m_3 \in DY(pw(i)\bullet spies(tr))) \vee h_2(m_1,m_2,m_3) \in DY(spies(tr)))$

4. $(tr \in \text{TCAMP} \wedge h_2(m_1,m_2,m_3) \in DY(spies(tr))) \Rightarrow$
$((m_1,m_2,m_3 \in DY(spies(tr))) \vee$
$(h_2(m_1,m_2,m_3) \in spies(tr) \wedge m_3 \notin DY(pw(i)\bullet spies(tr))))$

The proof of property (1) is similar to that of property 118-(1). The refutation of the assumption $h_1(m_1,m_2,m_3) \in DY(pw(i)\bullet spies(tr))$ is performed employing the following property together with a correctness theorem similar to 54:

$$(tr \in \text{TCAMP} \wedge \neg(m_1,m_2,m_3 \in DY(pw(i)\bullet spies(tr))) \wedge h_1(m_1,m_2,m_3) \notin DY(spies(tr))) \Rightarrow (pw(i)\bullet spies(tr)) \cap ccl_2(h_1(m_1,m_2,m_3)) = \emptyset$$

Property (2) is shown by case distinction: If $h_1(m_1,m_2,m_3)$ can be derived from h_1-parts in $DY(ik)$, the proof is trivial. Otherwise, we distinguish again two complementary cases:

- In case $pair(m_2,h_1(m_1,m_2,m_3)) \in ik$ holds, the assumption $\neg(m_1,m_2,m_3 \in DY(ik))$ permits to deduce the required events in tr and the required conditions on the involved agents for the application of the forward secrecy 100. It allows us to obtain $m_3 \notin DY(pw(i)\bullet ik)$, as required.
- In case $pair(m_2,h_1(m_1,m_2,m_3)) \notin ik$ holds, we refute the assumption $h_1(m_1,m_2,m_3) \in DY(ik)$, employing the following property:

$$(tr \in \text{TCAMP} \wedge \neg(m_1,m_2,m_3 \in DY(spies(tr))) \wedge pair(m_2,h_1(m_1,m_2,m_3)) \notin spies(tr)) \Rightarrow spies(tr) \cap ccl_2(h_1(m_1,m_2,m_3)) = \emptyset$$

Properties (3) and (4) are shown similar to properties (1) and (2), respectively.

14.2.5 Derivable fst- and snd-Messages

In this section, we describe the employed regularity properties about the derivable composed messages that can be derived *only* by composition.

According to the regular messages in TC-AMP, the derivable fst- and snd-objects do not possess confidential fst- and respectively snd-parts:

Property$^{\text{VSE}}$ 144 (Trivial Composed Messages):

1. $(tr \in \text{TCAMP} \wedge obj^{fst}(m,m_1) \wedge m \in DY(pw(i)\bullet spies(tr))) \Rightarrow m_1 \in DY(pw(i)\bullet spies(tr))$

2. $(tr \in \text{TCAMP} \wedge obj^{fst}(m,m_1) \wedge m \in DY(spies(tr))) \Rightarrow m_1 \in DY(spies(tr))$

3. $(tr \in \text{TCAMP} \wedge obj^{snd}(m,m_1) \wedge m \in DY(pw(i)\bullet spies(tr))) \Rightarrow m_1 \in DY(pw(i)\bullet spies(tr))$

4. $(tr \in \text{TCAMP} \wedge obj^{snd}(m,m_1) \wedge m \in DY(spies(tr))) \Rightarrow m_1 \in DY(spies(tr))$

Properties 1–4 are shown similar to properties 3–6 in Sec. 12.2.5.

14.3 Proof of the Basis Simulation Relation Lemma

The basis simulation relation lemma is shown (similarly to 107) by induction on TC-AMP traces.

The base case is trivial: The empty trace does not belong to the relevant traces, because it does not include neither a first nor a second TC-AMP message.

In the step case, we distinguish whether there is a regular occurrence of π in ik.

Case I: If there is a regular occurrence of π in ik, the induction hypothesis provides xy_{ik} that satisfies $\Phi(xy_{ik},ik,\pi,\pi')$. For the extended observable messages $ik_{ex} = ik \cup ik_{ev}$, we need to define xy_{ex} relative to ik and ik_{ev} and show that $\Phi(xy_{ex},ik \cup ik_{ev},\pi,\pi')$ holds:

1. In case $ev = note(ag',pair(ag(i),pair(pw(i),pair(num(j),num(k)))))$ and $ag' \in bad$, this event yields $ik_{ev} = \{pair(ag(i),pair(pw(i),pair(num(j),num(k))))\}$ and we distinguish two cases:

   (a) If $pw(i) \notin DY(ik)$, we define xy_{ex} by adapting xy_{ik} through

   - removing all pairs (m,m) for all $m = \oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1)))$ and all first messages $\oplus(*(nc_1,g_1),\ominus(*(pw(i),g_2))),\oplus(*(nc_3,g_1),\ominus(*(pw(i),g_2))) \in ik$ with $g_1 = num(j)$ and $g_2 = num(k)$,
   - replacing all pairs (m,m) and $(\ominus(m),\ominus(m))$ for all first messages $m = \oplus(*(nc_1,g_1),\ominus(*(pw(i),g_2)))$, generator $g_1 = num(j)$ and $g_2 = num(k)$ with $(*(nc_1,g_1),*(nc_1,g_1))$ and $(\ominus(*(nc_1,g_1)),\ominus(*(nc_1,g_1)))$,
   - replacing every (m,m) and $(\ominus(m),\ominus(m))$ for $m = *(pw(i),*(nc_2,g_1))$ and $g_1 = num(j)$ with $(*(nc_2,g_1),*(nc_2,g_1))$ and $(\ominus(*(nc_2,g_1)),\ominus(*(nc_2,g_1)))$,
   - removing every $(inv(m),inv(m))$, (m_1,m_1) and every $(\ominus(m_1),\ominus(m_1))$ for $m = \oplus(*(nc_1,g_1),\ominus(*(pw(i),g_2)))$, $g_1 = num(j)$, $g_2 = num(k)$ and $m_1 = \oplus(*(inv(m),*(nc_2,*(nc_1,g_1))),*(nc_2,g_1))$,
   - replacing every (m,m) and $(\ominus(m),\ominus(m))$ for

     $$m = \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(pw(i),g_2))),g_1)))$$

     $g_1 = num(j)$ and $g_2 = num(k)$ with $(*(nc_2,*(nc_1,g_1)),*(nc_2,*(nc_1,g_1)))$ and $(\ominus(*(nc_2,*(nc_1,g_1))),\ominus(*(nc_2,*(nc_1,g_1))))$,
   - adding for every (m,m) with $m = \oplus(*(nc_1,g_1),\ominus(*(\pi,g_2)))$ and $g_2 = num(j)$ or $g_2 = num(k)$ the pairs
     - $(\ominus(g_2),\ominus(g_2))$, $(\ominus(*(\pi,g_2)),\ominus(*(\pi',g_2)))$, $(*(\pi,g_2),*(\pi',g_2))$,
     - $(*(inv(\pi),*(nc_1,g_1)),$ $\oplus(\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(\pi,g_2)))),g_2))$,
     - $(\ominus(*(inv(\pi),*(nc_1,g_1))),$ $\oplus(\oplus(\ominus(*(inv(\pi'),*(nc_1,g_1)))),*(inv(\pi'),*(\pi,g_2))),\ominus(g_2)))$,
     - $(*(nc_1,g_1),\oplus(\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2))),*(\pi',g_2)))$ and

– $(\ominus(*(nc_1,g_1)),\oplus(\oplus(\ominus(*(nc_1,g_1)),*(\pi,g_2)),\ominus(*(\pi',g_2))))$,

- and finally including $(ag(i),ag(i))$, $(pw(i),pw(i))$, $(num(j),num(j))$ and $(num(k),num(k))$.

(b) Otherwise, we define xy_{ex} by adapting xy_{ik} through including $(ag(i),ag(i))$, if not any.

In case $ev = note(ag, pair(pw(i), pair(num(j),num(k))))$ and $ag \in bad$, we proceed according to the same principle.

2. In case

$$ev = send(ag,ag',nc_1,\oplus(*(nc_1,g_1),\ominus(*(pw,g_2)))),$$

we have

$$ik_{ev} = \{nc_1,\oplus(*(nc_1,g_1),\ominus(*(pw,g_2)))\}, \text{ if } ag \in bad$$
$$ik_{ev} = \{\oplus(*(nc_1,g_1),\ominus(*(pw,g_2)))\}, \text{ otherwise.}$$

In the former case, we define xy_{ex} by

$$xy_{ex} := xy_{ik} \cup \{(nc_1,nc_1)\}.$$

In the latter case, we distinguish the following cases:

(a) If $pw = \pi$, we define xy_{ex} by adapting xy_{ik} through

- adding the pairs (m_1,m_1), $(\ominus(m_1),\ominus(m_1))$, (m_2,m_3) and $(\ominus(m_2),\ominus(m_3))$ for every (m,m) in xy_{ik} with $m = \oplus(*(nc_3,g_1),\ominus(*(\pi,g_2)))$ and for $m_1 = \oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1)))$, $m_2 = *(inv(\pi),m_1)$ and $m_3 = *(inv(\pi'),m_1)$,
- adding (m,m) and $(\ominus(m),\ominus(m))$ for $m = \oplus(*(nc_1,g_1),\ominus(*(\pi,g_2)))$,
- adding $(inv(\pi),inv(\pi'))$, (m_1,m_2) and $(\ominus(m_1),\ominus(m_2))$ for

$$m_1 = \oplus(*(inv(\pi),*(nc_1,g_1)),\ominus(g_2)),$$
$$m_2 = \oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(\pi,g_2)))),$$

- and when $g_2 \in DY(ik)$, adding the pairs $(\ominus(g_2),\ominus(g_2))$, $(*(\pi,g_2),*(\pi',g_2))$, $(\ominus(*(\pi,g_2)),\ominus(*(\pi',g_2)))$, together with (m_1,m_2) and $(\ominus(m_1),\ominus(m_2))$ for

$$m_1 = *(inv(\pi),*(nc_1,g_1))$$
$$m_2 = \oplus(\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(\pi,g_2)))),g_2)$$

and together with (m_3,m_4) and $(\ominus(m_3),\ominus(m_4))$ for

$$m_3 = *(nc_1,g_1)$$
$$m_4 = \oplus(\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2))),*(\pi',g_2)).$$

(b) If $pw \neq \pi$ and $pw \notin DY(ik)$, we define xy_{ex} by adapting xy_{ik} through

- adding (m_1,m_1) and $(\ominus(m_1),\ominus(m_1))$ for every (m,m) in xy_{ik} with $m = \oplus(*(nc_3,g_1),\ominus(*(pw,g_2)))$ and for $m_1 = \oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1)))$,
- and adding (m,m) and $(\ominus(m),\ominus(m))$ for $m = \oplus(*(nc_1,g_1),\ominus(*(pw,g_2)))$.

(c) If $pw \neq \pi$ and $pw,g_2 \in DY(ik)$, we define xy_{ex} by

$$xy_{ex} := xy_{ik} \cup \{(*(nc_1,g_1),*(nc_1,g_1)),(\ominus(*(nc_1,g_1)),\ominus(*(nc_1,g_1)))\}.$$

3. In case

$$ev = send(ag,ag',nc_2,pair(*(nc_2,*(pw,g_1)),h_1(m_1,*(nc_2,*(pw,g_1)),m_3))),$$

we have $ik_{ev} = \{nc_2,pair(*(nc_2,*(pw,g_1)),h_1(m_1,*(nc_2,*(pw,g_1)),m_3))\}$, if $ag \in bad$ and $ik_{ev} = \{pair(*(nc_2,*(pw,g_1)),h_1(m_1,*(nc_2,*(pw,g_1)),m_3))\}$, otherwise. In the former case, we define xy_{ex} by

$$xy_{ex} := xy_{ik} \cup \{(nc_2,nc_2)\}.$$

In the latter case, we distinguish the following cases:

   (a) If $pw = \pi$, we define xy_{ex} by adapting xy_{ik} through
   - adding (m,m) and $(\ominus(m),\ominus(m))$ for $m = *(nc_2,*(pw,g_1))$,
   - adding $(inv(\pi),inv(\pi'))$, (m_1,m_2) and $(\ominus(m_1),\ominus(m_2))$ for $m_1 = *(nc_2,g_1)$ and $m_2 = *(inv(\pi'),*(\pi,*(nc_2,g_1)))$,
   - and, if $m_3 \notin DY(ik)$, adding (m,m) for $m = h_1(m_1,*(nc_2,*(pw,g_1)),m_3)$.

   (b) If $pw \neq \pi$ and $pw \notin DY(ik)$, we define xy_{ex} by adapting xy_{ik} through
   - adding (m,m) and $(\ominus(m),\ominus(m))$ for $m = *(nc_2,*(pw,g_1))$,
   - and, if $m_3 \notin DY(ik)$, adding (m,m) for $m = h_1(m_1,*(nc_2,*(pw,g_1)),m_3)$.

   (c) If $pw \neq \pi$ and $pw \in DY(ik)$, we define xy_{ex} by adapting xy_{ik} through
   - adding (m,m) and $(\ominus(m),\ominus(m))$ for $m = *(nc_2,g_1)$,
   - and, if $m_3 \notin DY(ik)$, adding (m,m) for $m = h_1(m_1,*(nc_2,*(pw,g_1)),m_3)$.

4. In case $ev = says(ag,ag',h_2(m_1,m_2,m_3))$, we have
$ik_{ev} = \{h_2(m_1,m_2,m_3)\}$. Here, we distinguish two cases:

   (a) If $m_3 \notin DY(ik)$, we define xy_{ex} by

$$xy_{ex} := xy_{ik} \cup \{(h_2(m_1,m_2,m_3),h_2(m_1,m_2,m_3))\}.$$

   (b) Otherwise, we set $xy_{ex} = xy_{ik}$.

5. In case $ev = note(spy,\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[pw,g_2],g_1))))$ for $m[pw,g_2] = \oplus(*(nc_1,g_1),\ominus(*(pw,g_2)))$, we have
$ik_{ev} = \{\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[pw,g_2],g_1)))\}$.

   If nc_1 or nc_2 is not confidential, then we set $xy_{ex} = xy_{ik}$. Otherwise, we distinguish two cases:

   (a) If $pw \in DY(ik)$, we define xy_{ex} by adapting xy_{ik} through
   - removing (m,m) for $m = h_1(m_1,m_2,m_3)$ or $m = h_2(m_1,m_2,m_3)$ with $m_3 = \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[pw,g_2],g_1)))$,
   - and adding (m,m) and $(\ominus(m),\ominus(m))$ for $m = *(nc_2,*(nc_1,g_1))$.

   (b) If $pw \notin DY(ik)$, we define xy_{ex} by adapting xy_{ik} through
   - removing (m,m) for $m = h_1(m_1,m_2,m_3)$ or $m = h_2(m_1,m_2,m_3)$ with $m_3 = \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[pw,g_2],g_1)))$,
   - adding $(inv(m[pw,g_2]),inv(m[pw,g_2]))$, (m,m) and $(\ominus(m),\ominus(m))$ for the message $m = \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[pw,g_2],g_1)))$ and for the message $m = \oplus(*(inv(m[pw,g_2]),*(nc_2,*(nc_1,g_1))),*(nc_2,g_1))$,

- and, if $pw = \pi$, by adding (m_1, m_2) and $(\ominus(m_1), \ominus(m_2))$ for

$$
\begin{aligned}
&m_1 = *(nc_2, *(nc_1, g_1)) \text{ and}\\
&m_2 = \oplus(\oplus(*(nc_2, *(nc_1, g_1)), *(nc_2, *(m[\pi, g_2], g_1))),\\
&\qquad \ominus(*(m[\pi, g_2], *(inv(\pi'), *(\pi, *(nc_2, g_1))))))),\\
&m_1 = *(m[\pi, g_2], *(nc_2, g_1)) \text{ and}\\
&m_2 = *(m[\pi, g_2], *(inv(\pi'), *(\pi, *(nc_2, g_1)))),\\
&m_1 = *(m[\pi, g_2], *(\pi, *(nc_2, g_1))) \text{ and}\\
&m_2 = *(m[\pi, g_2], *(\pi, *(nc_2, g_1))),\\
&m_1 = *(inv(m[\pi, g_2]), *(nc_2, *(nc_1, g_1))) \text{ and}\\
&m_2 = \oplus(\oplus(*(inv(m[\pi, g_2]), *(nc_2, *(nc_1, g_1))), *(nc_2, g_1)),\\
&\qquad \ominus(*(inv(\pi'), *(\pi, *(nc_2, g_1))))),\\
&m_1 = *(\pi, *(nc_2, *(nc_1, g_1))) \text{ and}\\
&m_2 = \oplus(\oplus(*(\pi', *(nc_2, *(nc_1, g_1))), *(\pi', *(nc_2, *(m[\pi, g_2], g_1)))),\\
&\qquad \ominus(*(m[\pi, g_2], *(\pi, *(nc_2, g_1))))),\\
&m_1 = *(\pi, *(inv(m[\pi, g_2]), *(nc_2, *(nc_1, g_1)))) \text{ and}\\
&m_2 = \oplus(\oplus(*(\pi', *(inv(m[\pi, g_2]), *(nc_2, *(nc_1, g_1)))), *(\pi', *(nc_2, g_1))),\\
&\qquad \ominus(*(\pi, *(nc_2, g_1)))),\\
&m_1 = \oplus(*(\pi, *(nc_2, *(nc_1, g_1))), *(m[\pi, g_2], *(\pi, *(nc_2, g_1)))) \text{ and}\\
&m_2 = \oplus(*(\pi', *(nc_2, *(nc_1, g_1))), *(\pi', *(nc_2, *(m[\pi, g_2], g_1)))), \text{ and for}\\
&m_1 = \oplus(*(\pi, *(inv(m[\pi, g_2]), *(nc_2, *(nc_1, g_1)))), *(\pi, *(nc_2, g_1))) \text{ and}\\
&m_2 = \oplus(*(\pi', *(inv(m[\pi, g_2]), *(nc_2, *(nc_1, g_1)))), *(\pi', *(nc_2, g_1))).
\end{aligned}
$$

6. In the fake case, i.e. $ik_{ev} = \{m\}$ for $m \in DY(ik)$, we set $xy_{ex} = xy_{ik}$.

In all these cases, we prove that $\Phi(xy_{ex}, ik \cup ik_{ev}, \pi, \pi')$ ensues from $\Phi(xy_{ik}, ik, \pi, \pi')$ and the corresponding case context. In particular, we have to show $\Phi(xy_{ik}, ik \cup \{m\}, \pi, \pi')$ in the fake case, which is mainly done with the help of the regularity properties (in Sec. 14.2) and the definition of Φ: We consider the cases where m matches any assumption of the rules Φ_1–Φ_8; Then, we apply in every case a corresponding regularity property permitting to obtain an equivalent message in ik; This implies that each pair that shall be added due to m (from $DY(ik)$) is already in xy_{ik}, as required.

For instance, the first rule of Φ_4 provides the message $m = \oplus(*(nc_1, g_1), \ominus(*(pw, g_2)))$ with $nc_1 \notin DY(ik \cup \{m\})$ and $pw, g_2 \in DY(ik \cup \{m\})$ and necessitates to show that $(*(nc_1, g_1), *(nc_1, g_1))$ and $(\ominus(*(nc_1, g_1)), \ominus(*(nc_1, g_1)))$ are in xy_{ex}, i.e. in xy_{ik}. Since $m \in DY(ik)$ we have $nc_1 \notin DY(ik)$, $pw, g_2 \in DY(ik)$ and $*(nc_1, g_1) \in DY(ik)$. This permits to apply the regularity property 140-(2) about the occurrence of $*$-objects in $DY(spies(tr))$. Here, $*(nc_1, g_1)$ matches only the cases for the first and the second TC-AMP steps:

- For the first TC-AMP step, we obtain $\oplus(*(nc_1, g_1), \ominus(pw', g_2')) \in ik$ where pw' and g_2' are in $DY(ik)$. This permits to get $(*(nc_1, g_1), *(nc_1, g_1)) \in xy_{ik}$ with the help of Φ_4 and then $(\ominus(*(nc_1, g_1)), \ominus(*(nc_1, g_1))) \in xy_{ik}$ with the help of Φ_0.

- For the second TC-AMP step, we obtain

$$pair(*(nc_2, *(pw', g_1)), h_1(m_1, *(nc_2, *(pw', g_1)), m_3)) \in ik \wedge pw' \in DY(ik).$$

  This permits to get $(*(nc_1, g_1), *(nc_1, g_1)) \in xy_{ik}$ with the help of Φ_5 and then $(\ominus(*(nc_1, g_1)), \ominus(*(nc_1, g_1))) \in xy_{ik}$ with the help of Φ_0.

Case II: In the complementary case, i.e. ik does not include neither a first nor a second TC-AMP message using π, we proceed as follows:

1. There is nc_1, g_1, g_2 with $ik_{ev} = \{\oplus(*(nc_1, g_1), \ominus(*(pw, g_2)))\}$: First, we use a lemma that provides a set xy_{in} containing $(\pi, \pi'), (\infty, \infty)$ and all pairs obtained according to the inclusion rules Φ_2, Φ_3, Φ_4 (`msg1-1` and `msg1-2`), Φ_5–Φ_8 and Φ_0 relative to ik. Then, we set $xy_{ex} = xy_{in} \cup xy_{l1} \cup xy_{l2}$, where

$$
\begin{aligned}
xy_{l1} = \{&(\oplus(*(nc_1, g_1), \ominus(*(pw, g_2))), \oplus(*(nc_1, g_1), \ominus(*(pw, g_2)))),\\
&(\oplus(\ominus(*(nc_1, g_1)), *(pw, g_2)), \oplus(\ominus(*(nc_1, g_1)), *(pw, g_2))),\\
&(\oplus(*(inv(\pi), *(nc_1, g_1)), \ominus(g_2)), \oplus(*(inv(\pi'), *(nc_1, g_1)),\\
&\quad \ominus(*(inv(\pi'), *(\pi, g_2))))),\\
&(\oplus(*(inv(\pi), *(nc_1, g_1)), \ominus(g_2)), \oplus(*(inv(\pi'), *(nc_1, g_1)),\\
&\quad \ominus(*(inv(\pi'), *(\pi, g_2))))),\\
&(\oplus(\ominus(*(inv(\pi), *(nc_1, g_1))), g_2), \oplus(\ominus(*(inv(\pi'), *(nc_1, g_1))),\\
&\qquad *(inv(\pi'), *(\pi, g_2)))),\\
&(inv(\pi), inv(\pi'))\},
\end{aligned}
$$

$xy_{l2} = \emptyset$, if $g_2 \notin DY(ik)$, and

$$
\begin{aligned}
xy_{l2} = \{&(\ominus(g_2), \ominus(g_2)), (*(\pi, g_2)), *(\pi', g_2)), (\ominus(*(\pi, g_2)), \ominus(*(\pi', g_2))),\\
&(*(inv(\pi), *(nc_1, g_1)),\\
&\quad \oplus(\oplus(*(inv(\pi'), *(nc_1, g_1)), \ominus(*(inv(\pi'), *(\pi, g_2)))), g_2)),\\
&(\ominus(*(inv(\pi), *(nc_1, g_1)))),\\
&\quad \oplus(\oplus(\ominus(*(inv(\pi'), *(nc_1, g_1)))), *(inv(\pi'), *(\pi, g_2))), \ominus(g_2))),\\
&(*(nc_1, g_1), \oplus(\oplus(*(nc_1, g_1), \ominus(*(\pi, g_2))), *(\pi', g_2))),\\
&(\ominus(*(nc_1, g_1)), \oplus(\oplus(\ominus(*(nc_1, g_1)), *(\pi, g_2)), \ominus(*(\pi', g_2))))\}, \text{ otherwise.}
\end{aligned}
$$

2. There is g_1, nc_2, m_1 and m_2 with $ik_{ev} = \{pair(*(nc_2, *(\pi, g_1)), h_1(m_1, *(nc_2, *(\pi, g_1)), m_2))\}$: First, we use a lemma that provides a set xy_{in} containing $(\pi, \pi'), (\infty, \infty)$ and all pairs obtained according to Φ_2–Φ_4, Φ_5 (`msg2-1`, `msg2-2` and `msg2-3`), Φ_6–Φ_8 and Φ_0 relative to ik. Then, we set $xy_{ex} = xy_{in} \cup xy_{l1} \cup xy_{l2}$, where

$$
\begin{aligned}
xy_{l1} = \{&(*(\pi, *(nc_2, g_1)), *(\pi, *(nc_2, g_1))),\\
&(\ominus(*(\pi, *(nc_2, g_1))), \ominus(*(\pi, *(nc_2, g_1)))),\\
&(*(nc_2, g_1), *(inv(\pi'), *(\pi, *(nc_2, g_1)))),\\
&(\ominus(*(nc_2, g_1)), \ominus(*(inv(\pi'), *(\pi, *(nc_2, g_1))))), (inv(\pi), inv(\pi'))\},
\end{aligned}
$$

$xy_{l2} = \emptyset$, if $m_2 \in DY(ik)$, and

$$xy_{l2} = \{(h_1(m_1, *(\pi, *(nc_2, g_1)), m_2), h_1(m_1, *(\pi, *(nc_2, g_1)), m_2))\}, \text{ otherwise.}$$

3. The complementary case is handled similar to the base case. In the fake case, we need additionally to show that m (from $DY(ik)$) does not match the previous cases (1) and (2). This is done with help of the regularity properties 141-(2) and 140-(2). They permit to obtain a first or a second TC-AMP message in ik with a protected occurrence of π, in case m (from $DY(ik)$) matches a similar TC-AMP message. This

permits to conclude by refutation, as ik does not include neither a first nor a second TC-AMP message using π.

In cases (1) and (2), we prove that $\Phi(xy_{ex}, ik \cup ik_{ev}, \pi, \pi')$ ensues from the applied lemma and the corresponding case context.

14.4 Handling of the Proof Obligations

In this section, we describe how the proof obligations are shown from the assumption Ω, the definition Φ of the basis relations (in Sec. 14.1) and the regularity properties (in Sec. 14.2).

14.4.1 Handling of Ψ^a:

Proof obligation Ψ^a is shown with the help of the regularity properties in 139. They provide for each kind of atomic messages c_i in $DY(\pi\bullet ik)$ (resp. $DY(\pi'\bullet ik)$) the premises for the inclusion rules of Φ, which in turn imply the occurrence of c_i in the domain (resp. codomain) of xy.

- The case for agent names $ag(j)$ is handled as sketched in the following table:

$ag(j) \in DY(\pi\bullet ik) \stackrel{?}{\vdash} ag(j) \in dom(xy)$ and $ag(j) \in DY(\pi'\bullet ik) \stackrel{?}{\vdash} ag(j) \in codom(xy)$
Property 139-(1) yields one case: $pair(ag(j), pair(pw(j), pair(num(k), num(k')))) \in ik \vdash^{\Phi_2} (ag(j), ag(j)) \in xy$

- The case for numerical data $num(j)$ is handled as sketched in the following table:

$num(j) \in DY(\pi\bullet ik) \stackrel{?}{\vdash} num(j) \in dom(xy)$ and $num(j) \in DY(\pi'\bullet ik) \stackrel{?}{\vdash} num(j) \in codom(xy)$
Property 139-(2) yields four cases: (1) $pair(ag(k), pair(pw(k), pair(num(j), num(j')))) \in ik \vdash^{\Phi_2} (num(j), num(j)) \in xy$ (2) $pair(ag(k), pair(pw(k), pair(num(j'), num(j)))) \in ik \vdash^{\Phi_2} (num(j), num(j)) \in xy$ (3) $pair(pw(k), pair(num(j), num(j'))) \in ik \vdash^{\Phi_2} (num(j), num(j)) \in xy$ (4) $pair(pw(k), pair(num(j'), num(j))) \in ik \vdash^{\Phi_2} (num(j), num(j)) \in xy$

- The case for passwords $pw(j)$ is handled as sketched in the following table:

$pw(j) \in DY(\pi\bullet ik) \stackrel{?}{\vdash} pw(j) \in dom(xy)$
Property 139-(3) yields three cases: (1) $pw(j) = \pi \vdash^{\Omega,\Phi_1} (pw(j), \pi') \in xy$ (2) $pair(ag(j), pair(pw(j), pair(num(k), num(k')))) \in ik \vdash^{\Phi_2} (pw(j), pw(j)) \in xy$ (3) $pair(pw(j), pair(num(k), num(k'))) \in ik \vdash^{\Phi_2} (pw(j), pw(j)) \in xy$
$pw(j) \in DY(\pi'\bullet ik) \stackrel{?}{\vdash} pw(j) \in codom(xy)$
Property 139-(3) yields three cases: (1) $pw(j) = \pi' \vdash^{\Omega,\Phi_1} (\pi, pw(j)) \in xy$ (2) $pair(ag(j), pair(pw(j), pair(num(k), num(k')))) \in ik \vdash^{\Phi_2} (pw(j), pw(j)) \in xy$ (3) $pair(pw(j), pair(num(k), num(k'))) \in ik \vdash^{\Phi_2} (pw(j), pw(j)) \in xy$

  In case $pw(j) = \pi$ (resp. $pw(j) = \pi'$), the assumption Ω ensures that ik includes the regular messages necessary for the application of Φ_1.

- The case for nonces $nc(j)$ is handled as sketched in the following table:

$nc(j) \in DY(\pi.ik) \stackrel{?}{\vdash} nc(j) \in dom(xy)$ and $nc(j) \in DY(\pi'.ik) \stackrel{?}{\vdash} nc(j) \in codom(xy)$
Property 139-(4) yields one case: $nc(j) \in ik \stackrel{\Phi_3}{\vdash} (nc(j),nc(j)) \in xy$

14.4.2 Handling of Ψ^c:

According to proof obligation Ψ^c, all derivable composed messages that cannot be derived by composition must occur in the domain (resp. codomain) of the basis relation xy. This is shown with the help of the regularity properties about the derivable composed messages.

The regularity properties about $*$-, $\oplus$-, h_1- and h_2-messages provide the premises for corresponding inclusion rules in Φ_4–Φ_8, which in turn imply the occurrence of these composed messages in the domain (resp. codomain) of xy.

- The proof for $m \in dom(xy)$ where m is a $*$-object is reduced with the help of the regularity property 140-(1) to the cases given by the definitions of TCAMP_3$msg_{1,1}$, TCAMP_3$msg_{2,1}$ and TCAMP_3$oops_1$.

  The proof for the TCAMP_3$msg_{1,1}$-case (using in particular Φ_4) is sketched in the following table:

$isObj^*(m), m \in DY(\pi.ik), (obj^*(m,m_1,m_2) \Rightarrow m_1 \notin DY(\pi.ik)) \stackrel{?}{\vdash} m \in dom(xy)$
Property 140-(1) and TCAMP_3$msg_{1,1}$ yield: $\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))) \in ik, nc_1 \notin DY(\pi.ik), (pw,g_2 \in DY(\pi.ik)),$ $m = *(nc_1,g_1) \stackrel{?}{\vdash} m \in dom(xy)$ Properties 139-(3) and -(2) yield two cases: (1) $\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))) \in ik, nc_1 \notin DY(ik), (pw,g_2 \in DY(ik)),$ $m = *(nc_1,g_1) \stackrel{\Phi_4}{\vdash} (m,m) \in xy$ (2) $\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2))) \in ik, nc_1 \notin DY(ik), g_2 \in DY(ik),$ $m = *(nc_1,g_1) \stackrel{\Phi_4}{\vdash} (m,\oplus(\oplus(m,\ominus(*(\pi,g_2))),*(\pi',g_2))) \in xy$

  The proof for the TCAMP_3$msg_{2,1}$-case (using in particular Φ_5) is sketched in the following table:

$isObj^*(m), m \in DY(\pi.ik), (obj^*(m,m_1,m_2) \Rightarrow m_1 \notin DY(\pi.ik)) \stackrel{?}{\vdash} m \in dom(xy)$
Property 140-(1) and TCAMP_3$msg_{2,1}$ yield two cases: (1) $pair(*(nc_2,*(pw,g_1)),m_1) \in ik, nc_2 \notin DY(\pi.ik), pw \in DY(\pi.ik),$ $m = *(nc_2,g_1) \stackrel{?}{\vdash} m \in dom(xy)$ (2) $pair(*(nc_2,*(pw,g_1)),m_1) \in ik, nc_2, pw \notin DY(\pi.ik), m = *(nc_2,*(pw,g_1))$ $\stackrel{?}{\vdash} m \in dom(xy)$ Property 139-(3) transforms (1) into two cases: (1.1) $pair(*(nc_2,*(pw,g_1)),m_1) \in ik, nc_2 \notin DY(ik), pw \in DY(ik),$ $m = *(nc_2,g_1) \stackrel{\Phi_5}{\vdash} (m,m) \in xy$ (1.2) $pair(*(nc_2,*(\pi,g_1)),m_1) \in ik, m = *(nc_2,g_1)$ $\stackrel{\Phi_5}{\vdash} (m,*(inv(\pi'),*(\pi,m))) \in xy$ (2) $pair(*(nc_2,*(pw,g_1)),m_1) \in ik, nc_2, pw \notin DY(ik), m = *(nc_2,*(pw,g_1))$ $\stackrel{\Phi_5}{\vdash} (m,m) \in xy$

  The proof for the TCAMP_3$oops_1$-case (using in particular Φ_7) is sketched in the following table:

$isObj^*(m), m \in DY(\pi\bullet ik), (obj^*(m,m_1,m_2) \Rightarrow m_1 \notin DY(\pi\bullet ik)) \stackrel{?}{\vdash} m \in dom(xy)$
Property 140-(1) and TCAMP_3$oops_1$ yields: $\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),g_1))) \in ik,$ $(nc_1,nc_2 \notin DY(\pi\bullet ik)), pw \in DY(\pi\bullet ik), m = *(nc_2,*(nc_1,g_1)) \stackrel{?}{\vdash} m \in dom(xy)$ Property 139-(3) yields two cases: (1) $\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),g_1))) \in ik,$ $(nc_1,nc_2 \notin DY(ik)), pw \in DY(ik), m = *(nc_2,*(nc_1,g_1)) \stackrel{\Phi_7}{\vdash} (m,m) \in xy$ (2) $\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2))),g_1))) \in ik,$ $m = *(nc_2,*(nc_1,g_1)) \stackrel{\Phi_7}{\vdash} (m,\oplus(\oplus(m,*(nc_2,*(\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),g_1))),$ $\ominus(*(\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),*(inv(\pi'),*(\pi,*(nc_2,g_1))))))) \in xy$

- The proof for $m \in dom(xy)$ where m is a $\oplus$-object is reduced with the help of the regularity property 141-(1) to the cases given by the definitions of TCAMP_3msg_1, TCAMP_3mrg and TCAMP_3$oops$.

  For instance, the proof for the TCAMP_3mrg-case (using in particular Φ_8) is sketched in the following table:

$isObj^{\oplus}(m), m \in DY(\pi\bullet ik), (obj^*(m,m_1,m_2) \Rightarrow m_1 \notin DY(\pi\bullet ik)),$ $(obj^{\oplus}(m,m_1,m_2) \Rightarrow m_1 \notin DY(\pi\bullet ik)) \stackrel{?}{\vdash} m \in dom(xy)$
Property 141-(1) and TCAMP_3mrg yield: $\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),\oplus(*(nc_3,g_1),\ominus(*(pw,g_2))) \in ik, nc_1 \neq nc_3,$ $(nc_1,nc_3 \notin DY(\pi\bullet ik)), ((pw = \pi \wedge g_2 \notin DY(\pi\bullet ik)) \vee pw \notin DY(\pi\bullet ik)),$ $m = \oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))) \stackrel{?}{\vdash} m \in dom(xy)$ Confidentiality of π (by Ω) yields: $\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),\oplus(*(nc_3,g_1),\ominus(*(pw,g_2))) \in ik, nc_1 \neq nc_3,$ $(nc_1,nc_3,pw \notin DY(ik)), m = \oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))) \stackrel{\Phi_8}{\vdash} (m,m) \in xy$

- The proof for $h_2(m_1,m_2,m_3) \in dom(xy)$ with the help of the regularity properties in 143 is sketched in the following table:

$h_2(m_1,m_2,m_3) \in DY(\pi\bullet ik), \neg(m_1,m_2,m_3 \in DY(\pi\bullet ik)) \stackrel{?}{\vdash} h_2(m_1,m_2,m_3) \in dom(xy)$
Property 143-(3) yields: $h_2(m_1,m_2,m_3) \in DY(ik), \neg(m_1,m_2,m_3 \in DY(\pi\bullet ik)) \stackrel{?}{\vdash} h_2(m_1,m_2,m_3) \in dom(xy)$ Property 143-(4) yields: $h_2(m_1,m_2,m_3) \in ik, m_3 \notin DY(pw(i)\bullet ik) \stackrel{?}{\vdash} h_2(m_1,m_2,m_3) \in dom(xy)$ $DY(ik) \subseteq DY(pw(i)\bullet ik)$ yields: $h_2(m_1,m_2,m_3) \in ik, m_3 \notin DY(ik) \stackrel{\Phi_6}{\vdash} (h_2(m_1,m_2,m_3),h_2(m_1,m_2,m_3)) \in xy$

  The proofs for the cases $h_2(m_1,m_2,m_3) \in codom(xy)$, $h_1(m_1,m_2,m_3) \in dom(xy)$ and for $h_1(m_1,m_2,m_3) \in codom(xy)$ are similar.

The regularity properties (in 144) about fst- and snd-messages, which do not possess confidential fst- and respectively snd-parts, permit to handle the corresponding cases in the proof of Ψ^c by refutation as described for similar proof situations in Sec. 12.4.2.

14.4.3 Handling of Ψ^b:

The proof of Ψ^b consists in showing $xy \subset DY(\pi\bullet ik) \times DY(\pi'\bullet ik)$ and that the restriction of xy on its domain and codomain is bijective. The former ensues trivially from the definition of Φ. The latter is proven by showing the following conjectures:

1. For all $(m,m) \in xy$ and $(m_x,m_y) \in xy$ with $m_x \neq m_y$, we have $m_x, m_y \neq m$.

2. For all $(m_x, m_y), (m_z, m_u) \in xy$ with $m_x \neq m_y$ and $m_z \neq m_u$, we have $m_x \neq m_z \vee m_y = m_u$ and $m_y \neq m_u \vee m_x = m_z$.

We show these conjectures by case distinctions on the structure of messages that occur in the pairs of xy, according to Φ.

14.4.3.1 Pairs with Atomic Messages:

We start with the pairs that have atomic messages at the first or second position. These pairs are of the form (c_i, c_i) with $c_i \in DY(ik)$ or (π, π') with $\pi, \pi' \notin DY(ik)$ and $\pi \neq \pi'$, as ensued from the property Ω. This permits clearly to show conjectures (1) and (2) for all pairs of xy that map atomic messages. Furthermore, conjectures (1) and (2) hold for any pair of this first group and any other pair of xy, which maps some f-object to some g-object.

14.4.3.2 Pairs with $*$-Objects:

In the second part of our proof, we focus on the pairs of xy that have $*$-objects at the first or second position. According to Φ, the pairs belonging to this second group are of the following forms:

1. $(*(nc_1, g_1), *(nc_1, g_1))$, where nc_1 is from a 1-st message using $pw \in DY(ik)$.
2. $(*(nc_2, g_1), *(nc_2, g_1))$, where nc_2 is from a 2-nd message using $pw \in DY(ik)$.
3. $(*(pw, *(nc_2, g_1)), *(pw, *(nc_2, g_1)))$, where nc_2 is from a 2-nd message.
4. $(*(nc_2, *(nc_1, g_1)), *(nc_2, *(nc_1, g_1)))$, where nc_1 and nc_2 are from a 1-st and respectively a 2-nd message using $pw \in DY(ik)$.
5. $(*(m[g_2, \pi], *(\pi, *(nc_2, g_1))), *(m[g_2, \pi], *(\pi, *(nc_2, g_1))))$, where $m[g_2, \pi]$ abbreviates a 1-st message $\oplus(*(nc_1, g_1), \ominus(*(\pi, g_2)))$.
6. $(*(\pi, *(nc_2, *(nc_1, g_1))), \oplus(\oplus(*(\pi', *(nc_2, *(nc_1, g_1))), *(\pi', *(nc_2, *(m[g_2, \pi], g_1)))), \ominus(*(m[g_2, \pi], *(\pi, *(nc_2, g_1))))))$.
7. $(*(\pi, g_2), *(\pi', g_2))$.
8. $(*(inv(\pi), *(nc_1, g_1)), \oplus(\oplus(*(inv(\pi'), *(nc_1, g_1)), \ominus(*(inv(\pi'), *(\pi, g_2)))), g_2))$.
9. $(*(nc_1, g_1), \oplus(\oplus(*(nc_1, g_1), \ominus(*(\pi, g_2))), *(\pi', g_2)))$.
10. $(*(nc_2, g_1), *(inv(\pi'), *(\pi, *(nc_2, g_1))))$.
11. $(*(nc_2, *(nc_1, g_1)), \oplus(\oplus(*(nc_2, *(nc_1, g_1)), *(nc_2, *(m[g_2, \pi], g_1))), \ominus(*(m[g_2, \pi], *(inv(\pi'), *(\pi, *(nc_2, g_1))))))$.
12. $(*(m[g_2, \pi], *(nc_2, g_1)), *(m[g_2, \pi], *(inv(\pi'), *(\pi, *(nc_2, g_1)))))$.
13. $(*(inv(m[g_2, \pi]), *(nc_2, *(nc_1, g_1))), \oplus(\oplus(*(inv(m[g_2, \pi]), *(nc_2, *(nc_1, g_1))), *(nc_2, g_1)), \ominus(*(inv(\pi'), *(\pi, *(nc_2, g_1))))))$.
14. $(*(\pi, *(inv(m[g_2, \pi]), *(nc_2, *(nc_1, g_1)))), \oplus(\oplus(*(\pi', *(inv(m[g_2, \pi]), *(nc_2, *(nc_1, g_1)))), *(\pi', *(nc_2, g_1))), \ominus(*(\pi, *(nc_2, g_1)))))$.

We first describe the handling of the first proof task:

- For $(*(nc_1,g_1),*(nc_1,g_1))$ in (1), the required $*(nc_1,g_1) \neq m_x$ and $*(nc_1,g_1) \neq m_y$ hold for the pairs (m_x,m_y) in cases (6)–(8) and (11)–(14), as these messages have different structures.

  Regarding (m_x,m_y) in case (9), the structure of m_y differs from the structure of $*(nc_1,g_1)$ and the nonce in $m_x = *(nc_1,g_1)$ can be shown to differ from the nonce used in case (1), as the former is generated in a 1-st message using $\pi \notin DY(ik)$ and the latter in a 1-st message using $pw \in DY(ik)$.

  Regarding (m_x,m_y) in case (10), the structure of m_y differs from the structure of $*(nc_1,g_1)$ and the nonce in $m_x = *(nc_2,g_1)$ can be shown to differ from the nonce used in case (1), as the former is generated in a 2-nd message using $\pi \notin DY(ik)$ and the latter in a 1-st message using $pw \in DY(ik)$.

- For $(*(nc_2,g_1),*(nc_2,g_1))$ in (2), we use simlar proof arguments as in the previous case.

- For $(*(pw,*(nc_2,g_1)),*(pw,*(nc_2,g_1)))$ in (3), the required $*(pw,*(nc_2,g_1)) \neq m_x$ and $*(pw,*(nc_2,g_1)) \neq m_y$ hold for the pairs (m_x,m_y) in cases (6)–(14), as these messages have different structures.

- For $(*(nc_2,*(nc_1,g_1)),*(nc_2,*(nc_1,g_1)))$ in (4), the required $*(nc_2,*(nc_1,g_1)) \neq m_x$ and $*(nc_2,*(nc_1,g_1)) \neq m_y$ hold for the pairs (m_x,m_y) in cases (6)–(10) and (12)–(14), as these messages have different structures.

  Regarding (m_x,m_y) in case (11), the structure of m_y differs from the structure of $*(nc_2,*(nc_1,g_1))$ and the nonces in $m_x = *(nc_2,*(nc_1,g_1))$ can be shown to differ from the nonces used in case (4), as the former are generated in a TC-AMP run using $\pi \notin DY(ik)$ and the latter in a TC-AMP run using $pw \in DY(ik)$.

- For pairs
$$(*(m[g_2,\pi],*(\pi,*(nc_2,g_1))),*(m[g_2,\pi],*(\pi,*(nc_2,g_1))))$$
in (5), we have $*(m[g_2,\pi],*(\pi,*(nc_2,g_1))) \neq m_x$ and $*(m[g_2,\pi],*(\pi,*(nc_2,g_1))) \neq m_y$ for the pairs (m_x,m_y) in cases (6)–(14), as these messages have different structures.

The second proof task is handled as follows:

- For (m_x,m_y) in case (6) and the pairs (m_z,m_u) in cases (7)–(14), we have $m_x \neq m_z$ and $m_y \neq m_u$, as these messages have different structures.

- For $(*(\pi,g_2),*(\pi',g_2))$ in case (7) and the pairs (m_z,m_u) in cases (8)–(14), we have $*(\pi,g_2) \neq m_z$ and $*(\pi',g_2) \neq m_u$, as these messages have different structures.

- For (m_x,m_y) in case (8) and the pairs (m_z,m_u) in cases (9)–(14), we have $m_x \neq m_z$ and $m_y \neq m_u$, as these messages have different structures.

- For (m_x,m_y) in case (9) and the pairs (m_z,m_u) in cases (11)–(14), we have $m_x \neq m_z$ and $m_y \neq m_u$, as these messages have different structures.

  Regarding (m_z,m_u) in case (10), the structure of m_u differs from the structure of m_y and the nonce in $m_z = *(nc_2,g_1)$ can be shown to differ from the nonce used in case (9), as the former is generated in a 2-nd message while the latter is generated in a 1-st message.

- For (m_x,m_y) in case (10) and the pairs (m_z,m_u) in cases (11)–(14), we have $m_x \neq m_z$ and $m_y \neq m_u$, as these messages have different structures.

- For (m_x,m_y) in case (11) and the pairs (m_z,m_u) in cases (12)–(14), we have $m_x \neq m_z$ and $m_y \neq m_u$, as these messages have different structures.

- For (m_x, m_y) in case (12) and the pairs (m_z, m_u) in cases (13) and (14), we have $m_x \neq m_z$ and $m_y \neq m_u$, as these messages have different structures.
- For (m_x, m_y) in case (13) and the pair (m_z, m_u) in case (14), we have $m_x \neq m_z$ and $m_y \neq m_u$, as these messages have different structures.

14.4.3.3 Pairs with $\oplus$-Objects:

In the third part of our proof, we focus on the pairs of xy that have $\oplus$-objects at the first or second position. According to Φ, the pairs belonging to this second group are of the following forms:

1. $(\oplus(*(nc_1, g_1), \ominus(*(pw, g_2))), \oplus(*(nc_1, g_1), \ominus(*(pw, g_2))))$.
2. $(\oplus(*(nc_2, *(nc_1, g_1)), *(nc_2, *(m[g_2, pw], g_1))),$
   $\oplus(*(nc_2, *(nc_1, g_1)), *(nc_2, *(m[g_2, pw], g_1))))$.
3. $(\oplus(*(inv(m[g_2, pw]), *(nc_2, *(nc_1, g_1))), *(nc_2, g_1)),$
   $\oplus(*(inv(m[g_2, pw]), *(nc_2, *(nc_1, g_1))), *(nc_2, g_1)))$.
4. $(\oplus(*(nc_1, g_1), \ominus(*(nc_3, g_1))), \oplus(*(nc_1, g_1), \ominus(*(nc_3, g_1))))$.
5. $(*(\pi, *(nc_2, *(nc_1, g_1))), \oplus(\oplus(*(\pi', *(nc_2, *(nc_1, g_1))), *(\pi', *(nc_2, *(m[g_2, \pi], g_1)))),$
   $\ominus(*(m[g_2, \pi], *(\pi, *(nc_2, g_1))))))$.
6. $(*(inv(\pi), *(nc_1, g_1)), \oplus(\oplus(*(inv(\pi'), *(nc_1, g_1)), \ominus(*(inv(\pi'), *(\pi, g_2)))), g_2))$.
7. $(*(nc_1, g_1), \oplus(\oplus(*(nc_1, g_1), \ominus(*(\pi, g_2))), *(\pi', g_2)))$.
8. $(*(nc_2, *(nc_1, g_1)), \oplus(\oplus(*(nc_2, *(nc_1, g_1)), *(nc_2, *(m[g_2, \pi], g_1))),$
   $\ominus(*(m[g_2, \pi], *(inv(\pi'), *(\pi, *(nc_2, g_1))))))$.
9. $(*(inv(m[g_2, \pi]), *(nc_2, *(nc_1, g_1))),$
   $\oplus(\oplus(*(inv(m[g_2, \pi]), *(nc_2, *(nc_1, g_1))), *(nc_2, g_1)), \ominus(*(inv(\pi'), *(\pi, *(nc_2, g_1))))))$.
10. $(*(\pi, *(inv(m[g_2, \pi]), *(nc_2, *(nc_1, g_1)))),$
    $\oplus(\oplus(*(\pi', *(inv(m[g_2, \pi]), *(nc_2, *(nc_1, g_1)))), *(\pi', *(nc_2, g_1))),$
    $\ominus(*(\pi, *(nc_2, g_1)))))$.
11. $(\oplus(*(inv(\pi), *(nc_1, g_1)), \ominus(g_2)), \oplus(*(inv(\pi'), *(nc_1, g_1)), \ominus(*(inv(\pi'), *(\pi, g_2)))))$.
12. $(\oplus(*(\pi, *(nc_2, *(nc_1, g_1))), *(m[g_2, \pi], *(\pi, *(nc_2, g_1)))),$
    $\oplus(*(\pi', *(nc_2, *(nc_1, g_1))), *(\pi', *(nc_2, *(m[g_2, \pi], g_1)))))$.
13. $(\oplus(*(\pi, *(inv(m[g_2, \pi]), *(nc_2, *(nc_1, g_1)))), *(\pi, *(nc_2, g_1)))$
    $\oplus(*(\pi', *(inv(m[g_2, \pi]), *(nc_2, *(nc_1, g_1)))), *(\pi', *(nc_2, g_1))))$.
14. $(\oplus(*(inv(\pi), *(nc_1, g_1)), \ominus(*(inv(\pi), *(nc_3, g_1)))),$
    $\oplus(*(inv(\pi'), *(nc_1, g_1)), \ominus(*(inv(\pi'), *(nc_3, g_1)))))$.

Note that cases (5)–(10) are already considered in Sec. 14.4.3.2, as they have $*$-objects at the first position.

We first describe the handling of the first proof task:

- For pairs
  $$(\oplus(*(nc_1, g_1), \ominus(*(pw, g_2))), \oplus(*(nc_1, g_1), \ominus(*(pw, g_2))))$$
  in (1), we have $\oplus(*(nc_1, g_1), \ominus(*(pw, g_2))) \neq m_x$ and $\oplus(*(nc_1, g_1), \ominus(*(pw, g_2))) \neq m_y$ for the pairs (m_x, m_y) in cases (5)–(14), as these messages have different structures.

- For (m,m) in (2), where m equals $\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[g_2,pw],g_1)))$, the required $m \neq m_x$ and $m \neq m_y$ hold for the pairs (m_x,m_y) in cases (5)–(14), as these messages have different structures.
- For (m,m) in (3), where m equals $\oplus(*(inv(m[g_2,pw]),*(nc_2,*(nc_1,g_1))),*(nc_2,g_1))$, the required $m \neq m_x$ and $m \neq m_y$ hold for the pairs (m_x,m_y) in cases (5)–(14), as these messages have different structures.
- For pairs
$$(\oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))),\oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))))$$
in (4), we have $\oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))) \neq m_x$ and $\oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))) \neq m_y$ for the pairs (m_x,m_y) in cases (5)–(14), as these messages have different structures.

The second proof task is handled as follows:

- The pairs (m_x,m_y) in cases (5)–(10) are already shown in Sec. 14.4.3.2 to mutually fulfill the second proof task. Furthermore, they have $*$-objects as first message m_x and $\oplus$-objects with *three* basic $\oplus$-parts as second message m_y. Since the pairs (m_z,m_u) in cases (11)–(14) have $\oplus$-objects with *two* basic $\oplus$-parts as first message m_z and as second message m_u, we have $m_x \neq m_z$ and $m_y \neq m_u$. That is, we may focus on cases (11)–(14), only.
- For (m_x,m_y) in case (11) and the pairs (m_z,m_u) in cases (12)–(14), we have $m_x \neq m_z$ and $m_y \neq m_u$, as these messages have different structures.
- For (m_x,m_y) in case (12) and the pairs (m_z,m_u) in cases (13) and (14), we have $m_x \neq m_z$ and $m_y \neq m_u$, as these messages have different structures.
- For (m_x,m_y) in case (13) and the pair (m_z,m_u) in case (14), we have $m_x \neq m_z$ and $m_y \neq m_u$, as these messages have different structures.

14.4.3.4 Remaining Pairs:

In the rest of our proof, we focus on the remaining pairs of xy according to Φ, which are of the following forms:

1. $(h_1(m_1,*(pw,*(nc_2,g_1)),*(nc_2,\oplus(m_1,\oplus(*(pw,g_2),*(m_1,g_1))))),$
$h_1(m_1,*(pw,*(nc_2,g_1)),*(nc_2,\oplus(m_1,\oplus(*(pw,g_2),*(m_1,g_1))))))$.
2. $(h_2(\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),m_2,\oplus(*(inv(pw),*(nc_1,m_2)),$
$*(inv(pw),*(\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),m_2)))),$
$h_2(\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),m_2,\oplus(*(inv(pw),*(nc_1,m_2)),$
$*(inv(pw),*(\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),m_2)))))$.
3. $(inv(m[g_2,pw]),inv(m[g_2,pw]))$.
4. $(inv(\pi),inv(\pi'))$.

The required dis-equalities for the first and the second proof task are shown trivially, as the compared messages have different structures.

14.4.4 Handling of Ψ^1_* and Ψ^2_*:

The remaining conditions of Ψ are inclusion rules where the premises require a composed message in the domain or the codomain of xy that is derivable by composition. According to Φ, this requirement holds only for $*$-, $\oplus$-, $\ominus$-, inv-, h_1- and h_2-objects. In this section, we consider the pairs that include $*$- or $\oplus$-objects that are relevant for the proof of Ψ^1_* and Ψ^2_*.

We first describe the handling of Ψ^1_*. Since the relevant pairs (m_x, m_y) for Ψ^1_* satisfy $isSyn^*(m_x)$, we may focus on the cases (1)–(14) in Sec. 14.4.3.2 and the cases (2), (3) and (12)–(14) in Sec. 14.4.3.3. In cases (1), (2), (4), and (9)–(11) from Sec. 14.4.3.2 and in cases (2) and (3) from Sec. 14.4.3.3, all left (common) $*$-sub-messages of m_x are confidential. This permits to show Ψ^1_* by refuting its assumption. For the remaining cases, we have to provide for every left $*$-sub-message m_{x1} of m_x that belongs to $DY(\pi.ik)$ the corresponding pairs required for fulfilling Ψ^1_*. We first handle the cases from Sec. 14.4.3.2.

- For $(*(pw,*(nc_2,g_1)),*(pw,*(nc_2,g_1)))$ in case (3), we have $nc_2, pw \notin DY(ik)$. This implies $nc_2 \notin DY(\pi.ik)$ and $pw \in DY(\pi.ik) \Rightarrow pw = \pi$. That is, we need to identify further included pairs only when $pw = \pi$. In this case, the pair (π, π') and case (10) from Sec. 14.4.3.2 permit to show the conclusion of Ψ^1_*.

- For pais
$$(*(m[g_2,\pi],*(\pi,*(nc_2,g_1))),*(m[g_2,\pi],*(\pi,*(nc_2,g_1))))$$
in case (5), where $m[g_2,\pi]$ abbreviates a 1-st message $\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2)))$, we get $nc_2 \notin DY(\pi.ik)$. Here, we have $m[g_2,\pi], \pi \in DY(\pi.ik)$. For the $*$-sub-message $m[g_2,\pi]$, case (1) from Sec. 14.4.3.3 and case (3) from Sec. 14.4.3.2 permit to show the conclusion of Ψ^1_*; For the $*$-sub-message π, the pair (π, π') and case (12) from Sec. 14.4.3.2 are used to show the conclusion of Ψ^1_*.

- $(*(\pi,*(nc_2,*(nc_1,g_1))),\oplus(\oplus(*(\pi',*(nc_2,*(nc_1,g_1))),*(\pi',*(nc_2,*(m[g_2,\pi],g_1)))),$ $\ominus(*(m[g_2,\pi],*(\pi,*(nc_2,g_1))))))$ is the pair of case (6), where $nc_2, nc_1 \notin DY(\pi.ik)$ holds. For the non-confidential $*$-sub-message π, the pair (π, π') and case (11) from Sec. 14.4.3.2 permit to show the conclusion of Ψ^1_*.

- For $(*(\pi,g_2),*(\pi',g_2))$ in case (7), we have $g_2 \in DY(ik)$. That is, (g_2,g_2) is in xy and this together with the pair (π, π') permit to show the conclusion of Ψ^1_*.

- For pairs
$$(*(inv(\pi),*(nc_1,g_1)),\oplus(\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(\pi,g_2)))),g_2))$$
in case (8), we have $nc_1 \notin DY(\pi.ik)$. Case (4) from Sec. 14.4.3.4 and case (9) from Sec. 14.4.3.2 permit to show the conclusion of Ψ^1_*.

- For $(*(m[g_2,\pi],*(nc_2,g_1)),*(m[g_2,\pi],*(inv(\pi'),*(\pi,*(nc_2,g_1)))))$ in case (12), we have $nc_2 \notin DY(\pi.ik)$. Case (1) from Sec. 14.4.3.3 and case (10) from Sec. 14.4.3.2 permit to show the conclusion of Ψ^1_*.

- For pairs
$$\begin{array}{l}(*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1))),\\ \oplus(\oplus(*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1))),*(nc_2,g_1)),\\ \quad\ominus(*(inv(\pi'),*(\pi,*(nc_2,g_1))))))\end{array}$$
in case (13), we have $nc_2, nc_1 \notin DY(\pi.ik)$. Case (3) from Sec. 14.4.3.4 and case (11) from Sec. 14.4.3.2 permit to show the conclusion of Ψ^1_*.

- For pairs
$$\begin{array}{l}(*(\pi,*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),\\ \oplus(\oplus(*(\pi',*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),*(\pi',*(nc_2,g_1))),\\ \quad\ominus(*(\pi,*(nc_2,g_1)))))\end{array}$$
in case (14), we have $nc_2, nc_1 \notin DY(\pi.ik)$. Focusing on π, the pair (π, π') and case (13) from Sec. 14.4.3.2 permit to show the conclusion of Ψ^1_*. For $inv(m[g_2,\pi])$, case (3) from Sec. 14.4.3.4 and case (6) from Sec. 14.4.3.2 are used to show the conclusion of Ψ^1_*.

Next, we handle the cases from Sec. 14.4.3.3.

- For pairs

$$(\oplus(*(\pi,*(nc_2,*(nc_1,g_1))),*(m[g_2,\pi],*(\pi,*(nc_2,g_1)))),$$
$$\oplus(*(\pi',*(nc_2,*(nc_1,g_1))),*(\pi',*(nc_2,*(m[g_2,\pi],g_1)))))$$

  in case (12), we have $nc_2 \notin DY(\pi.ik)$. The pair (π,π') and case (2) from Sec. 14.4.3.3 permit to show the conclusion of Ψ^1_*.

- $(\oplus(*(\pi,*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),*(\pi,*(nc_2,g_1))) \oplus(*(\pi',*(inv(m[g_2,\pi]),$ $*(nc_2,*(nc_1,g_1)))),*(\pi',*(nc_2,g_1))))$ in case (13) implies $nc_2 \notin DY(\pi.ik)$. The pair (π,π') and case (3) from Sec. 14.4.3.3 permit to show the conclusion of Ψ^1_*.

- For pairs

$$(\oplus(*(inv(\pi),*(nc_1,g_1)),\ominus(*(inv(\pi),*(nc_3,g_1)))),$$
$$\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(nc_3,g_1)))))$$

  in case (14), we have $nc_1 \neq nc_3$. Case (4) from Sec. 14.4.3.4 and case (4) from Sec. 14.4.3.3 permit to show the conclusion of Ψ^1_*.

Similarly, the relevant pairs (m_x,m_y) for Ψ^2_* satisfy $isSyn^*(m_y)$. So, we may focus on the cases (1)–(7) and (10)–(14) in Sec. 14.4.3.2 and the cases (2), (3) and (11)–(14) in Sec. 14.4.3.3. In cases (1)–(4), (6), (11), (13) and (14) from Sec. 14.4.3.2 and in cases (2) and (3) from Sec. 14.4.3.3, all left (common) $*$-sub-messages of m_y are confidential. This permits to show Ψ^2_* by refuting its assumption. For the remaining cases, we have to provide for every left $*$-sub-message m_{y1} of m_y that belongs to $DY(\pi'.ik)$ the corresponding pairs required for fulfilling Ψ^2_*. We first handle the cases from Sec. 14.4.3.2.

- $(*(m[g_2,\pi],*(\pi,*(nc_2,g_1))),*(m[g_2,\pi],*(\pi,*(nc_2,g_1))))$ in case (5), where $m[g_2,\pi]$ abbreviates a 1-st message $\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2)))$, implies $nc_2,\pi \notin DY(\pi'.ik)$. Case (1) from Sec. 14.4.3.3 and case (3) from Sec. 14.4.3.2 permit to show the conclusion of Ψ^2_*.

- For $(*(\pi,g_2),*(\pi',g_2))$ in case (7), the handling of Ψ^2_* is similar to Ψ^1_* (see above).

- For $(*(nc_2,g_1),*(inv(\pi'),*(\pi,*(nc_2,g_1))))$ in case (10), we have $nc_2,\pi \notin DY(\pi'.ik)$. Case (4) from Sec. 14.4.3.4 and case (3) from Sec. 14.4.3.3 permit to show the conclusion of Ψ^2_*.

- $(*(m[g_2,\pi],*(nc_2,g_1)),*(m[g_2,\pi],*(inv(\pi'),*(\pi,*(nc_2,g_1)))))$ in case (12) implies $nc_2,\pi \notin DY(\pi'.ik)$. For $m[g_2,\pi]$, case (1) from Sec. 14.4.3.3 and case (10) from Sec. 14.4.3.2 permit to show the conclusion of Ψ^2_*. For $inv(\pi')$, Case (4) from Sec. 14.4.3.4 and case (5) from Sec. 14.4.3.3 are used to show the conclusion of Ψ^2_*.

Next, we handle the cases from Sec. 14.4.3.3.

- The pair

$$(\oplus(*(inv(\pi),*(nc_1,g_1)),\ominus(g_2)),\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(\pi,g_2)))))$$

  in case (11) is handled with case (4) from Sec. 14.4.3.4 and case (1) from Sec. 14.4.3.3, which permit to show the conclusion of Ψ^2_*.

- For pairs

$$(\oplus(*(\pi,*(nc_2,*(nc_1,g_1))),*(m[g_2,\pi],*(\pi,*(nc_2,g_1)))),$$
$$\oplus(*(\pi',*(nc_2,*(nc_1,g_1))),*(\pi',*(nc_2,*(m[g_2,\pi],g_1)))))$$

  in case (12), we have $nc_2 \notin DY(\pi'.ik)$. The pair (π,π') and case (2) from Sec. 14.4.3.3 permit to show the conclusion of Ψ^2_*.

- $(\oplus(*(\pi,*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),*(\pi,*(nc_2,g_1))) \oplus(*(\pi',*(inv(m[g_2,\pi]),$ $*(nc_2,*(nc_1,g_1)))), *(\pi',*(nc_2,g_1))))$ in case (13) implies $nc_2 \notin DY(\pi'.ik)$. The pair (π,π') and case (3) from Sec. 14.4.3.3 permit to show the conclusion of Ψ^2_*.

- For pairs

$$(\oplus(*(inv(\pi),*(nc_1,g_1)),\ominus(*(inv(\pi),*(nc_3,g_1)))),$$
$$\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(nc_3,g_1)))))$$

  in case (14), we use case (4) from Sec. 14.4.3.4 and case (4) from Sec. 14.4.3.3 to show the conclusion of Ψ^2_*.

14.4.5 Handling of $\Psi^1_{*,\oplus}$ and $\Psi^2_{*,\oplus}$:

In this section, we consider the pairs in xy that include $\oplus$-objects relevant for the proof of $\Psi^1_{*,\oplus}$ and $\Psi^2_{*,\oplus}$.

We first describe the handling of $\Psi^1_{*,\oplus}$. Since the relevant pairs (m_x,m_y) for $\Psi^1_{*,\oplus}$ include $\oplus$-objects m_x with non-common left $*$-parts, we may focus on the cases (1)–(4) and (11)–(14) in Sec. 14.4.3.3. In cases (4) and (14) from Sec. 14.4.3.3, all left non-common $*$-parts of m_x are confidential. This permits to show $\Psi^1_{*,\oplus}$ by refuting its assumption. For the remaining cases, we have to provide for every non-common left $*$-part m_{x1} of m_x that belongs to $DY(\pi.ik)$ the corresponding pairs required for fulfilling $\Psi^1_{*,\oplus}$.

- For $(\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))))$ in case (1), we have $nc_1 \notin DY(\pi.ik)$ and $pw \in DY(\pi.ik) \Rightarrow pw = \pi$. For $\pi \in DY(\pi.ik)$, the pair (π,π') and case (11) from Sec. 14.4.3.3 are used to show the conclusion of $\Psi^1_{*,\oplus}$.

- For pairs

$$(\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[g_2,pw],g_1))),$$
$$\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[g_2,pw],g_1))))$$

  in case (2), we have $nc_1 \notin DY(\pi.ik)$. Considering $m[g_2,pw] \in DY(\pi.ik)$, cases (1) and (3) from Sec. 14.4.3.3 are used to show the conclusion of $\Psi^1_{*,\oplus}$.

- For pairs

$$(\oplus(*(inv(m[g_2,pw]),*(nc_2,*(nc_1,g_1))),*(nc_2,g_1)),$$
$$\oplus(*(inv(m[g_2,pw]),*(nc_2,*(nc_1,g_1))),*(nc_2,g_1)))$$

  in case (3), we have $nc_1 \notin DY(\pi.ik)$. Considering $inv(m[g_2,pw]) \in DY(\pi.ik)$, case (3) from Sec. 14.4.3.4 and case (2) from Sec. 14.4.3.3 are used to show the conclusion of $\Psi^1_{*,\oplus}$.

- For pairs

$$(\oplus(*(inv(\pi),*(nc_1,g_1)),\ominus(g_2)),\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(\pi,g_2)))))$$

  in case (11), we have $nc_1 \notin DY(\pi.ik)$. For $inv(\pi) \in DY(\pi.ik)$, case (4) from Sec. 14.4.3.4 and case (1) from Sec. 14.4.3.3 are used to show the conclusion of $\Psi^1_{*,\oplus}$.

- For pairs

$$(\oplus(*(\pi,*(nc_2,*(nc_1,g_1))),*(m[g_2,\pi],*(\pi,*(nc_2,g_1)))),$$
$$\oplus(*(\pi',*(nc_2,*(nc_1,g_1))),*(\pi',*(nc_2,*(m[g_2,\pi],g_1)))))$$

  in case (12), we have $nc_1 \notin DY(\pi\bullet ik)$. For $m[g_2,pw] \in DY(\pi\bullet ik)$, cases (1) and (13) from Sec. 14.4.3.3 are used to show the conclusion of $\Psi^1_{*,\oplus}$.

- For pairs

$$(\oplus(*(\pi,*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),*(\pi,*(nc_2,g_1))),$$
$$\oplus(*(\pi',*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),*(\pi',*(nc_2,g_1))))$$

  in case (13), we have $nc_1 \notin DY(\pi\bullet ik)$. For $inv(m[g_2,pw]) \in DY(\pi\bullet ik)$, case (3) from Sec. 14.4.3.4 and case (12) from Sec. 14.4.3.3 are used to show the conclusion of $\Psi^1_{*,\oplus}$.

Similarly, the relevant pairs (m_x,m_y) for $\Psi^2_{*,\oplus}$ include $\oplus$-objects m_y with non-common left $*$-parts. So, we may focus on the cases (1)–(14) in Sec. 14.4.3.3. In cases (1), (4), (11) and (14) from Sec. 14.4.3.3, all left non-common $*$-parts of m_y are confidential. This permits to show $\Psi^2_{*,\oplus}$ by refuting its assumption. For the remaining cases, we have to provide for every non-common left $*$-part m_{y1} of m_y that belongs to $DY(\pi'\bullet ik)$ the corresponding pairs required for fulfilling $\Psi^2_{*,\oplus}$.

- Cases (2), (3), (12) and (13) are handled as described for the proof of $\Psi^1_{*,\oplus}$.

- For pairs

$$(*(\pi,*(nc_2,*(nc_1,g_1))),$$
$$\oplus(\oplus(*(\pi',*(nc_2,*(nc_1,g_1))),*(\pi',*(nc_2,*(m[g_2,\pi],g_1)))),$$
$$\ominus(*(m[g_2,\pi],*(\pi,*(nc_2,g_1))))))$$

  in case (5), we have $nc_1,\pi \notin DY(\pi\bullet ik)$. For $m[g_2,pw] \in DY(\pi'\bullet ik)$, cases (1) and (10) from Sec. 14.4.3.3 are used to show the conclusion of $\Psi^2_{*,\oplus}$. For $\pi' \in DY(\pi'\bullet ik)$, the pair (π,π') and case (8) from Sec. 14.4.3.3 permit to show the conclusion of $\Psi^2_{*,\oplus}$.

- For pairs

$$(*(inv(\pi),*(nc_1,g_1)),\oplus(\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(\pi,g_2)))),g_2))$$

  in case (6), we have $nc_1,\pi \notin DY(\pi\bullet ik)$. For $inv(\pi') \in DY(\pi'\bullet ik)$, case (4) from Sec. 14.4.3.4 and case (7) from Sec. 14.4.3.3 permit to show the conclusion of $\Psi^2_{*,\oplus}$.

- For pairs $(*(nc_1,g_1),\oplus(\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2))),*(\pi',g_2)))$ in case (7), we have $nc_1,\pi \notin DY(\pi\bullet ik)$. Considering $\pi' \in DY(\pi'\bullet ik)$, the pair (π,π') and case (6) from Sec. 14.4.3.3 permit to show the conclusion of $\Psi^2_{*,\oplus}$. .

- For pairs

$$(*(nc_2,*(nc_1,g_1)),$$
$$\oplus(\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[g_2,\pi],g_1))),$$
$$\ominus(*(m[g_2,\pi],*(inv(\pi'),*(\pi,*(nc_2,g_1)))))))$$

  in case (8), we have $nc_1,\pi \notin DY(\pi\bullet ik)$. Considering $m[g_2,pw] \in DY(\pi'\bullet ik)$, cases (1) and (9) from Sec. 14.4.3.3 are used to show the conclusion of $\Psi^2_{*,\oplus}$. For $inv(\pi') \in DY(\pi'\bullet ik)$, case (4) from Sec. 14.4.3.4 and case (5) from Sec. 14.4.3.3 permit to show the conclusion of $\Psi^2_{*,\oplus}$.

- For pairs

$$(*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1))),\\ \oplus(\oplus(*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1))),*(nc_2,g_1)),\\ \ominus(*(inv(\pi'),*(\pi,*(nc_2,g_1))))))$$

in case (9), we have $nc_1, \pi \notin DY(\pi\bullet ik)$. For $inv(m[g_2,pw]) \in DY(\pi'\bullet ik)$, case (3) from Sec. 14.4.3.4 and case (8) from Sec. 14.4.3.3 are used to show the conclusion of $\Psi^2_{*,\oplus}$. For $inv(\pi') \in DY(\pi'\bullet ik)$, case (4) from Sec. 14.4.3.4 and case (10) from Sec. 14.4.3.3 permit to show the conclusion of $\Psi^2_{*,\oplus}$.

- For pairs

$$(*(\pi,*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),\\ \oplus(\oplus(*(\pi',*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),*(\pi',*(nc_2,g_1))),\\ \ominus(*(\pi,*(nc_2,g_1)))))$$

in case (10), we have $nc_1, \pi \notin DY(\pi\bullet ik)$. Considering $inv(m[g_2,pw]) \in DY(\pi'\bullet ik)$, case (3) from Sec. 14.4.3.4 and case (5) from Sec. 14.4.3.3 are used to show the conclusion of $\Psi^2_{*,\oplus}$. For $\pi' \in DY(\pi'\bullet ik)$, the pair (π,π') and case (9) from Sec. 14.4.3.3 permit to show the conclusion of $\Psi^2_{*,\oplus}$.

14.4.6 Handling of $\Psi^1_\oplus$ and $\Psi^2_\oplus$:

In this section, we consider the pairs in xy that include $\oplus$-objects relevant for the proof of $\Psi^1_\oplus$ and $\Psi^2_\oplus$.

We first describe the handling of $\Psi^1_\oplus$. Since the relevant pairs (m_x, m_y) for $\Psi^1_\oplus$ include $\oplus$-objects m_x, we may focus on the cases (1)–(4) and (11)–(14) in Sec. 14.4.3.3. In cases (1), (4), (11) and (14) from Sec. 14.4.3.3, all $\oplus$-parts of m_x are confidential. This permits to show $\Psi^1_\oplus$ by refuting its assumption. For the remaining cases, we have to provide for every $\oplus$-part m_{x1} of m_x that belongs to $DY(\pi\bullet ik)$ the corresponding pairs required for fulfilling $\Psi^1_\oplus$.

- For pairs

$$(\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[g_2,pw],g_1))),\\ \oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[g_2,pw],g_1))))$$

in case (2), the $\oplus$-parts of the 1-st message are in $DY(\pi\bullet ik)$ only if $pw = \pi$. Here, cases (12) and (11) from Sec. 14.4.3.2 permit to show the conclusion of $\Psi^1_\oplus$.

- For pairs

$$(\oplus(*(inv(m[g_2,pw]),*(nc_2,*(nc_1,g_1))),*(nc_2,g_1)),\\ \oplus(*(inv(m[g_2,pw]),*(nc_2,*(nc_1,g_1))),*(nc_2,g_1)))$$

in case (3), the $\oplus$-parts of the 1-st message are in $DY(\pi\bullet ik)$ only if $pw = \pi$. Here, cases (10) and (13) from Sec. 14.4.3.2 permit to show the conclusion of $\Psi^1_\oplus$.

- For pairs

$$(\oplus(*(\pi,*(nc_2,*(nc_1,g_1))),*(m[g_2,\pi],*(\pi,*(nc_2,g_1)))),\\ \oplus(*(\pi',*(nc_2,*(nc_1,g_1))),*(\pi',*(nc_2,*(m[g_2,\pi],g_1)))))$$

in case (12), the $\oplus$-parts of the 1-st message are in $DY(\pi\bullet ik)$. Here, cases (5) and (6) from Sec. 14.4.3.2 permit to show the conclusion of $\Psi^1_\oplus$.

- For pairs

$$(\oplus(*(\pi,*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),*(\pi,*(nc_2,g_1))),$$
$$\oplus(*(\pi',*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),*(\pi',*(nc_2,g_1))))$$

in case (13), the $\oplus$-parts of the 1-st message are in $DY(\pi\bullet ik)$. Here, cases (3) and (14) from Sec. 14.4.3.2 permit to show the conclusion of $\Psi^1_\oplus$.

Similarly, the relevant pairs (m_x,m_y) for $\Psi^2_\oplus$ include $\oplus$-objects m_y. So, we may focus on the cases (1)–(14) in Sec. 14.4.3.3. In cases (1)–(4) and (11)–(14) from Sec. 14.4.3.3, all $\oplus$-parts of m_y are confidential. This permits to show $\Psi^2_\oplus$ by refuting its assumption. For the remaining cases, we have to provide for every $\oplus$-part m_{y1} of m_x that belongs to $DY(\pi'\bullet ik)$ the corresponding pairs required for fulfilling $\Psi^2_\oplus$.

- For the pairs

$$(*(\pi,*(nc_2,*(nc_1,g_1))),$$
$$\oplus(\oplus(*(\pi',*(nc_2,*(nc_1,g_1))),*(\pi',*(nc_2,*(m[g_2,\pi],g_1)))),$$
$$\ominus(*(m[g_2,\pi],*(\pi,*(nc_2,g_1))))))$$

in case (5), the 2-nd message has one pair of $\oplus$-parts in $DY(\pi'\bullet ik)$. Here, case (5) from Sec. 14.4.3.2 and case (12) from Sec. 14.4.3.3 permit to show the conclusion of $\Psi^2_\oplus$.

- For the pairs

$$(*(inv(\pi),*(nc_1,g_1)),\oplus(\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(\pi,g_2)))),g_2))$$

in case (6), the 2-nd message has one pair of $\oplus$-parts in $DY(\pi'\bullet ik)$. Here, the pair (g_2,g_2) and case (11) from Sec. 14.4.3.3 permit to show the conclusion of $\Psi^2_\oplus$.

- For $(*(nc_1,g_1),\oplus(\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2))),*(\pi',g_2)))$ in case (7), the 2-nd message has one pair of $\oplus$-parts in $DY(\pi'\bullet ik)$. Here, case (7) from Sec. 14.4.3.2 and case (1) from Sec. 14.4.3.3 permit to show the conclusion of $\Psi^2_\oplus$.

- For the pairs

$$(*(nc_2,*(nc_1,g_1)),$$
$$\oplus(\oplus(*(nc_2,*(nc_1,g_1)),*(nc_2,*(m[g_2,\pi],g_1))),$$
$$\ominus(*(m[g_2,\pi],*(inv(\pi'),*(\pi,*(nc_2,g_1)))))))$$

in case (8), the 2-nd message has one pair of $\oplus$-parts in $DY(\pi'\bullet ik)$. Here, case (12) from Sec. 14.4.3.2 and case (2) from Sec. 14.4.3.3 permit to show the conclusion of $\Psi^2_\oplus$.

- For the pairs

$$(*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1))),$$
$$\oplus(\oplus(*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1))),*(nc_2,g_1)),$$
$$\ominus(*(inv(\pi'),*(\pi,*(nc_2,g_1))))))$$

in case (9), the 2-nd message has one pair of $\oplus$-parts in $DY(\pi'\bullet ik)$. Here, case (10) from Sec. 14.4.3.2 and case (3) from Sec. 14.4.3.3 permit to show the conclusion of $\Psi^2_\oplus$.

- For the pairs

$$(*(\pi,*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1)))),$$
$$\oplus(\oplus(*(\pi',*(inv(m[g_2,\pi]),*(nc_2,*(nc_1,g_1))))),$$
$$*(\pi',*(nc_2,g_1))),\ominus(*(\pi,*(nc_2,g_1)))))$$

in case (10), the 2-nd message has one pair of $\oplus$-parts in $DY(\pi'\bullet ik)$. Here, case (3) from Sec. 14.4.3.2 and case (13) from Sec. 14.4.3.3 permit to show the conclusion of $\Psi^2_\oplus$.

14.4.7 Handling of $\Psi^1_{\oplus,mrg}$ and $\Psi^2_{\oplus,mrg}$:

In this section, we consider the pairs in xy that include $\oplus$-objects relevant for the proof of $\Psi^1_{\oplus,mrg}$ and $\Psi^2_{\oplus,mrg}$.

The relevant couples of pairs (m_x, m_y) and (m_z, m_u) for $\Psi^1_{\oplus,mrg}$ (resp. $\Psi^2_{\oplus,mrg}$) include $\oplus$-objects m_x and m_z (resp. m_y and m_u) with common $\oplus$-parts, modulo application of $\ominus$. Thus, we may focus on the cases (1)–(14) in Sec. 14.4.3.3. In cases (2), (3), (5), (8)–(10), (12) and (13), the 1-st and 2-nd messages do not have common $\oplus$-parts with other 1-st respectively 2-nd messages in other (different) cases. This permits to show $\Psi^1_{\oplus,mrg}$ and $\Psi^2_{\oplus,mrg}$ by refuting its assumption. For the remaining cases, we explain below for every $\oplus$-part of a 1-st or a 2-nd message that occurs in several cases how the corresponding pairs required for fulfilling $\Psi^1_{\oplus,mrg}$ and/ or $\Psi^2_{\oplus,mrg}$ are obtained.

- For $(\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))))$ in case (1), message $*(nc_1,g_1)$ occurs in a first TC-AMP message either with $pw \notin DY(\pi\bullet ik)$ or with $pw = \pi$ and $g_2 \notin DY(ik)$. Such $*(nc_1,g_1)$ occurs again in pairs

  $$(\oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))),\oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))))$$

  from case (4). It does not fit with the occurrence of similar $\oplus$-parts in pairs of case (7), as these are used in first TC-AMP messages with $g_2 \in DY(ik)$. Here, case (1) itself implies the required pair

  $$(\oplus(*(nc_3,g_1),\ominus(*(pw,g_2))),\oplus(*(nc_3,g_1),\ominus(*(pw,g_2))))$$

  to fulfill $\Psi^1_{\oplus,mrg}$ and $\Psi^2_{\oplus,mrg}$.

  In the same context, $*(pw,g_2)$ occurs again in pairs

  $$(\oplus(*(nc_3,g_1),\ominus(*(pw,g_2))),\oplus(*(nc_3,g_1),\ominus(*(pw,g_2))))$$

  from case (1), (but not in pairs from case (7)). Here, case (4) implies the required pair

  $$(\oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))),\oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))))$$

  to fulfill $\Psi^1_{\oplus,mrg}$ and $\Psi^2_{\oplus,mrg}$.

- For $(\oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))),\oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))))$ in case (4), $*(nc_1,g_1)$ occurs in a first TC-AMP message fulfilling the conditions either of case (1) or of case (7). The former case is handled above. For the latter case, case (7) implies the required pair $(*(nc_3,g_1),\oplus(\oplus(*(nc_3,g_1),\ominus(*(\pi,g_2))),*(\pi',g_2)))$ to fulfill $\Psi^2_{\oplus,mrg}$.

  In the same context, $\oplus(\ominus(*(\pi,g_2)),*(\pi',g_2))$ from pairs

  $$(*(nc_1,g_1),\oplus(\oplus(*(nc_1,g_1),\ominus(*(\pi,g_2))),*(\pi',g_2)))$$

  in case (7) occurs again in other pairs of the same case together with $*(nc_3,g_1)$. Here, case (4) implies the required pair

  $$(\oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))),\oplus(*(nc_1,g_1),\ominus(*(nc_3,g_1))))$$

  to fulfill $\Psi^2_{\oplus,mrg}$.

- For $(*(inv(\pi),*(nc_1,g_1)),\oplus(\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(\pi,g_2)))),g_2))$ in case (6), g_2 is in $DY(ik)$. Thus, $*(inv(\pi'),*(nc_1,g_1))$ does not fit with similar $\oplus$-parts in case (11), where g_2 is confidential. It occurs again only in pairs

  $$(\oplus(*(inv(\pi),*(nc_1,g_1)),\ominus(*(inv(\pi),*(nc_3,g_1)))),$$
  $$\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(nc_3,g_1)))))$$

from case (14). Here, case (6) itself implies the required pair

$$(*(inv(\pi),*(nc_3,g_1)),\oplus(\oplus(*(inv(\pi'),*(nc_3,g_1)),\ominus(*(inv(\pi'),*(\pi,g_2)))),g_2))$$

to fulfill $\Psi^2_{\oplus,mrg}$.

In the same context, $\oplus(\ominus(*(inv(\pi'),*(\pi,g_2))),g_2)$ from case (6) can occur in two different pairs, e.g., with $*(inv(\pi'),*(nc_1,g_1))$ and with $*(inv(\pi'),*(nc_3,g_1))$. Here, case (14) implies the required pair $(\oplus(*(inv(\pi),*(nc_1,g_1)),\ominus(*(inv(\pi),*(nc_3,g_1)))),$ $\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(nc_3,g_1)))))$ to fulfill $\Psi^2_{\oplus,mrg}$.

- $(\oplus(*(inv(\pi),*(nc_1,g_1)),\ominus(g_2)),\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(\pi,g_2)))))$ in case (11) includes a confidential atomic g_2. Thus, this g_2 (resp. $*(inv(\pi'),*(\pi,g_2))$) occurs again in other pairs of the same case, e.g., with $*(inv(\pi),*(nc_3,g_1))$ (resp. $*(inv(\pi'),*(nc_3,g_1))$). Here, case (14) implies the required pair

  $$(\oplus(*(inv(\pi),*(nc_1,g_1)),\ominus(*(inv(\pi),*(nc_3,g_1)))),$$
  $$\oplus(*(inv(\pi'),*(nc_1,g_1)),\ominus(*(inv(\pi'),*(nc_3,g_1)))))$$

  to fulfill $\Psi^1_{\oplus,mrg}$ and $\Psi^2_{\oplus,mrg}$.

  In the same context, message $*(inv(\pi),*(nc_1,g_1))$ (resp. $*(inv(\pi'),*(nc_3,g_1))$) from case (11) occurs in pairs from case (14). Here, case (11) itself implies the required pair

  $$(\oplus(*(inv(\pi),*(nc_3,g_1)),\ominus(g_2)),\oplus(*(inv(\pi'),*(nc_3,g_1)),\ominus(*(inv(\pi'),*(\pi,g_2)))))$$

  to fulfill $\Psi^1_{\oplus,mrg}$ and $\Psi^2_{\oplus,mrg}$.

14.4.8 Handling of Ψ^1_{inv} and Ψ^2_{inv}:

In this section, we consider the pairs in xy that have inv-objects as 1-st or 2-nd messages, which are relevant for the handling of the proof obligations Ψ^1_{inv} and Ψ^2_{inv}. According to the definition of Φ, all these pairs are of the form

- $(inv(\pi),inv(\pi'))$ or
- $(inv(\oplus(*(nc_1,g_1),\ominus(*(pw,g_2)))),inv(\oplus(*(nc_1,g_1),\ominus(*(pw,g_2)))))$.

Hence, conditions Ψ^1_{inv} and Ψ^2_{inv} are clearly satisfied by the pair (π,π') and the pairs $(\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))),\oplus(*(nc_1,g_1),\ominus(*(pw,g_2))))$ in case (1) from Sec. 14.4.3.3.

14.4.9 Proof Obligations Handled by Refutation:

The proof obligations Ψ^i_{fst}, Ψ^i_{snd}, $\Psi^i_{h_1}$ and $\Psi^i_{h_2}$ for $i \in \{1,2\}$ are shown by refutation as described in Sec. 12.4.5.

The proof of $\Psi^1_{h_2}$ is based on Φ and the regularity property 143-(4), as sketched in the following table:

$(h_2(m_1,m_2,m_3),m_y) \in xy,(m_1,m_2,m_3 \in DY(\pi.ik))$ $\vdash^?(m_1,m_{y1}),(m_2,m_{y2}),(m_3,m_{y3}) \in xy \wedge m_y = h_2(m_{y1},m_{y2},m_{y3})$
Φ yields: $m_y = h_2(m_1,m_2,m_3),m_y \in ik,m_3 \notin DY(ik),(m_1,m_2,m_3 \in DY(\pi.ik))$ $\vdash^?(m_1,m_{y1}),(m_2,m_{y2}),(m_3,m_{y3}) \in xy \wedge h_2(m_1,m_2,m_3) = h_2(m_{y1},m_{y2},m_{y3})$ Property 143-(4) yields: $h_2(m_1,m_2,m_3) \in ik,m_3 \notin DY(pw(k).ik),(m_1,m_2,m_3 \in DY(\pi.ik)) \vdash^! \bot$

Given an arbitrary $(h_2(m_1,m_2,m_3),m_y) \in xy$, the definition of Φ binds $h_2(m_1,m_2,m_3)$ and m_y to h_2-messages in ik having confidential third h_2-part m_3. This permits to deduce by property 143-(4) that m_3 cannot occur in $DY(pw(k)\bullet ik)$ for any password $pw(k)$. Hence, this allows us to conclude by refuting the assumption $m_3 \in DY(\pi\bullet ik)$.

The proof obligations $\Psi^1_{h_2}$, $\Psi^1_{h_1}$ and $\Psi^2_{h_1}$ are shown similar to $\Psi^1_{h_2}$.

14.4.10 Handling of Γ:

Proof obligation Γ requires that every element in $kb = \pi\bullet ik$ is mapped by $\overset{xy}{\rightsquigarrow}$ to the element in $kb' = \pi'\bullet ik$ at the same position. This holds obviously for the first elements π and π', as Φ ensures $(\pi,\pi') \in xy$. For the remaining elements, the proof is by the following lemma:

Property$^{\mathbf{VSE}}$ 145 (Γ Property of $\overset{xy}{\rightsquigarrow}$):

$$
\begin{aligned}
&(tr \in \text{TCAMP} \wedge \pi \notin DY(spies(tr)) \wedge (\forall m \in spies(tr) : \neg uses(m,\pi')) \wedge \\
&\ ((\exists g_1,g_2,nc_1 : \oplus(*(nc_1,g_1),\ominus(*(\pi,g_2))) \in spies(tr)) \vee \\
&\ \ (\exists g_1,nc_2,m_{2,2} : pair(*(nc_2,*(\pi,g_1)),m_{2,2}) \in spies(tr))) \wedge \\
&\ \Phi(xy,spies(tr),\pi,\pi') \wedge m \in DY(spies(tr))) \\
&\Rightarrow m \overset{xy}{\rightsquigarrow} m
\end{aligned}
$$

Similar to property 120, this lemma is shown by induction on $|m|$.

Base Case: For $m = c_i$ and $c_i \in DY(ik)$, lemma 122 provides $c_i \overset{xy}{\leftrightarrow} m'$ with $(c_i,m') \in xy$. According to Φ, we distinguish mainly two cases:

- $(c_i,m') = (\pi,\pi')$: This yields $c_i = \pi$ and thus $\pi \in DY(ik)$, which permits to conclude by refutation based on Ω.
- (c_i,m') is of the form $(ag(j),ag(j))$, $(num(j),num(j))$, $(nc(j),nc(j))$ or $(pw(j),pw(j))$: This yields $m' = c_i$, as required.

Step Case: For an arbitrary composed $m \in DY(ik)$, we show $m \overset{xy}{\rightsquigarrow} m$ simply by Φ when $m \in dom(xy)$: According to Φ, xy includes pairs (m_1,m_2) with $m_1 \neq m_2$ only if $m_1 \notin DY(ik)$ holds. This can be simply shown with the help of the basic and other dedicated confidentiality properties. Consequently, $m \in DY(ik)$ and $(m,m') \in xy$ imply $m' = m$, as required.

In the complementary case, Ψ^c ensures that m can be composed from sub-messages in $DY(\pi\bullet ik)$. We want to show (in the following case distinction) that these sub-messages belong to $DY(ik)$, in order to obtain $m \overset{xy}{\rightsquigarrow} m$ by the induction hypothesis.

- In case $isObj^{\oplus}(m)$, we apply the regularity property 141-(2). This yields four cases:
  - There is $m_1,m_2 \in DY(ik)$ with $obj^{\oplus}(m,m_1,m_2)$ or $syn^*(m,m_1,m_2)$: Here, the induction hypothesis applies.
  - Three further cases bind m to regular messages in ik. For instance, we obtain a case where $m = \oplus(*(nc_1,g_1),\ominus(*(pw,g_2)))$, $m \in ik$ and $nc_1,pw \notin DY(ik)$ hold. This implies $(m,m) \in xy$, according to the second rule of Φ_4. Hence, we conclude this case by refutation using $m \notin dom(xy)$.
- In case $isObj^*(m)$, we apply the regularity property 140-(2). This yields four cases:

  - There is $m_1, m_2 \in DY(ik)$ with $obj^*(m, m_1, m_2)$: Here, the induction hypothesis applies.
  - Three further cases bind m to regular messages in ik. For instance, we obtain a case where $m = *(nc_1, g_1)$, $\oplus(m, \ominus(*(pw, g_2))) \in ik$, $pw, g_2 \notin DY(ik)$ and $nc_1 \notin DY(ik)$ hold. This implies $(m, m) \in xy$, according to the first rule of Φ_4. Hence, we conclude this case by refutation using $m \notin dom(xy)$.

- In case $m = h_1(m_1, m_2, m_3)$ (resp. $m = h_2(m_1, m_2, m_3)$), the regularity property 143-(2) (resp. 143-(4)) provides $m_1, m_2, m_3 \in DY(ik)$, because otherwise m must belong to $dom(xy)$.

- All other cases are trivially reduced (using corresponding regularity properties in 144) to sub-messages in $DY(ik)$, permitting to conclude with the induction hypothesis.

Chapter 15

Conclusion and Future Work

The formal analysis of cryptographic protocols is a very active research area. Various approaches have been published in the last three decades. The focus was first on the (automated) verification of standard protocol properties, i.e. confidentiality and authentication. Later it moved to application-specific protocol properties relying also on cryptographic primitives beyond the simple *enc-dec* scenario. Coping with these (new) algebraic properties is therefore crucial for future work in this field.

In this thesis, we presented a comprehensive approach for the inductive verification of cryptographic protocols based on complex algebraic specifications. Besides the verification of standard properties as well as other kinds of trace properties, our approach includes a (new) proof technique for indistinguishability properties. This allows to verify properties, like resistance against offline testing, anonymity, and fairness in e-voting.

In addition, our approach combines tool-support with the flexibility to add new concepts and enhance the degree of automatization. For that reason, it promises to broaden the scope of formal protocol verification to many new protocol families and security properties. With respect to this goal, we provide a brief summary of our contributions and propose related future work.

15.1 Extensions of Equational Systems

We have presented an approach to extend *general* equational specifications with concepts that permit the definition of recursive functions and proofs by induction. In cases where no explicit constructors for freely generated structures are given, the *implicit* structures are made explicit by predicates obj^f (one for each function symbol f with arity $n > 0$). obj^f expresses a relation between *composed objects* and *their direct substructures*, meaning that the objects are result of a constructor-type application of f to the direct substructures. Based on these predicates we describe the effect of basic operations. This extension is gained by analyzing the operational effects of an equivalent (modular) rewriting system.

Our approach is an important contribution to inductive theorem proving in general, in the same direction as [93].

Future Work: In the axiomatization schemata we use the fact that the non-orientable equations can be partitioned into permutative theories on same function symbols. We expect that our axiomatization approach can be straightforwardly adapted to the more general case where the non-orientable equations are finite equivalence class theories. It is worth to investigate and pursue this idea with further case studies from other fields of formal methods, where the models are based on more general algebraic specifications. Furthermore, the systematic generation of our axioms from a complete modular rewriting system can be exploited to implement more tool-support (see Sec. 15.4).

15.2 A New Proof Technique for Confidentiality Properties

To support confidentiality proofs we have introduced recursive check-functions that test single immediately observable messages (in ik) to protect given sets S of secrets. The check-functions are shown to be correct with respect to an inductively defined attacker model. They can be *pessimistic*, in the sense that not every item classified as "critical" actually permits the derivation of some element in S.

Our proof technique makes use of *canonical* basic check-functions (ccl_1) that are defined *the same way in all* message algebras. Additional techniques are necessary to cope with situations where ccl_1 cannot be applied directly.

- Secrets *outside*, i.e. without any protected occurrence inside, protocol messages can be *partially known* (by a description). In these cases the set S cannot be given explicitly as required by ccl_1. To solve this problem, we prove confidentiality of a secret s by reduction to substructures required for its composition. Their confidentiality is shown according to the applicable technique. For the necessary checks and the reduction theorems, we use a simplified variant of ccl_1 (ccl_2).

  The functions ccl_1 and ccl_2 are *uniform*, but restricted to secrets that cannot be obtained by *merging*.

- The problem with merging is that there are *infinitely many* items allowing a derivation of s. For s *outside* protocol messages, we dealt with this problem by a protocol-specific invariant that permits to get rid of irrelevant merging operations. It provides a *complete* case distinction on the derivable messages from ik that are of the same type as s. This is used to show that none of the derivable messages matches s.

  Appropriate invariants identify the secrets inside protocol messages and these are shown confidential using either ccl_1 or ccl_3 (see below).

- The problem with merging of secrets *inside* protocol messages is even more complicated because they are protected by other secrets to prevent their derivation by extraction. Therefore, we adapted our testing approach relative to sets S of secrets using a new check-function ccl_3 that extends ccl_1 with additional checks to integrate merging. In particular, we use *just the relevant* results of merging operations given by an additional argument P. The corresponding correctness proof of ccl_3 requires additional conditions for S and P.

As the verification of PACE and TC-AMP does not require a check-function ccl_3, we demonstrated the corresponding technique with a devised example protocol. We expect that the proposed solutions are *complete*, in the sense that confidentiality properties in other case studies and other message algebras can be verified using these techniques. We claim that new kinds of operations (if any) that pose a similar problem like merging can be handled following these lines.

Future Work: More case studies are clearly of interest to confirm our claim and to further develop our techniques.

In addition, the critical operation effects are identified by a systematic analysis of the basic operations and provide guidance for the instantiation of the schemata in the definition of the check-functions and their corresponding proofs. These proofs including the protocol-specific invariants follow a common proof schema. This can be exploited to implement stronger tool-support (see Sec. 15.4).

15.3 A Proof Technique for Indistinguishability Properties

Indistinguishability properties are defined for given pairs of knowledge bases kb, kb' using generic derivations δ and δ'. The equivalence $\delta(kb) = \delta'(kb) \Leftrightarrow \delta(kb') = \delta'(kb')$ ensures that

all algorithms using equality checks for the computation of a property-specific predicates yield same result for kb and kb'.

We have developed techniques to prove this equivalence using an appropriate simulation relation $\rightsquigarrow$ satisfying $\delta(kb) \rightsquigarrow \delta(kb')$ for all generic derivations δ.

Our axiomatization is based on an enumeration of generic derivations applied to inputs (kb/kb'). Required properties of simulation relations are proved by induction on the indices of the enumeration. This is done *only once for each message algebra* in a corresponding *central theorem*. This theorem reduces the required property to necessary and sufficient conditions on a finite set of message pairs, used to define *infinite* simulation relations by composition, and on the knowledge the attacker is able to obtain from kb and kb', respectively. They can be established by help of regularity properties verified *by trace induction*.

Noteworthy, our central theorem applies to all kinds of indistinguishability properties achieved by cryptographic protocols.

Future Work: We applied our proof technique to show resistance against offline password testing (of PACE and TC-AMP). It is worth to consider further case studies, in particular security protocols aiming at anonymity in various application contexts and fairness in e-voting.

We developed the above techniques for the PACE and the TC-AMP algebra. The parts dealing with derivations by canonical operations (constructor-type and extraction operations) are uniform, i.e. handled according to a same schema. Only the parts dealing with derivations by non-canonical operations need to be handled by more or less algebra-specific adaptations. Again, the majority of the proof steps are carried out according to same schemata and this can be for sure exploited to implement more tool-support (see Sec. 15.4).

15.4 Tool-Support and Case Studies

We applied our inductive method for the verification of PACE in the VSE tool. For that purpose, we added a second branch to the protocol verification framework of VSE which we had previously implemented for the *enc-dec* scenario. This framework includes specification facilities in form of (re-usable) abstract data type (ADT) theories and proof construction facilities in form of proof heuristics.

The re-use of ADT theories and specification schemes from the previous VSE framework allowed us to re-use many domain-specific proof heuristics. Additionally, we added frequently used proof heuristics for our new proof techniques, in order to achieve the verification of PACE with acceptable proof effort. The proof heuristics of the second branch permitted us to save up in average more than 80 % of the required interactions in a step-by-step mode.

In the thesis, we also described our proof techniques in the context of the TC-AMP algebra together with their applications to the TC-AMP protocol. The result is a fully worked out security proof. The presented specification and verification details are quite sufficient to obtain corresponding machine-checked proofs in VSE, by re-using and adapting the above mentioned new branch in the VSE tool to the TC-AMP algebra and protocol.

Future Work: Our presentation of the new techniques emphasized the generic parts and described how to deal with the algebra-specific parts by appropriate adaptations. Additionally, we structured our implementation of the second branch in VSE based on the locality feature of the VSE tool. These aspects are expected to support the adaptation of the algebra-specific ADT theories and of proof heuristics to new message algebras.

Clearly, more tool-support in VSE facilitates the replay of our verification of TC-AMP in VSE to obtain a machine-checked security proof and the application of our approach to further case studies. Support for specification tasks and more proof heuristics can be implemented based on the described specification and proof schemata. Additionally, it is worth to investigate whether domain-specific reformulation and analogy-driven adaptation techniques are helpful.

Our approach is not VSE-specific. The described general methodology provides the basis for an implementation in similar tools like Isabelle/HOL, [80], KIV, [53], and Coq, [67], where extensions do not as in VSE change the kernel inference machine.

Finally, the flexibility of the inductive method regarding extensions with fully new concepts, as demonstrated in this thesis, can be confirmed by trying the verification of new protocol classes and their application-specific properties.

Bibliography

[1] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. *Theor. Comput. Sci.*, 367(1-2):2–32, 2006.

[2] Martín Abadi and Cédric Fournet. Private authentication. *Theor. Comput. Sci.*, 322(3):427–476, September 2004.

[3] Myrto Arapinis, Tom Chothia, Eike Ritter, and Mark Ryan. Analysing unlinkability and anonymity using the applied pi calculus. *2010 23rd IEEE Computer Security Foundations Symposium*, pages 107–121, 2010.

[4] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P. C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The avispa tool for the automated validation of internet security protocols and applications. In Kousha Etessami and Sriram K. Rajamani, editors, *Computer Aided Verification*, pages 281–285, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[5] Alessandro Armando, Roberto Carbone, and Luca Compagna. Satmc: A sat-based model checker for security-critical systems. In Erika Ábrahám and Klaus Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 31–45, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[6] Alessandro Armando and Luca Compagna. Sat-based model-checking for security protocols analysis. In D. Gollmann, J. Lopez, C.A. Meadows, and E. Okamoto, editors, *International Journal of Information Security*. Springer, September 2007.

[7] Serge Autexier, Dieter Hutter, Heiko Mantel, and Axel Schairer. System description: Inka 5.0 - a logical voyager. In H.Ganzinger, editor, *Proceedings 16th International Conference on Automated Deduction, CADE-16*. Springer-Verlag, LNAI 1632, 1999.

[8] F. Baader and K. U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. *J. Symbolic Computation*, 21:211–243, 1996.

[9] Leo Bachmair and Nachum Dershowitz. Completion for rewriting modulo a congruence. *Theor. Comput. Sci.*, 67(2&3):173–201, 1989.

[10] Michael Backes, Catalin Hritcu, and Matteo Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Computer Security Foundations Symposium, 2008. CSF'08. IEEE 21st*, pages 195–209. IEEE, 2008.

[11] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A composable cryptographic library with nested operations. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 220–230, New York, NY, USA, 2003. ACM.

[12] David A. Basin, Srdjan Capkun, Patrick Schaller, and Benedikt Schmidt. Let's get physical: Models and methods for real-world security protocols. In *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009, Munich, Germany, August 17-20, 2009. Proceedings*, pages 1–22, 2009.

[13] David A. Basin, Jannik Dreier, and Ralf Sasse. Automated symbolic proofs of observational equivalence. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 1144–1155, 2015.

[14] David A. Basin, Sebastian Mödersheim, and Luca Viganò. Ofmc: A symbolic model checker for security protocols. *Int. J. Inf. Sec.*, 4(3):181–208, 2005.

[15] Giampaolo Bella, Fabio Massacci, and Lawrence C. Paulson. Verifying the SET registration protocols. *IEEE Journal on Selected Areas in Communications*, 21(1):77–87, 2003.

[16] Giampaolo Bella and Lawrence C. Paulson. Kerberos version 4: Inductive analysis of the secrecy goals. In *Computer Security - ESORICS 98, 5th European Symposium on Research in Computer Security, Louvain-la-Neuve, Belgium, September 16-18, 1998, Proceedings*, pages 361–375, 1998.

[17] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *SP '92: Proceedings of the 1992 IEEE Symposium on Security and Privacy*, page 72, Washington, DC, USA, 1992. IEEE Computer Society.

[18] Jens Bender, D. Kügler, M. Margraf, and I. Naumann. Privacy-friendly revocation management without unique chip identifiers for the german national id card. *Computer Fraud and Security*, 2010:14–17, 2010.

[19] J. A. Bergstra and J. V. Tucker. Equational specifications, complete term rewriting systems, and computable and semicomputable algebras. *Journal of the Association for Computing Machinery*, 42:1194–1230, 1995.

[20] Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 82–96, Cape Breton, Nova Scotia, Canada, June 2001. IEEE Computer Society.

[21] Bruno Blanchet. Automatic verification of correspondences for security protocols. *Journal of Computer Security*, 17(4):363–434, July 2009.

[22] Bruno Blanchet. Using Horn clauses for analyzing security protocols. In Véronique Cortier and Steve Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*, volume 5 of *Cryptology and Information Security Series*, pages 86–111. IOS Press, MAR 2011.

[23] Bruno Blanchet. Modeling and verifying security protocols with the applied pi calculus and ProVerif. *Foundations and Trends in Privacy and Security*, 1(1–2):1–135, October 2016.

[24] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.

[25] Yohan Boichut, Nikolai Kosmatov, Laurent Vigneron, Y. Boichut, N. Kosmatov, and L. Vigneron. Validation of prouvé protocols using the automatic tool ta4sp. In *3rd Taiwanese-French Conference on Information Technology*, pages 467–480, 2006.

[26] Thierry Boy de la Tour and Mnacho Echenim. Permutative rewriting and unification. *Inf. Comput.*, 205(4):624–650, April 2007.

[27] Alan Bundy, Frank van Harmelen, Christian Horn, and Alan Smaill. The oyster-clam system. In Mark E. Stickel, editor, *10th International Conference on Automated Deduction, Kaiserslautern, FRG, July 24-27, 1990, Proceedings*, volume 449 of *Lecture Notes in Computer Science*, pages 647–648. Springer, 1990.

[28] Denis Butin, David Gray, and Giampaolo Bella. Towards verifying voter privacy through unlinkability. In *Proceedings of the 5th International Conference on Engineering Secure Software and Systems*, ESSoS'13, pages 91–106, Berlin, Heidelberg, 2013. Springer-Verlag.

[29] Rohit Chadha, Ştefan Ciobâcă, and Steve Kremer. Automated verification of equivalence properties of cryptographic protocols. In Helmut Seidl, editor, *Programming Languages and Systems —Proceedings of the 21th European Symposium on Programming (ESOP'12)*, volume 7211 of *Lecture Notes in Computer Science*, pages 108–127, Tallinn, Estonia, March 2012. Springer.

[30] Lassaad Cheikhrouhou, Andreas Nonnengart, Werner Stephan, Frank Koob, and Georg Rock. Automating interactive protocol verification. In *KI 2008: Advances in Artificial Intelligence, 31st Annual German Conference on AI, KI 2008, Kaiserslautern, Germany, September 23-26, 2008. Proceedings*, volume 5243 of *Lecture Notes in Computer Science*, pages 30–37. Springer, 2008.

[31] Lassaad Cheikhrouhou, Georg Rock, Werner Stephan, Matthias Schwan, and Gunter Lassmann. Verifying a chip-card-based biometric identification protocol in VSE. In Janusz Górski, editor, *Proceedings of the 25th International Conference on Computer Safety, Security and Reliability (SAFECOMP 2006)*, volume 4166 of *Lecture Notes in Computer Science*, pages 42–56. Springer-Verlag, 2006.

[32] Lassaad Cheikhrouhou, Werner Stephan, Özgür Dagdelen, Marc Fischlin, and Markus Ullmann. Merging the cryptographic security analysis and the algebraic-logic security proof of PACE. In Neeraj Suri and Michael Waidner, editors, *Sicherheit 2012: Sicherheit, Schutz und Zuverlässigkeit, Beiträge der 6. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI), 7.-9. März 2012 in Darmstadt*, volume P-195 of *LNI*, pages 83–94. GI, 2012.

[33] Lassaad Cheikhrouhou, Werner Stephan, and Markus Ullmann. A new approach to the inductive verification of cryptographic protocols based on message algebras. In Marek Kosta and Thomas Sturm, editors, *MACIS 2013 –Fifth International Conference on Mathematical Aspects of Computer and Information Sciences*, 2013.

[34] Vincent Cheval. Apte: an algorithm for proving trace equivalence. In Erika Ábrahám and JKlaus Havelund, editors, *Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14)*, volume 8413 of *Lecture Notes in Computer Science*, pages 587–592, Grenoble, France, April 2014. Springer Berlin Heidelberg.

[35] Vincent Cheval and Bruno Blanchet. Proving more observational equivalences with proverif. In *Proceedings of the Second International Conference on Principles of Security and Trust*, POST'13, pages 226–246, Berlin, Heidelberg, 2013. Springer-Verlag.

[36] Yannick Chevalier, Luca Compagna, Jorge Cuellar, Paul Hankes Drielsma, Jacopo Mantovani, Sebastian Moedersheim, and Laurent Vigneron. A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. In *Workshop on Specification and Automated Processing of Security Requirements - SAPS'2004*, page 13 p,

Linz, Austria, 2004. Austrian Computer Society. Colloque avec actes et comité de lecture. internationale.

[37] Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. An NP decision procedure for protocol insecurity with XOR. *Theor. Comput. Sci.*, 338(1-3):247–274, 2005.

[38] Hubert Comon-Lundh and Stéphanie Delaune. The finite variant property: How to get rid of some algebraic properties. In *Proceedings of the 16th International Conference on Term Rewriting and Applications*, RTA'05, pages 294–307, Berlin, Heidelberg, 2005. Springer-Verlag.

[39] Hubert Comon-Lundh and Stéphanie Delaune. The finite variant property: How to get rid of some algebraic properties. In Jürgen Giesl, editor, *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *Lecture Notes in Computer Science*, pages 294–307, Nara, Japan, April 2005. Springer.

[40] Ricardo Corin, Jeroen Doumen, and Sandro Etalle. Analysing password protocol security against off-line dictionary attacks. *Electr. Notes Theor. Comput. Sci.*, 121:47–63, 2005.

[41] Véronique Cortier and Stéphanie Delaune. A method for proving observational equivalence. In *Proceedings of the 22nd IEEE Computer Security Foundations Symposium, CSF 2009, Port Jefferson, New York, USA, July 8-10, 2009*, pages 266–276, 2009.

[42] Véronique Cortier, Stéphanie Delaune, and Pascal Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.

[43] Véronique Cortier and Bogdan Warinschi. Computationally sound, automated proofs for security protocols. In Shmuel Sagiv, editor, *Programming Languages and Systems, 14th European Symposium on Programming,ESOP 2005*, volume 3444 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.

[44] Cas J. Cremers. The scyther tool: Verification, falsification, and analysis of security protocols. In *Proceedings of the 20th International Conference on Computer Aided Verification*, CAV '08, pages 414–418, Berlin, Heidelberg, 2008. Springer-Verlag.

[45] Cas J. F. Cremers and Dennis Jackson. Prime, order please! revisiting small subgroup and invalid curve attacks on protocols using diffie-hellman. *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*, pages 78–7815, 2019.

[46] Casper: A Compiler for the Analysis of Security Protocols. `http://www.cs.ox.ac.uk/people/gavin.lowe/Security/Casper/index.html`. visited on 2022-01-30.

[47] FDR2 User Manual. `http://www.cs.ox.ac.uk/projects/concurrency-tools/fdr-2.94-html-manual/index.html`. visited on 2022-01-29.

[48] D. Dolev and A. C. Yao. On the security of public key protocols. In *SFCS '81: Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*, pages 350–357, Washington, DC, USA, 1981. IEEE Computer Society.

[49] Ben Donovan, Paul Norris, and Gavin Lowe. Analyzing a library of security protocols using Casper and FDR. In *Proceedings of the Workshop on Formal Methods and Security Protocols*, 1999.

[50] Jannik Dreier, Charles Duménil, Steve Kremer, and Ralf Sasse. Beyond subterm-convergent equational theories in automated verification of stateful protocols. In Matteo Maffei and Mark Ryan, editors, *Principles of Security and Trust*, pages 117–140, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg.

[51] Jannik Dreier, Lucca Hirschi, Saša Radomirović, and Ralf Sasse. Verification of stateful cryptographic protocols with exclusive or. *Journal of Computer Security*, 28(1), February 2020.

[52] Francisco Durán, Steven Eker, Santiago Escobar, José Meseguer, and Carolyn L. Talcott. Variants, unification, narrowing, and symbolic reachability in maude 2.6. In *Proceedings of the 22nd International Conference on Rewriting Techniques and Applications, RTA 2011, May 30 - June 1, 2011, Novi Sad, Serbia*, pages 31–40, 2011.

[53] Gidon Ernst, Jörg Pfähler, Gerhard Schellhorn, Dominik Haneberg, and Wolfgang Reif. KIV: overview and verifythis competition. *Int. J. Softw. Tools Technol. Transf.*, 17(6):677–694, 2015.

[54] Santiago Escobar, Catherine A. Meadows, and José Meseguer. State space reduction in the maude-nrl protocol analyzer. In *Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*, pages 548–562, 2008.

[55] F. Javier Thayer Fábrega. Strand spaces: Proving security protocols correct. *J. Comput. Secur.*, 7(2-3):191–230, March 1999.

[56] Kryptographische Vorgaben für Projekte der Bundesregierung, Teil 3: Intelligente Messsysteme. BSI TR-03116, 3. March 2021.

[57] The electronic ID card. `https://www.bsi.bund.de/EN/Topics/ElectrIDDocuments/eIDcard/eIDcard_node.html`. visited on 2022-01-30.

[58] Bernd Finkbeiner, Christian Müller, Helmut Seidl, and Eugen Zălinescu. Verifying security policies in multi-agent workflows with loops. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, Oct 2017.

[59] Federal Office for Information Security (BSI). Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token – Part 1 – eMRTDs with BAC/PACEv2 and EACv1, BSI TR-03110-1, Version 2.20, 26. February 2015. `https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03110/TR-03110_node.html`. visited on 2022-01-30.

[60] Federal Office for Information Security (BSI). Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token – Part 2 – Protocols for electronic IDentification, Authentication and trust Services (eIDAS), BSI TR-03110-2, Version 2.21. `https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03110/TR-03110_node.html`. visited on 2022-01-30.

[61] Federal Office for Information Security (BSI). Technical Guideline: Advanced Security Mechanisms for Machine Readable Travel Documents – Extended Access Control (EAC), BSI TR-03110, Version 1.0. `https://www.befreite-dokumente.de/www.befreite-dokumente.de/eingereichte-akten/tr-03110-eac-1.0/attachment_download/publication_download.pdf`. visited on 2022-01-20.

[62] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, ASIACRYPT '92, pages 244–251, London, UK, UK, 1993. Springer-Verlag.

[63] Lucca Hirschi, David Baelde, and Stéphanie Delaune. A method for unbounded verification of privacy-type properties. *J. Comput. Secur.*, 27(3):277–342, 2019.

[64] Gerard Huet. Confluence reductions: Abstract properties and applications to term rewriting systems. *J. ACM*, 1980.

[65] Dieter Hutter, Bruno Langenstein, Georg Rock, Jörg H. Siekmann, Werner Stephan, and Roland Vogt. Formal software development in the verification support environment (vse). *Journal of Experimental and Theoretical Artificial Intelligence*, 12:383–406, 2000.

[66] Doc 9303: Machine Readable Travel Documents, Eighth Edition, 2021, Part 11: Security Mechanisms for MRTDs. `https://www.icao.int/publications/Documents/9303_p11_cons_en.pdf`. visited on 2022-01-30.

[67] The Coq Proof Assistant. `https://coq.inria.fr/`. visited on 2022-01-30.

[68] David P. Jablon. Strong password-only authenticated key exchange. *SIGCOMM Comput. Commun. Rev.*, 26(5):5–26, 1996.

[69] David P. Jablon. Extended password key exchange protocols immune to dictionary attack. In *Proceedings of the WETICE'97 Workshop on Enterprise Security*, Cambridge, MA, USA, 1997.

[70] Jean-Pierre Jouannaud. Confluent and coherent equational term rewriting systems: ... *SIAM J. Comput.*, 15(4):1155–1194, November 1986.

[71] Jean-Pierre Jouannaud and Hélène Kirchner. Completion of a set of rules modulo a set of equations. *SIAM J. Comput.*, 15(4):1155–1194, November 1986.

[72] Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *Programming Languages and Systems, 14th European Symposium on Programming,ESOP 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*, pages 186–200, 2005.

[73] Taekyoung Kwon. Practical authenticated key agreement using passwords. In *the 7th Information Security Conference (ISC*, pages 1–12. Springer-Verlag, 2004.

[74] Yongjian Li and Jun Pang. Formalizing provable anonymity in isabelle/hol. *Form. Asp. Comput.*, 27(2):255–282, March 2015.

[75] Jacques Loeckx, Hans-Dieter Ehrich, and Markus Wolf. *Specification of abstract data types*. Wiley, 1996.

[76] Catherine A. Meadows. The NRL protocol analyzer: An overview. *J. Log. Program.*, 26(2):113–131, 1996.

[77] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The tamarin prover for the symbolic analysis of security protocols. In *Proceedings of the 25th International Conference on Computer Aided Verification*, CAV'13, pages 696–701, Berlin, Heidelberg, 2013. Springer-Verlag.

[78] Markus Müller. A Comparison of Tools for Cryptographic Protocol Analysis. Master's thesis, Saarland University, Germany, 2012.

[79] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, December 1978.

[80] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.

[81] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proceedings of the 5th International Workshop on Security Protocols*, pages 25–35, London, UK, UK, 1998. Springer-Verlag.

[82] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1-2):85–128, 1998.

[83] Lawrence C. Paulson. Inductive analysis of the internet protocol TLS. *ACM Trans. Inf. Syst. Secur.*, 2(3):332–351, 1999.

[84] Gerald E. Peterson and Mark E. Stickel. Complete sets of reductions for some equational theories. *J. ACM*, 28(2):233–264, April 1981.

[85] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, May 1998.

[86] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, November 1998.

[87] Georg Rock, Gunter Lassmann, Mathias Schwan, and Lassaad Cheikhrouhou. Verisoft–secure biometric identification system. In *Digital Excellence*, pages 83–97. Springer, 2008.

[88] Sonia Santiago, Santiago Escobar, Catherine A. Meadows, and José Meseguer. A formal definition of protocol indistinguishability and its verification using maude-npa. In Sjouke Mauw and Christian Damsgaard Jensen, editors, *Security and Trust Management - 10th International Workshop, STM 2014, Wroclaw, Poland, September 10-11, 2014. Proceedings*, volume 8743 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2014.

[89] Ralf Sasse, Santiago Escobar, Catherine Meadows, and José Meseguer. Protocol analysis modulo combination of theories: A case study in maude-npa. In *Proceedings of the 6th International Conference on Security and Trust Management*, STM'10, pages 163–178, Berlin, Heidelberg, 2011. Springer-Verlag.

[90] Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. Automated analysis of diffie-hellman protocols and advanced security properties. In *2012 IEEE 25th Computer Security Foundations Symposium*, pages 78–94, 2012.

[91] Benedikt Schmidt, Simon Meier, Cas J. F. Cremers, and David A. Basin. Automated analysis of diffie-hellman protocols and advanced security properties. *2012 IEEE 25th Computer Security Foundations Symposium*, pages 78–94, 2012.

[92] Benedikt Schmidt, Ralf Sasse, Cas J. F. Cremers, and David A. Basin. Automated verification of group key agreement protocols. *2014 IEEE Symposium on Security and Privacy*, pages 179–194, 2014.

[93] Claus Sengler. Induction on non-freely generated data types. Technical report, DFKI, 1996.

[94] Jörg H. Siekmann. Unification theory. *J. Symb. Comput.*, 7(3-4):207–274, March 1989.

[95] D. Singelee and B. Preneel. Location verification using secure distance bounding protocols. In *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pages 7 pp.–840, Nov 2005.

[96] Dawn Xiaodong Song. Athena: A new efficient automatic checker for security protocol analysis. In *Proceedings of the 12th IEEE Workshop on Computer Security Foundations*, CSFW '99, pages 192–, Washington, DC, USA, 1999. IEEE Computer Society.

[97] Common Authentication Protocol Specification Language (CAPSL). `http://www.csl.sri.com/projects/capsl/`. visited on 2022-01-30.

[98] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. Anonymous connections and onion routing. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, SP '97, pages 44–, Washington, DC, USA, 1997. IEEE Computer Society.

[99] Tamarin-Prover Manual, Security Protocol Analysis in the Symbolic Model. `https://tamarin-prover.github.io/manual/tex/tamarin-manual.pdf`. visited on 2022-01-30.

[100] Mathieu Turuani. The cl-atse protocol analyser. In Frank Pfenning, editor, *Term Rewriting and Applications*, pages 277–286, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[101] Markus Ullmann, Dennis Kügler, Heike Neumann, Sebastian Stappert, and Vögeler Matthias. Password authenticated key agreement for contactless smart cards. In *Proceedings of the 4-th Workshop on RFID Security, Budapest*, pages 140–161, 2008.

[102] Luca Viganò. Automated security protocol analysis with the avispa tool. *Electronic Notes in Theoretical Computer Science*, 155:61–86, 2006.

[103] Christoph Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *CADE*, 1999.

[104] Sarah Winkler and Aart Middeldorp. Normalized completion revisited. In Femke van Raamsdonk, editor, *Proceedings of the 24th International Conference on Rewriting Techniques and Applications*, volume 21 of *Leibniz International Proceedings in Informatics*, pages 319–334, 2013.

[105] Thomas Wu. The secure remote password protocol. In *Internet Society Symposium on Network and Distributed System Security*, pages 97–111, 1998.

[106] F. Zamanian and H. Mala. A new anonymous unlinkable mobile payment protocol. In *2016 6th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 117–122, 2016.

Appendix A

Completion of the TC-AMP Equations

In this appendix we give an excerpt about the completion of the equations in the TC-AMP algebra. As explained in Sec. 4.6 we start the completion with the following 10 rewrite rules:

$$
\begin{aligned}
r_1 &: \quad fst(pair(x,y)) \to x \\
r_2 &: \quad snd(pair(x,y)) \to y \\
r_3 &: \quad \oplus(x,\ominus(x)) \to \infty \\
r_4 &: \quad \oplus(x,\infty) \to x \\
r_5 &: \quad *(x,*(inv(x),y)) \to y \\
r_6 &: \quad inv(inv(x)) \to x \\
r_7 &: \quad *(x,\oplus(y,z)) \to \oplus(*(x,y),*(x,z)) \\
r_8 &: \quad \ominus(\infty) \to \infty \\
r_9 &: \quad \ominus(\ominus(x)) \to x \\
r_{10} &: \quad \ominus(\oplus(x,y)) \to \oplus(\ominus(x),\ominus(y))
\end{aligned}
$$

Applying our completion procedure in Sec. 4.4 (by hand), we added two rewrite rules:

$$
\begin{aligned}
r_{11} &: \quad *(x,\infty) \to \infty \\
r_{12} &: \quad *(x,\ominus(y)) \to \ominus(*(x,y))
\end{aligned}
$$

In the following we explain how rules r_{11}–r_{12} are obtained and give an excerpt about the confluence check of the resulting rewrite system.

A.1 Adding Rule $r_{11} : *(x,\infty) \to \infty$

In the superposition of $r_4 : \oplus(u,\infty) \to u$ on $r_7 : *(x,\oplus(y,z)) \to \oplus(*(x,y),*(x,z))$, the unification $\oplus(u,\infty) =_u \oplus(y,z)$ yields

1. $\sigma_1 = \{y \mapsto u, z \mapsto \infty\}$ and
2. $\sigma_2 = \{y \mapsto \oplus(\infty,x_1), u \mapsto \oplus(z,x_1)\}$.

In case (1), we need to join $\oplus(*(x,u),*(x,\infty))$ and $*(x,u)$.
In case (2), we need to join the terms $\oplus(*(x,\oplus(\infty,x_1)),*(x,z))$ and $*(x,\oplus(z,x_1))$, i.e. the terms $\oplus(\oplus(*(x,\infty),*(x,x_1)),*(x,z))$ and $\oplus(*(x,z),*(x,x_1))$.

This requires to add a first intermediate rule $r_1' : \oplus(*(x,u),*(x,\infty)) \to *(x,u)$.

In the superposition of $r_5 : *(x,*(inv(x),y)) \to y$ on $r_1' : \oplus(*(u,v),*(u,\infty)) \to *(u,v)$, the unification $*(x,*(inv(x),y)) =_u *(u,v)$ yields

1. $\sigma_1 = \{u \mapsto x, v \mapsto *(inv(x),y)\}$,
2. $\sigma_2 = \{v \mapsto *(x,x_1), u \mapsto inv(x), y \mapsto x_1\}$, and
3. $\sigma_3 = \{v \mapsto *(x,*(inv(x),x_2)), y \mapsto *(u,x_2)\}$.

In case (1), we need to join $*(x,*(inv(x),y))$, i.e. y, and $\oplus(y,*(x,\infty))$.
In case (2), we need to join $*(inv(x),*(x,x_1))$, i.e. x_1, and $\oplus(x_1,*(inv(x),\infty))$.
In case (3), we need to join $*(u,*(x,*(inv(x),x_2)))$, i.e. $*(u,x_2)$, and $\oplus(*(u,x_2),*(u,\infty))$.
This requires to add a second intermediate rule $r_2' : \oplus(y,*(x,\infty)) \to y$.

In the superposition of $r_4 : \oplus(x,\infty) \to x$ with $r_2' : \oplus(u,*(v,\infty)) \to u$, the unification $\oplus(x,\infty) =_u \oplus(u,*(v,\infty))$ yields

- $\sigma_1 = \{u \mapsto \infty, x \mapsto *(v,\infty)\}$, and
- $\sigma_2 = \{x \mapsto \oplus(*(v,\infty),x_1), u \mapsto \oplus(\infty,x_1)\}$.

In case (1), we need to join ∞ and $*(v,\infty)$.
In case (2), we need to join $\oplus(\infty,x_1)$, i.e. x_1, and $\oplus(*(v,\infty),x_1)$.
This requires to add rule $r_{11} : *(v,\infty) \to \infty$. Afterwards, rules r_1' and r_2' become redundant.

A.2 Adding Rule $r_{12} : *(x,\ominus(y)) \to \ominus(*(x,y))$

In the superposition of $r_3 : \oplus(u,\ominus u) \to \infty$ on $r_7 : *(x,\oplus(y,z)) \to \oplus(*(x,y),*(x,z))$, the unification $\oplus(u,\ominus u) =_u \oplus(y,z)$ yields

1. $\sigma_1 = \{y \mapsto u, z \mapsto \ominus(u)\}$ and
2. $\sigma_2 = \{y \mapsto \oplus(\ominus(\oplus(z,x_1)),x_1), u \mapsto \oplus(z,x_1)\}$.

In case (1), we need to join $*(x,\infty)$, i.e. ∞, and $\oplus(*(x,u),*(x,\ominus(u)))$.
In case (2), we need to join $*(x,\infty)$ and $\oplus(*(x,\oplus(\ominus(\oplus(z,x_1)),x_1)),*(x,z))$, i.e. ∞ and $\oplus(\oplus(\oplus(*(x,\ominus(z)),*(x,\ominus(x_1))),*(x,x_1)),*(x,z))$.
This requires to add a first intermediate rule $r_1' : \oplus(*(x,u),*(x,\ominus(u))) \to \infty$

In the superposition of $r_1' : \oplus(*(u,v),*(u,\ominus(v))) \to \infty$ and $r_3 : \oplus(x,\ominus x) \to \infty$, the unification $\oplus(x_1,\oplus(x,\ominus(x))) =_u \oplus(x_2,\oplus(*(u,v),*(u,\ominus(v))))$ yields

1. $\sigma_1 = \{x_1 \mapsto \oplus(*(u,v),*(u,\ominus(v))), x_2 \mapsto \oplus(x,\ominus(x))\}$,
2. $\sigma_2 = \{x_1 \mapsto \oplus(\oplus(*(u,v),*(u,\ominus(v))),x_3), x_2 \mapsto \oplus(x_3,\oplus(x,\ominus(x)))\}$,
3. $\sigma_3 = \{x_1 \mapsto *(u,\ominus(v)), x_2 \mapsto \ominus(*(u,v))\}$,
4. $\sigma_4 = \{x_1 \mapsto *(u,\ominus(v)), x_2 \mapsto \oplus(x_4,\ominus(\oplus(*(u,v),x_4)))\}$,
5. $\sigma_5 = \{x_1 \mapsto *(u,v), x_2 \mapsto \ominus(*(u,\ominus(v)))\}$,
6. $\sigma_6 = \{x_1 \mapsto *(u,v), x_2 \mapsto \oplus(x_4,\ominus(\oplus(*(u,\ominus(v)),x_4)))\}$,
7. $\sigma_7 = \{x_2 \mapsto \oplus(\ominus(\oplus(*(u,v),*(u,\ominus(v)))),x_1)\}$,

8. $\sigma_8 = \{x_2 \mapsto \oplus(\oplus(x_4,\ominus(\oplus(\oplus(*(u,v),*(u,\ominus(v))),x_4))),x_1)\}$,
9. $\sigma_9 = \{x_1 \mapsto \oplus(x_3,*(u,v)),x_2 \mapsto \oplus(x_3,\ominus(*(u,\ominus(v))))\}$,
10. $\sigma_{10} = \{x_1 \mapsto \oplus(x_3,*(u,v)),x_2 \mapsto \oplus(x_3,\oplus(x_7,\ominus(\oplus(*(u,\ominus(v)),x_7))))\}$,
11. $\sigma_{11} = \{x_1 \mapsto \oplus(x_3,*(u,\ominus(v))),x_2 \mapsto \oplus(x_3,\ominus(*(u,v)))\}$, and
12. $\sigma_{12} = \{x_1 \mapsto \oplus(x_3,*(u,\ominus(v))),x_2 \mapsto \oplus(x_3,\oplus(x_7,\ominus(\oplus(*(u,v),x_7))))\}$.

Cases (1) and (2) do not really correspond to overlaps.
In case (3), we need to join $*(u,\ominus(v))$ and $\ominus(*(u,v))$.
In case (4), we need to join $*(u,\ominus(v))$ and $\oplus(x_4,\ominus(\oplus(*(u,v),x_4)))$, i.e. $\ominus(*(u,v))$.
In case (5), we need to join $*(u,v)$ and $\ominus(*(u,\ominus(v)))$.
In case (6), we need to join $*(u,v)$ and $\oplus(x_4,\ominus(\oplus(*(u,\ominus(v)),x_4)))$, i.e. $\ominus(*(u,\ominus(v)))$.
In case (7), we need to join x_1 and $\oplus(\ominus(\oplus(*(u,v),*(u,\ominus(v)))),x_1)$, i.e. x_1.
In case (8), we need to join x_1 and $\oplus(\oplus(x_4,\ominus(\oplus(\oplus(*(u,v),*(u,\ominus(v))),x_4))),x_1)$, i.e. x_1.
In case (9), we need to join $\oplus(x_3,*(u,v))$ and $\oplus(x_3,\ominus(*(u,\ominus(v))))$.
In case (10), we need to join $\oplus(x_3,*(u,v))$ and $\oplus(x_3,\oplus(x_7,\ominus(\oplus(*(u,\ominus(v)),x_7))))$, i.e. $\oplus(x_3,\ominus(*(u,\ominus(v))))$.
In case (11), we need to join $\oplus(x_3,*(u,\ominus(v)))$ and $\oplus(x_3,\ominus(*(u,v)))$.
In case (12), we need to join $\oplus(x_3,*(u,\ominus(v)))$ and $\oplus(x_3,\oplus(x_7,\ominus(\oplus(*(u,v),x_7))))$, i.e. $\oplus(x_3,\ominus(\oplus(*(u,v))))$.

This requires to add rule $r_{12} : *(u,\ominus(v)) \to \ominus(*(u,v))$. Afterwards, rule r_1' becomes redundant.

A.3 Overlaps of $*$-terms

A.3.1 Superposition of r_5 with r_5:

$r_5 : *(x,*(inv(x),y)) \to y$ and $r_5 : *(u,*(inv(u),v)) \to v$

Overlap at Top Position:

The unification $*(x,*(inv(x),y)) =_u *(u,*(inv(u),v))$ yields

1. $\sigma_1 = \{u \mapsto x, y \mapsto v\}$, and
2. $\sigma_2 = \{u \mapsto inv(x), y \mapsto *(inv(inv(x)),w), v \mapsto *(x,w)\}$.

Case (1) is trivial.
In case (2), we obtain $*(inv(inv(x)),w)$ and $*(x,w)$, where the former reduces to $*(x,w)$.

Overlap at Position 2:

In overlap of $*(x,*(inv(x),y))$ on $*(u,*(inv(u),v))$, the unification $*(x,*(inv(x),y)) =_u *(inv(u),v)$ yields

1. $\sigma_1 = \{x \mapsto inv(u), v \mapsto *(inv(inv(u)),y)\}$, and
2. $\sigma_2 = \{x \mapsto u, v \mapsto *(u,y)\}$.

In case (1), we obtain $*(inv(inv(u)),y)$ and $*(u,y)$, where the former reduces to $*(u,y)$.
In case (2), we obtain $*(u,y)$ and $*(u,y)$, which are equal.

In overlap of $*(x,*(inv(x),y))$ on $*(inv(u),*(u,v))$, the unification $*(x,*(inv(x),y)) =_u *(u,v)$ yields

1. $\sigma_1 = \{u \mapsto x, v \mapsto *(inv(x),y)\}$, and

2. $\sigma_2 = \{u \mapsto inv(x), v \mapsto *(x,y)\}$.

In case (1), we obtain $*(inv(x),y)$ and $*(inv(x),y)$, which are equal.
In case (2), we obtain $*(x,y)$ and $*(inv(inv(x)),y)$, where the latter reduces to $*(x,y)$.

Equational Overlap:

2-2 contexts: The unification

$$*(v_1,*(v_2,*(x,*(inv(x),y)))) =_u *(v_3,*(v_4,*(u,*(inv(u),v)))),$$

where $\{v_1,v_2\} = \{u,inv(u)\}$ and $\{v_3,v_4\} = \{x,inv(x)\}$ hold, yields one case (modulo the switch of v_i-s)

1. $\sigma_1 = \{v_1 \mapsto u, v_2 \mapsto inv(u), v_3 \mapsto x, v_4 \mapsto inv(x), y \mapsto v\}$.

Here, we obtain $*(u,*(inv(u),v))$ and $*(x,*(inv(x),v))$, which are reducible to v.

1-2 contexts: The unification $*(v_1,*(x,*(inv(x),y))) =_u *(v_2,*(v_3,*(u,*(inv(u),v))))$, where $v_1 \in \{u,inv(u)\}$ and $\{v_2,v_3\} = \{x,inv(x)\}$ hold, yields two cases (modulo the switch of v_2 and v_3)

1. $\sigma_1 = \{v_1 \mapsto u, v_2 \mapsto x, v_3 \mapsto inv(x), y \mapsto *(inv(u),v)\}$ and
2. $\sigma_2 = \{v_1 \mapsto inv(u), v_2 \mapsto x, v_3 \mapsto inv(x), y \mapsto *(u,v)\}$.

In case (1), we obtain $*(u,*(inv(u),v))$ and $*(x,*(inv(x),v))$, which are reducible to v. In case (2), we obtain $*(inv(u),*(u,v))$ and $*(x,*(inv(x),v))$, which are reducible to v.

1-1 contexts: The unification $*(v_1,*(x,*(inv(x),y))) =_u *(v_2,*(u,*(inv(u),v)))$, where $v_1 \in \{u,inv(u)\}$ and $v_2 \in \{x,inv(x)\}$ hold, yields four cases (modulo the switch of v_2 and v_3)

1. $\sigma_1 = \{v_1 \mapsto u, v_2 \mapsto x, x \mapsto u, y \mapsto v\}$,
2. $\sigma_2 = \{v_1 \mapsto u, v_2 \mapsto inv(x), x \mapsto inv(u), y \mapsto v\}$,
3. $\sigma_3 = \{v_1 \mapsto inv(u), v_2 \mapsto x, u \mapsto inv(x), y \mapsto v\}$ and
4. $\sigma_4 = \{v_1 \mapsto inv(u), v_2 \mapsto inv(x), u \mapsto x, y \mapsto v\}$.

In case (1), we obtain $*(u,v)$ and $*(u,v)$.
In case (2), we obtain $*(u,v)$ and $*(inv(inv(u)),v)$, where the latter reduces to $*(u,v)$.
In case (3), we obtain $*(inv(inv(x)),v)$ and $*(x,v)$, where the former reduces to $*(x,v)$.
In case (4), we obtain $*(inv(x),v)$ and $*(inv(x),v)$.

A.3.2 Superposition of r_5 with r_7:

$r_5 : *(x,*(inv(x),y)) \to y$ and $r_7 : *(u,\oplus(v,w)) \to \oplus(*(u,v),*(u,w))$

Overlap at Top Position:

The unification $*(x,*(inv(x),y)) =_u *(u,\oplus(v,w))$ fails.

Overlap of r_7 on Position 2 of r_5:

In overlap of r_7 on $*(x,*(inv(x),y))$, the unification $*(u,\oplus(v,w)) =_u *(inv(x),y)$ yields

1. $\sigma_1 = \{u \mapsto inv(x), y \mapsto \oplus(v,w)\}$.

We obtain $\oplus(v,w)$ and $*(x,\oplus(*(inv(x),v),*(inv(x),w)))$, where the latter is reducible to $\oplus(v,w)$.

In overlap of r_7 on $*(inv(x),*(x,y))$, the unification $*(u,\oplus(v,w)) =_u *(x,y)$ yields

1. $\sigma_1 = \{u \mapsto x, y \mapsto \oplus(v,w)\}$.

We obtain $\oplus(v,w)$ and $*(inv(x),\oplus(*(x,v),*(x,w)))$, where the latter is reducible to $\oplus(v,w)$.

Equational Overlap:

1-2 contexts: The unification $*(v_1,*(x,*(inv(x),y))) =_u *(v_2,*(v_3,*(u,\oplus(v,w))))$, where $v_1 \in \{u\}$ and $\{v_2,v_3\} = \{x,inv(x)\}$ hold, yields one case (modulo the switch of v_2 and v_3)

1. $\sigma_1 = \{v_1 \mapsto u, v_2 \mapsto x, v_3 \mapsto inv(x), y \mapsto \oplus(v,w)\}$.

Here, we obtain $*(u,\oplus(v,w))$ and $*(x,*(inv(x),\oplus(*(u,v),*(u,w))))$, which are reducible to $\oplus(*(u,v),*(u,w))$.

1-1 contexts: The unification $*(v_1,*(x,*(inv(x),y))) =_u *(v_2,*(u,\oplus(v,w)))$ fails.

A.3.3 Superposition of r_5 with r_{11}:

$r_5 : *(x,*(inv(x),y)) \to y$ and $r_{11} : *(u,\infty) \to \infty$

Overlap at Top Position:

The unification $*(x,*(inv(x),y)) =_u *(u,\infty)$ fails.

Overlap of r_{11} on Position 2 of r_5:

In overlap of r_{11} on $*(x,*(inv(x),y))$, the unification $*(u,\infty) =_u *(inv(x),y)$ yields

1. $\sigma_1 = \{u \mapsto inv(x), y \mapsto \infty\}$.

We obtain ∞ and $*(x,\infty)$, where the latter is reducible to ∞.

In overlap of r_{11} on $*(inv(x),*(x,y))$, the unification $*(u,\infty) =_u *(x,y)$ yields

1. $\sigma_1 = \{u \mapsto x, y \mapsto \infty\}$.

We obtain ∞ and $*(inv(x),\infty)$, where the latter is reducible to ∞.

Equational Overlap:

1-2 contexts: The unification $*(v_1,*(x,*(inv(x),y))) =_u *(v_2,*(v_3,*(u,\infty)))$, where $v_1 \in \{u\}$ and $\{v_2,v_3\} = \{x,inv(x)\}$ hold, yields one case (modulo the switch of v_2 and v_3)

1. $\sigma_1 = \{v_1 \mapsto u, v_2 \mapsto x, v_3 \mapsto inv(x), y \mapsto \infty\}$.

Here, we obtain $*(u,\infty)$ and $*(x,*(inv(x),\infty))$, which are reducible to ∞.

1-1 contexts: The unification $*(v_1,*(x,*(inv(x),y))) =_u *(v_2,*(u,\infty))$ fails.

A.3.4 Superposition of r_5 with r_{12}:

$r_5 : *(x,*(inv(x),y)) \to y$ and $r_{12} : *(u,\ominus(v)) \to \ominus(*(u,v))$

Overlap at Top Position:

The unification $*(x,*(inv(x),y)) =_u *(u,\ominus(v))$ fails.

Overlap of r_{12} on Position 2 of r_5:

In overlap of r_{12} on $*(x,*(inv(x),y))$, the unification $*(u,\ominus(v)) =_u *(inv(x),y)$ yields

1. $\sigma_1 = \{u \mapsto inv(x), y \mapsto \ominus(v)\}$.

We obtain $\ominus(v)$ and $*(x,\ominus(*(inv(x),v)))$, where the latter is reducible to $\ominus(v)$.

In overlap of r_{12} on $*(inv(x),*(x,y))$, the unification $*(u,\ominus(v)) =_u *(x,y)$ yields

1. $\sigma_1 = \{u \mapsto x, y \mapsto \ominus(v)\}$.

We obtain $\ominus(v)$ and $*(inv(x),\ominus(*(x,v)))$, where the latter is reducible to $\ominus(v)$.

Equational Overlap:

1-2 contexts: The unification $*(v_1,*(x,*(inv(x),y))) =_u *(v_2,*(v_3,*(u,\ominus(v))))$, where $v_1 \in \{u\}$ and $\{v_2,v_3\} = \{x,inv(x)\}$ hold, yields one case (modulo the switch of v_2 and v_3)

1. $\sigma_1 = \{v_1 \mapsto u, v_2 \mapsto x, v_3 \mapsto inv(x), y \mapsto \ominus(v)\}$.

Here, we obtain $*(u,\ominus(v))$ and $*(x,*(inv(x),\ominus(*(u,v))))$, and these are reducible to the term $\ominus(*(u,v))$.

1-1 contexts: The unification $*(v_1,*(x,*(inv(x),y))) =_u *(v_2,*(u,\ominus(v)))$ fails.

A.3.5 Superposition of r_7 with r_{11}:

$r_7 : *(x,\oplus(y,z)) \to \oplus(*(x,y),*(x,z))$ and $r_{11} : *(u,\infty) \to \infty$

Overlap at Top Position:

The unification $*(x,\oplus(y,z)) =_u *(u,\infty)$ fails.

Equational Overlap:

1-1 contexts: The unification $*(v_1,*(x,\oplus(y,z))) =_u *(v_2,*(u,\infty))$ fails.

Superposition of r_7 with r_{12}:

$r_7 : *(x,\oplus(y,z)) \to \oplus(*(x,y),*(x,z))$ and $r_{12} : *(u,\ominus(v)) \to \ominus(*(u,v))$

Overlap at Top Position:

The unification $*(x,\oplus(y,z)) =_u *(u,\ominus(v))$ fails.

Equational Overlap:

1-1 contexts: The unification $*(v_1,*(x,\oplus(y,z))) =_u *(v_2,*(u,\ominus(v)))$ fails.

A.3.6 Superposition of r_{11} with r_{12}:

$r_{11} : *(x,\infty) \to \infty$ and $r_{12} : *(u,\ominus(v)) \to \ominus(*(u,v))$

Overlap at Top Position:

The unification $*(x,\infty) =_u *(u,\ominus(v))$ fails.

Equational Overlap:

1-1 contexts: The unification $*(v_1,*(x,\infty)) =_u *(v_2,*(u,\ominus(v)))$ fails.

A.4 Overlaps of $\oplus$-terms

Referring to [39], we have a complete modular rewriting system $R'/A_\oplus$ where $R' = \{r_3,r_4,r_8,r_9,r_10\}$ and $A_\oplus$ corresponds to commutativity and associativity of $\oplus$. For that reason, we investigate only the overlaps of r_3 and r_4 on r_7.

A.4.1 Superposition of r_3 on r_7:

$r_3 : \oplus(x,\ominus(x)) \to \infty$ and $r_7 : *(u,\oplus(v,w)) \to \oplus(*(u,v),*(u,w))$

Overlap on Position 2 (without extension):

The unification $\oplus(x,\ominus(x)) =_u \oplus(v,w)$ (excluding redundancy by distinction of v and w) decomposes to:

1. $\{v =_u x, w =_u \ominus(x)\}$
   This yields $\sigma_1 = \{v \mapsto x, w \mapsto \ominus(x)\}$.
2. $\{v =_u \oplus(\ominus(x),v_1), x =_u \oplus(v_1,w)\}$
   This yields $\sigma_2 = \{v \mapsto \oplus(\ominus(\oplus(v_1,w)),v_1), x \mapsto \oplus(v_1,w)\}$.

In case (1), we obtain $*(u,\infty)$ and $\oplus(*(u,x),*(u,\ominus(x)))$, which are reducible to ∞.
In case (2), we obtain $*(u,\infty)$ and $\oplus(*(u,\oplus(\ominus(\oplus(v_1,w)),v_1)),*(u,w))$. The former reduces to ∞ and the latter reduces to $\oplus(\oplus(\oplus(\ominus(*(u,v_1)),\ominus(*(u,w))),*(u,v_1)),*(u,w))$ and finally to ∞.

Overlap on Position 2 (with extension):

The unification $\oplus(v_1,\oplus(x,\ominus(x))) =_u \oplus(v,w)$ (excluding redundancy by distinction of v and w and excluding the cases where $v_1 \in \{v,w\}$) decomposes to:

1. $\{v =_u \oplus(\ominus(x),v_1), w =_u x\}$
   This yields $\sigma_1 = \{v \mapsto \oplus(\ominus(x),v_1), w \mapsto x\}$.
2. $\{v =_u \oplus(\ominus(x),v_2), v_1 =_u \oplus(v_2,v_3), w =_u \oplus(x,v_3)\}$
   This yields $\sigma_2 = \{v \mapsto \oplus(\ominus(x),v_2), v_1 \mapsto \oplus(v_2,v_3), w \mapsto \oplus(x,v_3)\}$.

In case (1), we obtain $*(u,\oplus(v_1,\infty))$ and $\oplus(*(u,\oplus(\ominus(x),v_1)),*(u,x))$. The former reduces to $*(u,v_1)$; The latter reduces to $\oplus(\oplus(\ominus(*(u,x)),*(u,v_1)),*(u,x))$ and finally to $*(u,v_1)$.
In case (2), we obtain $*(u,\oplus(\oplus(v_2,v_3),\infty))$ and $\oplus(*(u,\oplus(\ominus(x),v_2)),*(u,\oplus(x,v_3)))$, on the other side. The former reduces to the term $\oplus(*(u,v_2),*(u,v_3))$; The latter reduces to the term $\oplus(\oplus(\ominus(*(u,x)),*(u,v_2)),\oplus(*(u,x),*(u,v_3)))$ and finally to $\oplus(*(u,v_2),*(u,v_3))$.

A.4.2 Superposition of r_4 on r_7:

$r_4 : \oplus(x,\infty) \to x$ and $r_7 : *(u,\oplus(v,w)) \to \oplus(*(u,v),*(u,w))$

Overlap on Position 2 (without extension):

The unification $\oplus(x,\infty) =_u \oplus(v,w)$ (excluding redundancy by distinction of v and w) decomposes to:

1. $\{v =_u x, w =_u \infty\}$
   This yields $\sigma_1 = \{v \mapsto x, w \mapsto \infty\}$.

2. $\{v =_u \oplus(\infty,v_1), x =_u \oplus(v_1,w)\}$
   This yields $\sigma_2 = \{v \mapsto \oplus(\infty,v_1), x \mapsto \oplus(v_1,w)\}$.

In case (1), we obtain $*(u,x)$ and $\oplus(*(u,x),*(u,\infty))$, where the latter is reducible to $*(u,x)$.
In case (2), we obtain $*(u,\oplus(v_1,w))$ and $\oplus(*(u,\oplus(\infty,v_1)),*(u,w))$. The former reduces to the term $\oplus(*(u,v_1),*(u,w))$ and the latter reduces to $\oplus(\oplus(*(u,\infty),*(u,v_1)),*(u,w))$ and finally to $\oplus(*(u,v_1),*(u,w))$.

Overlap on Position 2 (with extension):

The unification $\oplus(v_1,\oplus(x,\infty)) =_u \oplus(v,w)$ (excluding redundancy by distinction of v and w and excluding the cases where $v_1 \in \{v,w\}$) decomposes to:

1. $\{v =_u \oplus(\infty,v_1), w =_u x\}$
   This yields $\sigma_1 = \{v \mapsto \oplus(\infty,v_1), w \mapsto x\}$.

2. $\{v =_u \oplus(\infty,v_2), v_1 =_u \oplus(v_2,v_3), w =_u \oplus(x,v_3)\}$
   This yields $\sigma_2 = \{v \mapsto \oplus(\infty,v_2), v_1 \mapsto \oplus(v_2,v_3), w \mapsto \oplus(x,v_3)\}$.

In case (1), we obtain $*(u,\oplus(v_1,x))$ and $\oplus(*(u,\oplus(\infty,v_1)),*(u,x))$. The former reduces to the term $\oplus(*(u,v_1),*(u,x))$; The latter reduces to $\oplus(\oplus(*(u,\infty),*(u,v_1)),*(u,x))$ and finally to $\oplus(*(u,v_1),*(u,x))$.
In case (2), we obtain the terms $*(u,\oplus(\oplus(v_2,v_3),x))$ and $\oplus(*(u,\oplus(\infty,v_2)),*(u,\oplus(x,v_3)))$. The former reduces to the term $\oplus(\oplus(*(u,v_2),*(u,v_3)),*(u,x))$; The latter reduces to $\oplus(\oplus(*(u,\infty),*(u,v_2)),\oplus(*(u,x),*(u,v_3)))$ and finally to $\oplus(*(u,v_2),\oplus(*(u,x),*(u,v_3)))$.