UNIVERSITÄT
DES
SAARLANDES

Relating Space and Time in Cryptography

Jesko Dujmović

Dissertation
zur Erlangung des Grades
des Doktors der Naturwissenschaften
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

Saarbrücken, 2025

Tag des Kolloquiums:	10. Oktober 2025	
Dekan:	Prof. Dr. Roland Speicher	
Prüfungsausschuss:		
	Vorsitzende:	Prof. Dr. Martina Maggio
	Berichterstattende:	Dr. Nico Döttling
		Prof. Dr. Abhiskek Jain
		Prof. Dr. Markus Bläser
	Akademischer Mitarbeiter:	Dr. Hendrik Waldner

Abstract

This thesis studies the interplay between space and time in cryptography. It explores trade-offs between these two resources for honest parties and shows how fine-grained constraints on an adversary's capabilities can unlock new cryptographic functionalities. Adopting this perspective, we present results in four subfields of cryptography:

2PC: We prove tight lower bounds on the cost of oblivious transfer protocols, a central primitive for two-party computation, revealing a fundamental trade-off between communication and public-key operations.

SNARG: We construct designated-verifier SNARGs with very short proofs, highlighting a novel trade-off between proof size and verifier efficiency.

Incompressible Encryption: We construct a new incompressible encryption scheme. Incompressible encryption remains secure even after key exposure, assuming adversaries had limited space when the ciphertext was transmitted. This offers a lightweight alternative to forward secrecy for long messages.

Space-Hard Functions: We define and construct verifiable space-hard functions and space-lock puzzles, space-based analogues of verifiable delay functions and time-lock puzzles. These enable new applications such as deniable proofs and leverage space limitations of the adversary.

Together, these results demonstrate how a careful study of space-time trade-offs yields both foundational insights and practical cryptographic tools.

Zusammenfassung

Diese Arbeit untersucht das Zusammenspiel von Raum und Zeit in der Kryptographie. Sie analysiert Kompromisse zwischen diesen Ressourcen für ehrliche Parteien und zeigt, wie gezielte Beschränkungen der Fähigkeiten eines Angreifers neue Möglichkeiten eröffnen. Aus dieser Sicht liefern wir Beiträge in vier Bereichen:

2PC: Wir beweisen untere Schranken für die Kosten von Oblivious-Transfer-Protokollen, einem zentralen Primitive für Zwei-Parteien-Berechnung, und zeigen einen grundlegenden Trade-off zwischen Kommunikation und Public-Key-Operationen.

SNARG: Wir konstruieren Designated-Verifier-SNARGs mit sehr kurzen Beweisen und demonstrieren einen neuen Trade-off zwischen Beweisgröße und Verifizierer-Effizienz.

Inkomprimierbare Verschlüsselung: Wir stellen ein neues Schema vor, das auch nach Schlüsselkompromittierung sicher bleibt, solange der Angreifer beim Empfang nur begrenzten Raum hatte. Es bietet eine leichte Alternative zur Vorwärtsgeheimhaltung bei langen Nachrichten.

Raum-Harte Funktionen: Wir definieren und konstruieren die ersten verifizierbaren raumharte Funktionen und Raum-Lock-Rätsel, raumbasierte Gegenstücke zu Time-Lock-Rätseln. Diese ermöglichen neue Anwendungen wie ableugbare Beweise und nutzen Raumbeschränkung des Angreifers.

Diese Resultate zeigen, dass eine gezielte Analyse Kompromissen zwischen Raum und Zeit sowohl theoretische Einsichten als auch praxisnahe Werkzeuge liefert.

Acknowledgments

First and foremost, I would like to thank my advisor, Nico Döttling. Your passion for research has been a constant source of inspiration. The time I spent during my PhD has profoundly shaped my life, and I am deeply grateful for your guidance and support.

I also want to thank Rachit Garg and Gal Arnon—you quickly went from conference acquaintances to friends and collaborators. I was very fortunate to have the opportunity to visit Giulio Malavolta and Eylon Yogev for an internship and a scientific visit, respectively. I learned a great deal about how to do research from both of you.

I'm grateful to my other collaborators as well: Mohammad Hajiabadi, Maciej Obremski, Divesh Aggarwal, Pedro Branco, Rachit Garg, Antoine Joux, Julian Loss, Gal Arnon, Yuval Ishai, Wei Qi, Christoph Günther, and Krzysztof Pietrzak. Our thought-provoking discussions and shared struggles with challenging problems made the collaborations not only intellectually rewarding but also personally enjoyable. It was a true pleasure to work with each of you.

My gratitude extends beyond my collaborators. Working in cryptography has been an extraordinary experience—not only because of the fascinating research, but also thanks to the vibrant community behind it. I'm thankful to everyone who has helped shape and nurture the field into what it is today.

I'm especially grateful for the warm and welcoming environment within the CISPA crypto groups. Special thanks to Giacomo Santato, Riccardo Zanotto, Willy Quach, Benedikt Wagner, Anne Müller, and Eugenio Paracucci, who brought camaraderie to the long days, especially during intense deadline seasons. You made the office feel like a second home.

To my partner, K—your presence has brought immense joy and brightness into my life. To my close friend, N—thank you for consistently going to the gym with me. The routine has had a profoundly positive impact. D, I always look forward to our climbing and yapping sessions, and D and K, thank you for being such great friends and roommates. Together, you made the hard times bearable and the good times unforgettable.

Lastly, I want to thank my loving family for providing such an amazing environment to grow up in. My parents and my brother are a true joy to be around. I especially want to acknowledge my grandfather, Borna, to whom I dedicate this thesis. He has been a great inspiration to me and taught me many life lessons through his example.

Contributions

This thesis is comprised of four parts, each corresponding to one of the following papers, each of which I was the main author of:

[DH24]: Lower-Bounds on Public-Key Operations in PIR
Jesko Dujmovic, Mohammad Hajiabadi
Eurocrypt 2024

[ADI25]: Designated-Verifier SNARGs with One Group Element
Gal Arnon, Jesko Dujmovic, Yuval Ishai
Crypto 2025

[BDD22]: Rate-1 Incompressible Encryption from Standard Assumptions
Pedro Branco, Nico Döttling, Jesko Dujmovic
Theory of Cryptography Conference 2022

[DDJ24]: Space-Lock Puzzles and Verifiable Space-Hard Functions from Root-Finding in Sparse Polynomials
Nico Döttling, Jesko Dujmovic, Antoine Joux
Theory of Cryptography Conference 2024

For [DH24, ADI25, BDD22] I am the main author and significantly contributed to all aspects of the work. These works are unchanged up to minor modifications. [DDJ24] contains sections, to which I didn't contribute significantly, which is why these sections are not contained within the thesis.

During my time as a PhD student, I was also a main author of the following papers:

[DD22]: Maliciously Circuit-Private FHE from Information-Theoretic Principles
Nico Döttling, Jesko Dujmovic
Information Theoretic Cryptography 2022

[DGM24]: Time-Lock Puzzles with Efficient Batch Solving
Jesko Dujmovic, Rachit Garg, Giulio Malavolta
Eurocrypt 2024

[DDLO25]: Minicrypt PIR for Big Batches
Nico Döttling, Jesko Dujmovic, Julian Loss, Maciej Obremski
Under Submission

[DMQ24]: Registration-Based Encryption in the Plain Model
Jesko Dujmovic, Giulio Malavolta, Wei Qi
Public Key Cryptography 2025

[DGP25]: Space-Deniable Proofs
Jesko Dujmovic, Christoph U. Günther, Krzysztof Pietrzak
Theory of Cryptography Conference 2025

Furthermore, I significantly contributed to the paper:

[ADD+22]: Algebraic Restriction Codes and their Applications
Divesh Aggarwal, Nico Döttling, Jesko Dujmovic, Mohammad Hajiabadi, Giulio Malavolta, Maciej Obremski
Innovations in Theoretical Computer Science 2022 and Algorithmica 2023

Contents

Chapter 1

Introduction

Cryptography is the study of designing protocols that are secure against adversarial behaviour. Crucial to this goal are precise definitions of what it means to be secure and which resources the adversary has access to. We aim to minimize the resources required by honest parties while maximizing the resources an adversary must invest in compromising security.

In theoretical cryptography, many of these resources are traditionally modelled qualitatively. For example, a common question in the security of an encryption scheme is whether the adversary has access to a decryption oracle.

In this thesis, we instead adopt a quantitative perspective, focusing on fine-grained resource bounds. Among the most important quantitative resources in cryptography are the running time of algorithms, the memory usage of the parties, and the amount of communication required by a protocol. We group these into two fundamental categories: time (running time) and space (memory and communication).

This perspective motivates the central theme of this thesis: the relationship between space and time in cryptographic protocols. We investigate how these resources interact and demonstrate that, for specific protocols, there is an inherent trade-off between communication and computation. In some cases, we uncover previously unknown trade-offs. In others, we design protocols that leverage these trade-offs, made possible only through careful quantitative modelling of the adversary's capabilities.

Communication-Computation Trade-Offs in 2PC

The first topic of this thesis concerns secure two-party computation (2PC). In 2PC, two parties, typically referred to as Alice and Bob, jointly aim to compute a function f on their respective inputs a and b. At the end of the protocol, Alice should learn only the value $f(a, b)$, while Bob should learn nothing about Alice's input.

A central 2PC primitive is oblivious transfer (OT). In OT, Bob's input consists of two messages m_0 and m_1, and Alice's input is a choice bit $b \in \{0, 1\}$. At the end of the protocol, Alice learns m_b, but nothing about m_{1-b}, and Bob stays oblivious to which of its two messages was transferred. OT occupies this central role due to two key properties:

1. OT is relatively easy to construct due to its simplicity.

2. Given a protocol for OT, there are ways to construct a 2PC protocol for evaluating any Boolean circuit f [Yao82, Yao86].

The two primary resources relevant to OT protocols are communication and computation. Significant progress has been made in optimizing each of these individually: some constructions minimize the amount of data exchanged, while others focus on reducing local computational effort.

Optimizing Communication To evaluate the communication efficiency of OT protocols, it is instructive to compare them to insecure baselines that implement the same functionality.

There are two natural approaches: Bob can send his choice bit b, and Alice replies with m_b, resulting in 1 bit of Bob-to-Alice communication and $|m_0|$ bits from Alice to Bob. Alternatively, Alice can send both messages (m_0, m_1), requiring $2|m_0|$ bits of communication from Alice and no message from Bob. In both cases, the total communication is at least $|m_0| + 1$ bits when $|m_0| = |m_1|$.

For long messages, there has been significant progress in designing protocols that communicate $|m_0| + o(|m_0|)$ bits [IP07, DGI+19, GHO20, CGH+21, ADD+22]. Even more impressively, protocols now exist that perform n parallel instances of OT on single-bit messages with just $2n + o(n)$ bits of total communication [BDGM19, BBDP22, BDS23]. These protocols achieve near-optimal communication, but their main drawback is their substantial computational cost: they require numerous public-key operations.

Optimizing Computation To understand the direction another line of work took to make OT concretely fast, we will have to detour through the foundation of cryptography.

Cryptographic constructions are often divided into two very broad classes: *symmetric-key* cryptography, which includes all tools that can be built from random-like functions, and *public-key* cryptography, which enables tasks such as key agreement and requires more structure than a random function. Symmetric-key cryptography is generally much faster in practice, as it can leverage unstructured primitives optimized for efficient implementation. In contrast, public-key cryptography tends to be much slower due to reliance on structured mathematical problems that are inherently harder to implement efficiently. For example, the most popular symmetric-key encryption AES can be a factor of 1000 faster than one of the most popular public-key encryption schemes, RSA [RSA78]. Famously, Impagliazzo and Rudich [IR89] proved that one can not build key agreement from unstructured functions in a black box way.

Oblivious transfer belongs to the class of public-key primitives, as it can be used to implement key agreement. Consequently, it is typically expensive to implement directly. Since Yao's 2PC protocol requires a large number of OT instances, this cost becomes a severe bottleneck. To address this, a powerful technique known as *OT extension* was developed [Bea96, IKNP03]. The idea is to reduce the number of expensive public-key operations by generating many OT instances from a small number of base OTs, using only symmetric-key techniques for the bulk of the computation.

However, all known OT extension protocols trade faster evaluation against more communication, leading to the following natural question:

> *Are there communication-efficient OT protocols with very few public-key operations?*

In Chapter 3, we answer this question negatively. We relate OT to a primitive called private information retrieval and prove that the space-time trade-off between communication and computation (i.e. space and time) in private information retrieval is inherent.

Designated-Verifier SNARGs

One of cryptography's major success stories is the development of proof systems. Proof systems allow a prover to convince a verifier of the validity of a statement x. Their study has led to influential advances in theoretical computer science, including interactive proofs (IP) and probabilistically checkable proofs (PCP).

In IPs [GMR89], a prover and an efficient verifier interact until the verifier is convinced of the statement x. Famously, it is possible to prove statements from PSPACE languages interactively [Sha90]. These are languages for which solutions can be computed using polynomial space. Without interaction, only statements in NP can be proven, languages that can be efficiently verified.

In the same work, Goldwasser et al. [GMR89] introduced the concept of zero-knowledge proofs. Zero-knowledge enables a prover to convince a verifier of an NP statement without revealing any information about the witness. More precisely, it means that a simulator without access to the witness can produce a transcript that is computationally indistinguishable from one generated by an prover-verifier interaction.

In PCPs [BFLS91, FGM$^+$89], In contrast, the prover encodes the statement x into a long string, and the verifier only reads a few positions to check its validity. This line of research led to the PCP theorem [AS92, ALM$^+$92], which states that any NP statement can be encoded into a string of polynomial length; the verifier only reads a constant number of positions and is convinced of the validity of the statement with a constant probability. PCPs have influence far beyond proof systems, like establishing the hardness of approximation problems.

Inspired by these results and the work of Kilian [Kil92], Micali [Mic94] explored the question of an efficient prover with access to a witness w for an NP statement x can convince an efficient verifier while sending one message, which is often called a proof whose size is polylogarithmic in the size of the witness w. In this setting, however, we can only guarantee soundness against a computationally bounded prover. For this reason, it is called an argument system instead of a proof system. Such a primitive is known as a succinct non-interactive argument (SNARG). A popular variant of SNARGs requires proving that the prover knows a witness w for the statement instead of just proving the validity of the statement. These argument systems are then called succinct non-interactive arguments of knowledge (SNARK).

Previous SNARGs Kilian [Kil92] built a three-move interactive proof system using the Merkle commitments and PCPs. Micali [Mic94] then turned this protocol non-interactive using the Fiat-Shamir transform. Since then a lot of effort has gone into optimizing this approach of building SNARGs [BCS16, RRR16, BBHR18, ACFY24b] resulting in argument systems with excellent prover and verifier runtime and proofs of size > 40 KiB for a reasonable security level[1].

The other major way of constructing practical SNARGs is based on combining linear PCPs with cryptographic objects which work well with linearity [IKO07, BCI$^+$13, GGPR13, DFGK14, Gro16, BBB$^+$18, Lip24]. In linear PCPs, the proof is a linear function that is evaluated on the verifier's query. While there are different trade-offs, these proof systems generally have higher proving and verification costs and rely on pairing groups but can reach much smaller sizes. The smallest of these, currently, is a system called "Pari" by [DMS24], who reach a size of 1280 bits for a reasonable security level.

[1] We defer a discussion of what security level precisely means to later sections.

Dv-SNARGs In this work, we explore a closely related object called designated-verifier SNARGs (dv-SNARGs/dv-SNARKs). These are argument systems where the verifier first sends a message to the prover that is independent of the statement x, and the prover responds with a short proof. Any SNARG can be seen as a dv-SNARG in which the verifier's initial message is empty.

Dv-SNARKs have many applications, like verifying delegated computation. Here, we highlight applications of zero-knowledge dv-SNARKs to E-cash [Cha82]. The goal of E-cash is to maintain a balance at a bank while ensuring that spending cannot be linked to the owner. For this application, we require the dv-SNARK to be zero knowledge. Dv-SNARKs can be modified to also have zero knowledge by composing them with a non-interactive proof system that has zero knowledge.

Using zero-knowledge dv-SNARGs, there is a very flexible E-cash protocol with very little communication. In this protocol, the user commits to random strings and sends the commitment to the bank. The bank signs the commitment. To retrieve some of the balance from the bank, the user then proves to the bank using a zero-knowledge dv-SNARK. That they know a signed commitment and open an opening to that commitment.

Previous Dv-SNARGs The works of [BCI$^+$13, BIOW20] lay out an approach to how to combine linearly homomorphic encryption schemes with variants of linear PCPs to construct dv-SNARGs/dv-SNARKs. Barta et al. [BIOW20] manage to construct a dv-SNARG that has a size of 512 bits while not having to rely on pairing groups. A major downside of their construction is a relatively high soundness error: a malicious prover can convince the verifier of a false statement with probability $1/\mathsf{poly}$.

These are the smallest known SNARGs not relying on indistinguishability obfuscation (iO). Sahai and Waters [SW14] construct a SNARG with 128-bit sized proofs based on iO. This construction, however, is very far from practical.

This state of affairs poses two natural questions:

> *Are there group-based dv-SNARKs that are smaller than known publicly verifiable SNARGs without significant soundness error?*

and

> *Are there group-based dv-SNARKs much smaller than 512 bits, even with significant soundness error?*

Our Result We answer both of these questions positively. We provide a construction of a group-based dv-SNARG, which has a low soundness error and a size of 695 bits. We also construct a group-based dv-SNARG with constant soundness error and only 263 bits. In both regimes, our constructions reduce the proof size by approximately a factor of two compared to the smallest known prior works.

Verification vs. Proof Size This reduction in proof size does not come for free. The designated verifier must perform relatively heavy computation to verify the proof. All SNARG constructions we compare against, but Barta et al. [BIOW20] have a verifier that only needs to do very little computation after receiving the proof. We show a new trade-off between space and time in the SNARG design space.

Incompressible Encryption

In the second part of this thesis, we show that by simultaneously restricting an adversary's time and space resources, it is possible to give security guarantees that would be impossible under a limitation on just one resource alone.

A major goal of cryptography is to ensure privacy and integrity of communication. The main tool to ensure privacy in communication is encryption. Encryption allows one party to encrypt a message into a ciphertext. Now, it should be close to impossible for an adversary to determine the encrypted message. However, a party with the secret key for this ciphertext, the decryptor, should easily be able to retrieve the correct message.

Because encryption is such an important primitive, we want encryption schemes to be as secure as possible. Significant effort has gone into extending the security guarantees of encryption schemes in various directions. This helps secure communication against powerful adversaries in more situations. The most notable examples that go beyond semantic security are CCA security and post-quantum security. In this vein, we consider the following question

> *Can an encryption scheme be secure against an adversary that has the secret key, just like the decryptor?*

If the adversary has unrestricted access to the ciphertext and the secret key, it can just decrypt; therefore, the answer is: *"No!"*. But indeed, incompressible encryption [Dzi06b, GWZ22, BDD22, GWZ23] can guarantee some security even if the adversary learns the key. If the adversary is restricted from ever being able to access the *entire* ciphertext and the secret key simultaneously, security may hold.

Instead of having the ciphertext and the key simultaneously, the adversary first has the ciphertext and has to compress that ciphertext by just a little bit. Then, based on the compressed ciphertext and the secret key, the adversary wants to learn some information about the encrypted messages. We call an encryption scheme incompressible if it is secure even against these kinds of adversaries.

Incompressible encryption increases security guarantees by enforcing a space and time bound on the adversary. If we don't restrict one of the two resources in this setting, any security is lost.

Prior to our work, the best-known constructions of the incompressible encryption scheme either required long ciphertexts encrypting little amounts of data or relied on very heavy cryptographic machinery, namely indistinguishability obfuscation [GWZ22]. In Chapter 5, we construct incompressible encryption based on basic cryptographic building blocks (a variant of lossy trapdoor functions) while providing the ability to send long messages with very little communication overhead.

Space-Hard Functions

Time-hard cryptography underpins powerful applications, including unbiased randomness generation and fairness enforcement in multi-party protocols [LW15, TCLM21, SLM+23]. These applications provably can only be achieved with a more fine-grained control of the adversary's resources [Cle86, BBCE25]. Motivated by such applications, the cryptographic community has shown increasing interest in time-hard cryptography [RSW96, BGL+15, BDGM19, Wes19, Pie19, SLM+23, DGM24, BG25]. The fundamental building block in this area is the delay function, which requires a fixed amount of wall-clock time to evaluate even for massively parallel adversaries. Built upon delay functions, more advanced primitives have emerged notably, proofs of sequential work, verifiable delay functions, and time-lock puzzles.

While non-algebraic delay functions suffice for proofs of sequential work, practical constructions of verifiable delay functions and time-lock puzzles crucially rely on the delay function's algebraic structure.

Space-hard cryptography [Per09, DFKP15, AS15a, AB16, ACP+17] mirrors the goals of time-hard cryptography but targets an adversary's memory capacity rather than its sequential runtime. The core idea is to design functions that cannot be efficiently computed unless the adversary uses a large amount of memory.

Numerous constructions of memory-hard functions and proofs of space have been proposed [2] which serve as analogues to delay functions and proofs of sequential work, respectively. Previous to our work, there had not been any efficient construction of verifiable space-hard functions or space-lock puzzles, the primitives corresponding to verifiable delay functions and time-lock puzzles.

Subsequent work [DGP25] to this provides an interesting application to verifiable space-hard functions, namely deniable proofs. Say a user wants to identify themselves with a smartcard to be able to enter a building but doesn't want to leave cryptographic evidence that they entered the building. The solution is to prove to the opening mechanism that the user knows a secret key that identifies themselves or that they used a lot of space. Because the verifier (the opening mechanism) knows through physical restrictions that the computation was run on a smartcard, it can be sure that the smartcard does not have a lot of space and, therefore, the user is successfully identified. Given the transcript of this interaction, however, the verifier can not convince anyone else that the user entered the building because the verifier has a lot of space and could simulate such a transcript.

The lack of efficient verifiable space-hard functions and space-lock puzzles comes down to a shortage of algebraic structures that provide space-hardness while being easy to work with. This state of affairs raises the question:

> *Is there an algebraic structure that allows for efficient construction of verifiable space-hard functions and space-lock puzzles?*

In this work, we propose an algebraic foundation for space-hard cryptography: the problem of finding roots in sparse polynomials. We conjecture that this problem exhibits strong space-hardness properties and use it to construct, for the first time, verifiable space-hard functions and space-lock puzzles that are both efficient and practical.

This shows that, like time, space can serve as a meaningful and verifiable resource in cryptographic design.

[2] Both these primitives have variants with slightly differing functionality and ways to measure space.

Chapter 2

Preliminaries

2.1 Notation and Basics

We use the Landau notation to describe the asymptotic behavior of functions. We write $f(x) \in O(g(x))$ if there exists a constant c such that $|f(x)| \leq c|g(x)|$ for all x larger than some constant x_0. Further, we write $f(x) \in o(g(x))$ if $\lim_{x\to\infty} f(x)/g(x) = 0$. We denote the security parameter by λ. We say a function negl is negligible if for any polynomial poly we have $\mathsf{negl}(\lambda) \in o(\frac{1}{\mathsf{poly}(\lambda)})$.

The acronym PPT denotes "probabilistic polynomial time". If $\mathcal{A}$ is an algorithm, we denote by $y \leftarrow \mathcal{A}(x)$ the output y after running $\mathcal{A}$ on input x. If S is a (finite) set, we denote by $x \xleftarrow{\$} S$ the experiment of sampling uniformly at random an element x from S. If D is a distribution over S, we denote by $x \xleftarrow{\$} D$ the element x sampled from S according to D.

For two integers m and n, we define $[m, n] = \{m, m+1, \ldots, n\}$. Further, We define let $[n] = [1, n]$. For a vector $\mathbf{a} \in \mathbb{F}^\ell$, we let $\mathbf{a}[i]$ be the i-th entry.

2.1.1 Interactive Protocols

For $i \in \{r, s\}$, denoting receiver (r) and sender (s), we let $\mathsf{view}_i^\Pi(1^\lambda, x, y)$ denote the view of Party i in an honest execution of the protocol Π on 1^λ and on the parties' respective inputs x and y, where the view contains the private input and the random coins of the respective party, the protocol's transcript, and the transcript of oracle queries and their responses. We may omit the security parameter 1^λ whenever it is clear from the context.

2.1.2 Finite Fields

For a prime-power $q = p^k$ we denote the finite field of size q by $\mathbb{F}_q$. We call p the characteristic of $\mathbb{F}_q$. Recall a few basic facts about finite fields. For a field $\mathbb{F}_q$ of characteristic the polynomial functions $x \mapsto x^{p^i}$ are called *Frobenius automorphisms* and it holds that $x^q = x^{p^k} = x$ for all $x \in \mathbb{F}_q$. Hence, the q roots of the polynomial $X^q - X$ are precisely all the elements in $\mathbb{F}_q$. Consequently, if x is in an extension field of $\mathbb{F}_q$ and $x^q - x = 0$, then it must hold that $x \in \mathbb{F}_q$. Likewise, the $q-1$ roots of the polynomial $X^{q-1} - 1$ are exactly all non-zero elements in $\mathbb{F}_q$.

Sometimes equate prime order fields $\mathbb{F}_p$ with the congruence class $\mathbb{Z}_p$. For $a \in \mathbb{Z}_p$ we define the absolute value $|a|$ to be the minimun distance of an element in the congruence class of a to 0, more specifically $|a| = \min\{|b| \mid b \in \mathbb{Z}, b \mod p = a \mod p\}$. Further we define an operation that lifts $a \in \mathbb{Z}_p$ to the integers $\mathbb{Z}(a) = \min_{|b|}\{b \mid b \in \mathbb{Z}, b \mod p = a \mod p\}$ and similarly an operation that moves a to some $\mathbb{Z}_{p'}$, namely $\mathbb{Z}_{p'}(a) = \{b \mid \mathbb{Z}(a) \mod p' = b \mod p'\}$. Notice, if $p' > p$ then for $a \in \mathbb{Z}_p$ we have $a = \mathbb{Z}_p(\mathbb{Z}_{p'}(a))$.

2.1.3 Polynomials

We call the variable for polynomials X and a polynomial is usually denoted like this $f(X)$ or in explicit form, e.g. $X + X^2$. We call a value x a root of a polynomial $f(X)$ if $f(x) = 0$. The degree of a polynomial $f(X)$ is the highest power of X that appears in $f(X)$. A polynomial is called monic if the coefficient of the highest power of X is 1.

Recall that a univariate polynomial $f(X)$ is square-free if and only if it holds that $\gcd(f(X), f'(X)) = 1$, where $f'(X)$ is the formal derivative of $f(X)$ in X.

Lemma 2.1.1 (Bézout's Identity for Polynomials)**.** *Let $f(X)$ and $g(X)$ of degree d_1 and d_2 be two polynomials with greatest common divisor $d(X)$. Then there exist polynomials $a(X)$ and $b(X)$ such that*

$$a(X)f(X) + b(X)g(X) = d(X)$$

and $\deg(a(X)) < d_2$ and $\deg(b(X)) < d_1$. Moreover, for all $\bar{a}(X)$, $\bar{b}(X)$ then polynomials $\bar{a}(X)f(X) + \bar{b}(X)g(X)$ are exactly the multiples of $d(X)$.

Lemma 2.1.2 (Vandermonde Matrix Invertible)**.** *Over any field $\mathbb{F}$ the Vandermonde matrix*

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{pmatrix}$$

is invertible if the x_i are distinct.

2.2 Statististical Notions

We define the statistical distance of two probability distributions X, Y with finite domain D to be $\frac{1}{2}\sum_{\alpha \in D} |\Pr[X = \alpha] - \Pr[Y = \alpha]|$. We say X and Y are statistically indistinguishable if their statistical distance is negligible in the security parameter λ. We use the notation $X \approx_s Y$ to state that the distributions are statistically indistinguishable.

A list of useful definitions and lemmas about statistics follows.

Definition 2.2.1 (Average Min-Entropy [DORS08])**.** For two jointly distributed random variables (X,Y), the average min-entropy of X conditioned on Y is defined as

$$\tilde{H}_\infty(X|Y) = -log(\mathbb{E}_{y \xleftarrow{\$} Y}[max_x \Pr[X = x|Y = y]]).$$

Lemma 2.2.2 (Lemma 2.2 (b) of [DORS08])**.** *For random variables X, Y, Z where Y is supported over a set of size T, we have*

$$\tilde{H}_\infty(X|(Y,Z)) \geq \tilde{H}_\infty((X,Y)|Z) - log(T) \geq \tilde{H}_\infty(X|Z) - log(T).$$

Definition 2.2.3 (Average-Case Extractor [DORS08])**.** Let $n, d, m \in \mathbb{N}$. A function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ϵ) strong average-case min-entropy extractor if, for all random variables (X, Y) where X takes values in $\{0,1\}^n$ and $\tilde{H}_\infty(X|Y) \geq k$, we have that $(U_d, \mathsf{Ext}(X, U_d), Y)$ is ϵ-close to (U_d, U_m, Y), where U_d and U_m are independent uniformly random strings of length d and m respectively.

Lemma 2.2.4 (Generalized Leftover Hash Lemma 2.4 of [DORS08])**.** *Let $n, m \in \mathbb{N}$. Let $\{H_r : \{0,1\}^n \to \{0,1\}^m\}_{r \in R}$ be a family of universal hash functions, then $\mathsf{Ext}(x, r) \mapsto H_r(x)$ is an average-case (k, ϵ)-strong extractor whenever $m \leq k - 2log(\frac{1}{\epsilon}) + 2$.*

Lemma 2.2.5 (Hoeffding's Inquality)**.** *Let $X_1, \ldots, X_n$ be independent random variables where $X_i \in [-B, B]$ for $B > 0$ and let $X = \sum_i X_i$. Then for every $t > 0$,*

$$\Pr[|X - \mathbb{E}[X]| > t] < 2 \cdot \exp\left(-\frac{t^2}{2 \cdot n \cdot B^2}\right) .$$

Definition 2.2.6 (Universal Hash Functions [WC81])**.** H is family of strongly universal$_t$ hash functions that map from X to Y if for any distict $x_1, \ldots, x_t \in X$, and any possibly non-distinct $y_1, \ldots, y_t \in Y$, we have that

$$\Pr_{h \xleftarrow{\$} H}[h(x_1) = y_1, \ldots, h(x_t) = y_t] = |Y|^{-t}.$$

We describe the first two Bonferroni inequalities, where the first one is commonly referred to as the union bound.

Lemma 2.2.7 (Bonferroni Inequalities [Bon36])**.** *Let $A_1, \ldots, A_n$ be events. We have $\sum_{i \in [n]} \Pr[A_i] - \sum_{i \in [n], j \in [i-1]} \Pr[A_i \cap A_j] \leq \Pr[\cup_{i \in [n]} A_i] \leq \sum_{i \in [n]} \Pr[A_i]$.*

Lemma 2.2.8 (Polynomial Identity Lemma [Sch80, Zip79, DL78])**.** *Let $f(X)$ be a polynomial of degree d over a field $\mathbb{F}_q$. Let $S \subseteq \mathbb{F}_q$ be a set of size s. Then for a random $x \xleftarrow{\$} S$ we have $f(x) = 0$ with probability at most d/s.*

2.3 Proof Systems

Throughout the document we use different proof systems. We introduce these systems in the following section.

In our setting we are mostly concerned with proof systems for NP languages. We define these languages via a relation $\mathcal{R}$. This relation is a set of instance-witness pairs $\{(x, w)\}$ such that the NP verifier of these languages accepts the input (x, w). For a relation $\mathcal{R} := \{(x, w)\}$, we let $\mathcal{L}(\mathcal{R}) := \{x \mid \exists w,\ (x, w) \in \mathcal{R}\}$.

2.3.1 Linear PCPs and Strong Linear MIPs

Probabilistically checkable proofs (PCPs) [BFLS91, FGL$^+$96] are an extremely powerful tool from theoretical computer science. They have been very useful in proving inapproximability results and lead to the first succinct non-inteactive arguments (SNARGs). More on that topic later. Here we consider a variant thereof; linear PCPs.

A linear PCP is a proof system where a proof is a (affine) linear function on a verifier's queries. In standard PCPs the function is arbitrary.

Definition 2.3.1. A linear PCP $(\mathbf{P}, (V_Q, V_D))$ for a relation $\mathcal{R} = \{(x, w)\}$ over a finite field $\mathbb{F}$ is defined as follows:

- *Syntax.* We describe a linear PCP with input length n, proof length ℓ, and query complexity q:
  - The verifier query algorithm V_Q receives as input $x \in \mathbb{F}^n$. It outputs a state $\mathsf{st} \in \{0,1\}^*$, and q queries $\mathbf{a}_1, \ldots, \mathbf{a}_q \in \mathbb{F}^\ell$.
  - The (honest) prover algorithm $\mathbf{P}$ receives an input $x \in \mathbb{F}^n$ and witness $w \in \mathbb{F}^h$. It outputs a proof $\pi \in \mathbb{F}^\ell$.
  - The verifier decision algorithm V_D receives as input a state $\mathsf{st} \in \{0,1\}^*$, an input $x \in \mathbb{F}^n$, and query answers $b_1, \ldots, b_q \in \mathbb{F}$. It outputs a bit $b \in \{0,1\}$.
- *Perfect completeness.* A linear PCP has perfect completeness if for all $(x, w) \in \mathcal{R}$:

$$\Pr\left[V_D\left(\mathsf{st}, x, b_1, \ldots, b_q\right) = 1 \;\middle|\; \begin{array}{r} \pi \leftarrow \mathbf{P}(x, w) \\ (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ \forall i \in [q],\ b_i = \langle \pi, \mathbf{a}_i \rangle \end{array}\right] = 1 \ .$$

- *Soundness.* A linear PCP has soundness error (against affine strategies) δ if for every $x \notin \mathcal{L}(\mathcal{R})$, $\pi \in \mathbb{F}^\ell$, and $c_1, \ldots, c_q \in \mathbb{F}$:

$$\Pr\left[V_D\left(\mathsf{st}, x, b_1, \ldots, b_q\right) = 1 \;\middle|\; \begin{array}{c} (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ \forall i \in [q],\ b_i = \langle \pi, \mathbf{a}_i \rangle + c_i \end{array}\right] \leq \delta \ .$$

- *Knowledge.* A linear PCP satisfies knowledge soundness κ if there exists a PPT extractor Ext such that for every x, $\pi \in \mathbb{F}^\ell$, and $c_1, \ldots, c_q \in \mathbb{F}$ if,

$$\Pr\left[V_D\left(\mathsf{st}, x, b_1, \ldots, b_q\right) = 1 \;\middle|\; \begin{array}{c} (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ \forall i \in [q],\ b_i = \langle \pi, \mathbf{a}_i \rangle + c_i \end{array}\right] > \kappa \ ,$$

  then $(x, \mathsf{Ext}(x, \pi, c_1, \ldots, c_q)) \in \mathcal{R}$.

We say that a linear PCP is **smooth** if every query $\mathbf{a}_i$ is 1-wise uniform (i.e. it is uniform when only looking at one query at a time) over $\mathbb{F}^\ell$, and we say that it is **instance-independent** if $V_Q(x)$ is a function only in the size of the instance x, in which case we specify the verifiers input by $1^{|x|}$.

We additionally consider a strong variant of linear multiple prover interactive proofs (MIPs), where the honest prover is restricted to a single linear function, while the malicious adversary can reply to any query with an arbitrary (stateless) function. The name is inspired by the fact that in the malicious case one can think of the verifier sending the queries to different non-communicating provers.

Definition 2.3.2. A strong linear MIP $(\mathbf{P}, (V_Q, V_D))$ for a relation $\mathcal{R} = \{(x, w)\}$ over a finite field $\mathbb{F}$ is defined as follows:

- *Syntax.* We describe a linear PCP with input length n, proof length ℓ, and query complexity q:
  - The verifier query algorithm V_Q receives as input $x \in \mathbb{F}^n$. It outputs a state $\mathsf{st} \in \{0,1\}^*$, and q queries $\mathbf{a}_1, \ldots, \mathbf{a}_q \in \mathbb{F}^\ell$.

- The (honest) prover algorithm $\mathbf{P}$ receives an input $x \in \mathbb{F}^n$ and witness $w \in \mathbb{F}^h$. It outputs a proof $\pi \in \mathbb{F}^\ell$.
- The verifier decision algorithm V_D receives as input a state $\mathsf{st} \in \{0,1\}^*$, an input $x \in \mathbb{F}^n$, and query answers $b_1, \ldots, b_q \in \mathbb{F}$. It outputs a bit $b \in \{0,1\}$.

- *Perfect completeness.* A linear PCP has perfect completeness if for all $(x, w) \in \mathcal{R}$:

$$\Pr\left[V_D\left(\mathsf{st}, x, b_1, \ldots, b_q\right) = 1 \;\middle|\; \begin{array}{r} \pi \leftarrow \mathbf{P}(x, w) \\ (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ \forall i \in [q],\ b_i = \langle \pi, \mathbf{a}_i \rangle \end{array}\right] = 1 \ .$$

- *Soundness.* A q-query strong linear MIP has soundness error δ if for every $x \notin \mathcal{L}(\mathcal{R})$, and functions $f_1, \ldots, f_q$:

$$\Pr\left[V_D\left(\mathsf{st}, x, b_1, \ldots, b_q\right) = 1 \;\middle|\; \begin{array}{r} (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \end{array}\right] \leq \delta \ .$$

- *Knowledge.* A q-query strong linear MIP has knowledge soundness κ if there exists a an expected polynomial-time oracle-aided extractor Ext such that for every x and every set of functions $f_1, \ldots, f_q$ if

$$\Pr\left[V_D(\mathsf{st}, x, b_1, \ldots, b_q) = 1 \;\middle|\; \begin{array}{r} (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ b_i \leftarrow f_i(\mathbf{a}_i) \end{array}\right] > \kappa \ ,$$

  then $(x, \mathsf{Ext}^{f_1, \ldots, f_q}(x)) \in \mathcal{R}$.

As with linear PCPs, a linear MIP is is **instance-independent** if $V_Q(x)$ is a function in the size of x.

We also consider bounded variants of PCPs and (strong) MIPs:

Definition 2.3.3. A q-query LPCP (resp. strong LMIP) $(\mathbf{P}, (V_Q, V_D))$ over a relation $\mathcal{R}$ and finite field $\mathbb{F}_p$ with proof length ℓ is *B-bounded* with error α if for all $(x, w) \in \mathcal{R}$:

$$\Pr\left[b_1, \ldots, b_q \in [-B, B] \;\middle|\; \begin{array}{r} \pi \leftarrow \mathbf{P}(x, w) \\ (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ \forall i \in [q],\ b_i = \langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i) \rangle \in \mathbb{Z} \end{array}\right] \geq 1 - \alpha \ .$$

Note that any LPCP is $(p^2 \cdot \ell)$-bounded with error 0, in which case we either say that it is *trivially bounded*. Whenever we do not give an explicit bound, the LPCP is assumed to be trivially bounded.

Furthermore, observe that (by the union bound) if an LPCP is B-bounded with error α, then its t-wise repetition is B-bounded with error $t \cdot \alpha$.

2.3.2 Linear PCPs for this Work

In this work, we use the following linear PCPs known in the literature: For our results, we utilize the existence of the following linear PCPs for arithmetic circuits:

Theorem 2.3.4 ([BIOW20], Appendix B.1)**.** *Let $C \colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \mathbb{F}_p$ be an arithmetic circuit of size s over finite field $\mathbb{F}_p$. There exists a 2-query instance-oblivious LPCP over $\mathbb{F}_p$ for $\mathcal{R}_C = \left\{(x, w) \in \mathbb{F}^n \times \mathbb{F}^h \mid C(x, w) = 1\right\}$ with perfect completeness, knowledge soundness error $2/p$ against affine strategies, and proof length $s + s^2$ (field elements).*

If we restrict to Boolean circuits, then there is an LPCP with the same parameters which for any $\lambda \in \mathbb{N}$ with is $O(p^2 s\lambda)$-bounded with error $2^{-\lambda}$.

Theorem 2.3.5 ([BHI+24], Theorem 1.2)**.** *Let $C\colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \mathbb{F}_p$ be an arithmetic circuit of size s over finite field $\mathbb{F}_p$ with $p > 2$. There exists a 1-query instance-oblivious LPCP over $\mathbb{F}_p$ for the relation*

$$\mathcal{R}_C = \left\{(x, w) \in \mathbb{F}^n \times \mathbb{F}^h \mid C(x, w) = 1\right\}$$

with perfect completeness, knowledge soundness error $O(1/\sqrt{p})$ against affine strategies, and proof length $s \cdot \mathsf{poly}(p)$ (field elements).

By applying a transformation given in [IKO07, Section 5] from LPCPs to strong LMIPs to Theorem 2.3.5 we get the following:

Corollary 2.3.6. *Let $C\colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \mathbb{F}_p$ be an arithmetic circuit of size s over finite field $\mathbb{F}_p$ with $p > 2$. There exists a $O(1)$-query instance-oblivious strong LMIP over $\mathbb{F}_p$ for $\mathcal{R}_C = \left\{(x, w) \in \mathbb{F}^n \times \mathbb{F}^h \mid C(x, w) = 1\right\}$ with perfect completeness, knowledge soundness error $O(1)$ against affine strategies, and proof length $O(s \cdot \mathsf{poly}(p))$ (field elements).*

In Section 4.5.1 we give an alternate transformation from LPCP to strong LMIP which has better concrete parameters, which we apply to the LPCP described in Theorem 2.3.4.

In the upcoming open problems in Section 4.1.2, we add a conjecture that the LPCP of Theorem 2.3.5 is bounded. We stress that this conjecture is included here purely to formally define what we mean in the discussion, and is not used in this thesis.

Conjecture 2.3.7. The LPCP from Theorem 2.3.5 is $O(\lambda p^2\sqrt{s})$-bounded with error $2^{-\lambda}$.

2.3.3 Linearity Testing

In our construction of strong linear MIPs (Section 4.5.1) we utilize a variant of the [BLR90] linearity test for linear-consistent functions.

Definition 2.3.8. A triple of functions $f_1, f_2, f_3\colon \mathbb{F}^\ell \to \mathbb{F}$ is *linear-consistent* if there exists a linear function $g\colon \mathbb{F}^\ell \to \mathbb{F}$ and $c_1, c_2, c_3 \in \mathbb{F}$ so that $c_1 + c_2 = c_3$ and for every $i \in [3]$ and $z \in \mathbb{F}^\ell$, $f_i(z) = g(z) + c_i$.

Theorem 2.3.9 ([AHRS01], Theorem 2)**.** *Let $f_1, f_2, f_3\colon \mathbb{F}^\ell \to \mathbb{F}$. If*

$$\delta := \Pr_{z_1, z_2 \leftarrow \mathbb{F}^\ell}[f_1(z_1) + f_2(z_2) \neq f_3(z_1 + z_2)] < \frac{2}{9}\ ,$$

then there exist a triple of linear-consistent functions $g_1, g_2, g_3\colon \mathbb{F}^\ell \to \mathbb{F}$ so that for every $i \in [3]$, $\Delta(f_i, g_i) \leq \delta$.

2.4 Cryptographic Primitives

For two probability distributions X, Y with domain D, we sometimes use the notation $X \approx_c Y$ to state that the distributions are computationally indistinguishable. This means that for any PPT distinguisher $\mathcal{A} : D \to \{0, 1\}$ we have $\sum_{\alpha \in D} |\Pr[D(X) = 0] - \Pr[D(Y) = 0]|$ is negligible in λ.

Definition 2.4.1 (Pseudorandom Generator)**.** Let $m \in \mathsf{poly}(\lambda)$ with $m > \lambda$. A function $G : \{0,1\}^\lambda \to \{0,1\}^m$ is a pseudorandom generator if, for uniformly random $\mathsf{s} \xleftarrow{\$} \{0,1\}^\lambda$ and $r \xleftarrow{\$} \{0,1\}^m$, we have

$$G(\mathsf{s}) \approx_c r.$$

Definition 2.4.2 (Collision-Resitant Hash Function)**.** Let $n, m \in \mathsf{poly}(\lambda)$. A collision-resistant hash function is a seeded function $\mathsf{CRHF} : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^m$ with the property that for all PPT adversaries $\mathcal{A}$, uniformly random seed $\mathsf{s} \in \{0,1\}^\lambda$ we have $\mathcal{A}(\mathsf{s})$ outputs x,x' with $x \neq x'$ and $\mathsf{CRHF}_\mathsf{s}(x) = \mathsf{CRHF}_\mathsf{s}(x')$ with negligible probability.

2.4.1 Public-Key Encryption

Definition 2.4.3 (Public-Key Encryption)**.** A public-key encryption (PKE) scheme is a triple of PPT algorithms

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$**:** Given the security parameter λ the key-generation algorithm outputs a public key pk and a secret key sk.

$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m})$**:** Given a public key pk and a message m encryption outputs a ciphertext ct.

$\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$**:** Given a secret key sk and a ciphertext ct decryption outputs a message m.

Correctness. For all $\lambda, S \in \mathbb{N}$, messages m and $(\mathsf{pk}, \mathsf{sk})$ in the range of KeyGen we have that $\mathsf{m} = \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{m}))$.

IND-CPA security. For all $\lambda \in \mathbb{N}$ and all adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have that

$$\mathsf{Pr}\left[b \leftarrow \mathcal{A}_2(\mathsf{st}, c) : \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pk}) \\ b \xleftarrow{\$} \{0,1\} \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}[\lambda].$$

We now provide the definition of homomorphic encryption scheme.

Definition 2.4.4 (Homomorphic Encryption Scheme)**.** A homomorphic encryption scheme for function class $\mathcal{F}$ is PKE scheme with the following additional algorithm:

$\mathsf{ct} \leftarrow \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell))$**:** Given public key pk, a function $f \in \mathcal{F}$ and ciphertexts $(\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ the evaluation algorithm outputs a new ciphertext ct.

IND-CPA is defined in a analogous way as for PKE. We now present the definitions of homomorphic correctness and compactness.

Homomorphic correctness. For all $\lambda \in \mathbb{N}$, messages $\mathsf{m}_1, \ldots, \mathsf{m}_\ell$, any function f, $(\mathsf{pk}, \mathsf{sk})$ in the range of KeyGen we have that

$$f(\mathsf{m}_1, \ldots, \mathsf{m}_\ell) = \mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{Enc}(\mathsf{pk}, \mathsf{m}_1), \ldots, \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_\ell)))).$$

.

Compactness. There exists a polynomial p such that for all $\lambda \in \mathbb{N}$, all functions $f \in \mathcal{F}$, all inputs $\mathsf{m}_1, \ldots, \mathsf{m}_\ell$, all $(\mathsf{sk}, \mathsf{pk})$ in the support of $\mathsf{KeyGen}(1^\lambda)$, and all ct_i in the support of $\mathsf{Enc}(\mathsf{pk}, b_i)$ it holds that $|\mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, ..., \mathsf{ct}_\ell))| = p(\lambda, |f(\mathsf{m}_1, ..., \mathsf{m}_\ell)|)$.

Remark 2.4.5. We call an encryption scheme fully homomorphic (FHE) if $\mathcal{F}$ is the set of all poly sized circuits. We call an encryption scheme linearly homomorphic if $\mathcal{F}$ is the set of linear functions over the message space.

2.4.2 Oblivious Transfer

Oblivious transfer is a basic interactive protoco in which a sender and a receiver interact. In its most basic form the sender's input to the protocol are two bits $m^{(0)}$ and $m^{(1)}$ and the receiver's input is the so call choice bit $b \in \{0,1\}$. At the end of the protocol the receiver is supposed to learn $m^{(b)}$ but not $m^{(1-b)}$, also the sender is not supposed to learn b.

There are a few generalizations of this primitive two of which we are going to cover here. First, instead of bits $m^{(0)}$ and $m^{(1)}$ are sometimes k bit strings. Also, if we run ℓ instances of OT at the same time we call it ℓ-batch OT. These two can also be combined into ℓ-batch k-bit string OT.

Definition 2.4.6 (Oblivious Transfer (OT))**.** An ℓ-batch k-bit string OT protocol OT is a protocol between two interactive PPT programs $(\mathsf{OTR}, \mathsf{OTS})$, where OTR and OTS denote, respectively, the receiver and the sender.

$\mathsf{OTR}(1^\lambda, \ell, k, s)$: An interactive algorithm that takes in a security parameter 1^λ, batching parameter ℓ, message length k, and choice vector $s \in \{0,1\}^\ell$, and outputs $m \in \{0,1\}^\ell$.

$\mathsf{OTS}(1^\lambda, \ell, k, m^{(0)}, m^{(1)})$: An interactive algorithm that takes in a security parameter 1^λ, batching parameter 1^ℓ, message length 1^k and two message vectors $m^{(0)}, m^{(1)} \in \mathsf{M}^\ell$, for $\mathsf{M} = \{0,1\}^k$, and outputs $\perp$

We require the following.

Correctness. OT is $\alpha(\cdot)$-correct if for any λ, $s \in \{0,1\}^\ell$, $(m_i^{(0)}, m_i^{(1)})_{i\in[\ell]} \in (\mathsf{M}\times\mathsf{M})^\ell$, the probability over an honest interaction between $\mathsf{OTR}(1^\lambda, k, \ell, s)$ and $\mathsf{OTS}(1^\lambda, k, \ell, m^{(0)}, m^{(1)})$ that OTR outputs $(m_1^{(s_1)} \dots m_\ell^{(s_\ell)})$ is $\geq \alpha(\lambda)$. The protocol is perfectly correct if $\alpha = 1$. By default we require prefect correctness.

Semi-Honest Receiver Security. For any strings s_0, $s_1 \in \{0,1\}^\ell$, $m^{(0)}$, $m^{(1)} \in \mathsf{M}^\ell$ we have that $\mathsf{view}_s^{\mathsf{OT}}(s_0, (m^{(0)}, m^{(1)}))$ and $\mathsf{view}_s^{\mathsf{OT}}(s_1, (m^{(0)}, m^{(1)}))$ are computationally indistinguishable.

Semi-Honest Sender Security. For any $s \in \{0,1\}^\ell$ and $m^{(0)}$, $m^{(1)}$, $z^{(0)}$, $z^{(1)} \in \mathsf{M}^\ell$ such that $\{(m_i^{s_i})\} = \{(z_i^{s_i})\}$, we have that the two views

$$\mathsf{view}_r^{\mathsf{OT}}(s, (m^{(0)}, m^{(1)})) \text{ and } \mathsf{view}_r^{\mathsf{OT}}(s, (z^{(0)}, z^{(1)}))$$

are computationally indistinguishable.

OT Terminologies. We may sometimes refer to an ℓ-batch single-bit OT as an ℓ-batch OT. Also, whenever we say a k-bit string OT we mean $\ell = 1$.

We define notions of *rate* as asymptotic ratios between the actual communication under a given protocol and the best achievable communication under a (possibly) insecure protocol; i.e., for ℓ-batch single-bit OT the sender must communicate at least ℓ bits to the receiver, if perfect correctness is required. Therefore, the optimal download communication is ℓ. Similarly, the optimal total communication is 2ℓ.

Expected Download Rate. An ℓ-batch single-bit OT protocol has expected download rate c if for all λ, s, $m^{(0)}$, $m^{(1)}$, and all but finitely many ℓ

$$\frac{\ell}{d(\lambda,\ell)} \geq c,$$

where $d(\lambda,\ell)$ is expected communication from the sender $\mathsf{OTS}(1^\lambda,\ell,m^{(0)},m^{(1)})$ to the receiver $\mathsf{OTR}(1^\lambda,\ell,s)$.

Expected (Overall) Rate. An ℓ-batch single-bit OT protocol has expected (overall) rate c if for all λ, s, $m^{(0)}$, $m^{(1)}$, and all but finitely many ℓ

$$\frac{2\ell}{t(\lambda,\ell)} \geq c,$$

where $t(\lambda,\ell)$ is the expected total communication.

2.4.3 Private-Information Retrieval (PIR)

Private information retrieval (PIR) is a protocol between a user and servers. The servers have a database DB and the user wants to learn an entry of the database without revealing to the servers which entry it wants to learn.

This task is trivially achieved by a server just sending the entire database to the user. Therefore, we call a PIR protocol non-trivial if the communication to be significantly lower than the size of the entire database. In this work, we only consider PIR protocol with only one server.

Definition 2.4.7 (Non-Trivial PIR)**.** A non-trivial (single-server) private information retrieval ntPIR is an interactive protocol between two interactive PPT programs $(\mathsf{PIRU},\mathsf{PIRS})$, where PIRU and PIRS denote, respectively, the client (user) and the server.

$\mathsf{PIRU}(1^\lambda,1^n,i)$: An interactive algorithm that takes in a security parameter 1^λ, the database size n, and a choice index $i\in[n]$, and at the end of the interaction outputs $y\in\{0,1\}$.

$\mathsf{PIRS}(1^\lambda,1^n,\mathsf{DB})$: An interactive algorithm that takes in a security parameter 1^λ, database size n and a database $\mathsf{DB}\in\{0,1\}^n$, and outputs $\perp$.

We require the following properties.

Correctness. The PIR protocol is $\alpha(\cdot)$-correct if for any λ, n, $i\in[n]$ and $\mathsf{DB}\in\{0,1\}^n$, the probability over an honest interaction between $\mathsf{PIRU}(1^\lambda,1^n,i)$ and $\mathsf{PIRS}(1^\lambda,1^n,\mathsf{DB})$ that PIRU outputs DB_i is $\geq\alpha(\lambda)$. The protocol is perfectly correct if $\alpha=1$. By default we require perfect correctness.

Semi-Honest Client Security. For any n, i, $i'\in[n]$, databasis $\mathsf{DB}\in\{0,1\}^n$, we have $\mathsf{view}_s^{\mathsf{ntPIR}}(i,\mathsf{DB})$ and $\mathsf{view}_s^{\mathsf{ntPIR}}(i',\mathsf{DB})$ are computationally indistinguishable.

Non-Trivial Expected Download Communication. There exists a polynomial poly such that for all sufficiently large λ, for all $n\geq\mathsf{poly}(\lambda)$, for all $i\in[n]$, and $\mathsf{DB}\in\{0,1\}^n$, the expected communication from $\mathsf{PIRS}(1^\lambda,1^n,i)$ to $\mathsf{PIRU}(1^\lambda,1^n,\mathsf{DB})$ is $d(\lambda,n)<n$.

The following result shows that non-trivial PIR implies public-key cryptography in a black-box way. We use this theorem for our lower-bound results.

Theorem 2.4.8 ([DMO00])**.** *There exists a black-box construction of OT from a non-trivial PIR protocol.*

2.4.4 HILL-Entropic Encodings

We recall the notion of HILL-entropic encodings from [MW20]. Intuitively they are a way to encode a message such that it is easy to decode while the encoding itself having a lot of pseudo-entropy.

Definition 2.4.9 (HILL-Entropic Encodings)**.** An (α, β)-HILL-entropic encoding scheme with selective security in the CRS setting consists of two PTT algorithms:

- $c \leftarrow \mathsf{En}_{\mathsf{crs}}(1^\lambda, \mathsf{m})$: An encoding algorithm that takes a common random string crs and a message m producing an encoding c.
- $m \leftarrow \mathsf{De}_{\mathsf{crs}}(c)$: A decoding algorithm that takes a common random string crs and an encoding c and produces a message m.

Correctness. There is some negligible μ such that for all $\lambda \in \mathbb{N}$ and all $m \in \{0,1\}^*$ we have

$$\Pr[\mathsf{De}_{\mathsf{crs}}(\mathsf{En}_{\mathsf{crs}}(1^\lambda, \mathsf{m})) = \mathsf{m}] = 1 - \mu(\lambda).$$

α-Expansion. For all $\lambda, k \in \mathbb{N}$ and all $\mathsf{m} \in \{0,1\}^k$ we have $|\mathsf{En}_{\mathsf{crs}}(1^\lambda, \mathsf{m})| \leq \alpha(\lambda, k)$.

β-HILL-Entropy. There exists an algorithm SimEn s.t. for any polynomial $k = k(\lambda)$ and any ensamble of messages $\mathsf{m} = \{\mathsf{m}_\lambda\}$ of length $|\mathsf{m}_\lambda| = k(\lambda)$, consider the following "real" experiment:

- $\mathsf{crs} \xleftarrow{\$} \{0,1\}^{t(\lambda,k)}$
- $c \leftarrow \mathsf{En}_{\mathsf{crs}}(1^\lambda, \mathsf{m}_\lambda)$

and let CRS, C denote the random variables for the corresponding values in the "real" experiment. Also consider the following "simulated" experiment:

- $(\mathsf{crs}', c') \leftarrow \mathsf{SimEn}(1^\lambda, \mathsf{m}_\lambda)$

and let CRS', C' denote the random variables for the corresponding values in the "simulated" experiment. We require that $(CRS, C) \approx_c (CRS', C')$ and

$$\tilde{H}_\infty(C'|CRS') \geq \beta(\lambda, k).$$

We call a (α,β)-HILL-entropic encoding good if $\alpha(\lambda, k) = k(1 + o(1)) + \mathsf{poly}[\lambda]$ and $\beta(\lambda, k) = k(1 - o(1)) - \mathsf{poly}[\lambda]$. Moran and Wichs [MW20] provide good HILL-entropic encodings from DCR [Pai99, DJ01] or LWE [Reg05] in the CRS model. They also show that the CRS must be as big as the encoded message.

2.4.5 Polynomial Commitment Schemes

We recall the notion of polynomial commitment schemes.

Definition 2.4.10 (Polynomial Commitment)**.** A polynomial commitment scheme in a tuple four algorithms:

- $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, d)$: Gets as input the security parameter λ and a degree d. It outputs a common reference string crs.
- $(\mathsf{com}, \mathsf{st}) \leftarrow \mathsf{Commit}_{\mathsf{crs}}(f)$: Gets as inputa polynomial f and outputs a commitment com and a state st.
- $\tau \leftarrow \mathsf{Open}_{\mathsf{crs}}(f, x, \mathsf{st})$: Gets as input a position x and a state st and outputs an opening for $f(x)$ on x.
- $\{0,1\} \leftarrow \mathsf{Verify}_{\mathsf{crs}}(\mathsf{com}, x, y, \tau)$: Gets as input a commitment com, a position x, a value y, and an opening τ.

Completeness. A polynomial commitment is called complete if for any $d, \lambda \in \mathbb{N}$, $x \in \mathbb{F}$, polynomial f of degree d, crs from the support of $\mathsf{Setup}(1^\lambda, d)$ we have that if $(\mathsf{com}, \mathsf{st}) \leftarrow \mathsf{Commit}_{\mathsf{crs}}(f)$ we have

$$\mathsf{Verify}(\mathsf{com}, x, f(x), \mathsf{Open}(f, x, \mathsf{st})) = 1$$

Knowledge Soundness. There exists a PPT extractor Ext such that for every $d \in \mathbb{N}$, PPT adversary $\mathcal{A}$ we have that

$$\Pr\left[\begin{array}{c|c} f' \neq f \wedge \mathsf{Verify}_{\mathsf{crs}}(\mathsf{com}, x, y, \tau) = 1 & \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ f \leftarrow \mathcal{A}(\mathsf{crs}) \\ (\mathsf{com}, \mathsf{st}) \leftarrow \mathsf{Commit}_{\mathsf{crs}}(f) \\ f' \leftarrow \mathsf{Ext}^{\mathcal{A}}(\mathsf{crs}, \mathsf{com}) \\ (\tau, x, y) \leftarrow \mathcal{A}(\mathsf{st}) \end{array} \end{array}\right]$$

is negligible.

2.5 Assumptions and Associated Notation

In this subsection we elaborate on the cryptographic assumptions required to prove some of our results. We also explain some of the tools required in our use of the these assumptions.

2.5.1 Decisional Diffie-Hellman (DDH)

In the following we elaborate on one of the most popular assumptions for public-key cryptography. In fact, in the paper introducing public-key cryptography [DH76] Diffie and Hellman implicitly make this assumption. The assumption has held up to a lot of cryptoanalysis on certain groups.

Notation In the following, let $\mathcal{G}$ be a (prime-order) *group generator*, that is, $\mathcal{G}$ is an algorithm that takes as an input a security parameter 1^λ and outputs $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is the description of a multiplicative cyclic group, p is the order of the group which is always a prime number unless differently specified, and g is a generator of the group. Sometimes we denote the size of the group by $|\mathbb{G}|$.

When it is convenient we write $[a]$ for the value g^a. Similarly, if $\mathbf{A} \in \mathbb{Z}_p^{n\times m}$ is a matrix with entries $a_{i,j}$ then $[\mathbf{A}]$ denotes the matrix where each (i,j)-entry is the value $g^{a_{i,j}}$. Note that given $\mathbf{x} \in \mathbb{Z}_p^n$, $\mathbf{y} \in \mathbb{Z}_p^m$ and $[\mathbf{A}]$, we can compute $\mathbf{x}^T[\mathbf{A}] = [\mathbf{x}^T\mathbf{A}]$ and $[\mathbf{A}]\mathbf{y} = [\mathbf{A}\mathbf{y}]$.

In the following we state the decisional version of the Diffie-Hellman (DDH) assumption.

Definition 2.5.1 (Decisional Diffie-Hellman Assumption)**.** Let the description of the group be $(\mathbb{G}, p, g) \xleftarrow{\$} \mathcal{G}(1^\lambda)$. We say that the DDH assumption holds (with respect to $\mathcal{G}$) if for any PPT adversary $\mathcal{A}$

$$|\Pr[1 \leftarrow \mathcal{A}((\mathbb{G}, p, g), ([a], [b], [ab]))] - \Pr[1 \leftarrow \mathcal{A}((\mathbb{G}, p, g), ([a], [b], [c]))]| \leq \mathsf{negl}[\lambda]$$

where $a, b, c \xleftarrow{\$} \mathbb{Z}_p$.

2.5.2 Cryptographic Group Actions

While DDH and related assumptions have seem to be classically secure over certain groups it is known that they will not resist any attack by a sufficiently good quantum computer [Sho94, BL95, EG24]. To protect against the looming threat of quantum computers have suggested to move to isogeny based cryptography.

In this work, we use part of isogeny based cryptography which can be abstracted away as group actions. [ADMP20] explain and explore this connection well.

We quickly recap what group actions are to understand our notation. We had to slightly modify the notation of [ADMP20] to match our group notation.

Definition 2.5.2 (Group Action)**.** A group $\mathbb{G}$ is said to act on a set $\mathcal{X}$ if there is a map $\star : \mathbb{G} \times \mathcal{X} \to \mathcal{X}$ that satisfies the following two properties:

Identity : If $[0]$ is the identity element of $\mathbb{G}$, then for any $x \in \mathcal{X}$, we have $[0] \star x = x$

Compatibility : For any $[g], [h] \in \mathbb{G}$ and any $x \in \mathcal{X}$, we have $([g] + [h]) \star x = [g] \star ([h] \star x)$

In this work the require the group action to be regular and to be a weakly pseudorandom effective group action. A group action is regular if for any $x \in \mathcal{X}$ we have that $g \mapsto g \star x$ is a bijection between $\mathbb{G}$ and $\mathcal{X}$.

Definition 2.5.3 (Weak Pseudorandom Group Action)**.** A group action is weakly pseudorandom if the permutations $\{x \mapsto g \star x\}_{g\in\mathbb{G}}$ are weakly pseudorandom permutations.

2.5.3 Learning with Errors (LWE)

One of the most popular assumption that has held up to all quantum cryptanalysis is Learning with Errors [Reg05]. Intuitively, it relies on linear algebra being hard if the the equations are approximate.

Gaussian Distribution For any $\sigma \in \mathbb{R}_{>0}$ let $\rho_\sigma(\mathbf{x}) = exp(-\pi||\mathbf{x}||^2/\sigma^2)$ be the Gaussian function on $\mathbb{R}^m$ with deviation σ. Let χ_σ^m be the discrete Gaussian distribution over $\mathbb{Z}^m$ with deviation σ.

Definition 2.5.4 (Learning with Errors Assumption)**.** The LWE assumption states that for $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^m$, and $\mathbf{b} \xleftarrow{\$} \mathbb{Z}_q^n$ being uniformly random and $\mathbf{e} \leftarrow \chi_\sigma^n$ being sampled from a small gaussian distribution. We have that

$$(\mathbf{A}, \mathbf{As} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{b})$$

Definition 2.5.5 (Lattice Trapdoor)**.** A lattice trapdoor consists of two PPT algorithms

- $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapSamp}(1^n, 1^m, q)$ on inputs $n, m, q \in \mathbb{N}$ outputs matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{T} \in \mathbb{Z}_q^{m\times m}$
- $\mathbf{r} \leftarrow \mathsf{SampleD}(\mathbf{A}, \mathbf{T}, \mathbf{u}, s)$ on inputs $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{T} \in \mathbb{Z}_q^{m\times m}$, $\mathbf{u} \in \mathbb{Z}_q^n$, $s \in$ and outputs $\mathbf{r} \in \mathbb{Z}_q^m$

For any $n \geq 1$, $q \geq 2$, sufficiently large $m = \Omega(n \log(q))$, and sufficiently large $\sigma = \Omega(\sqrt{n}\log(q))$ these two distributions producing $(\mathbf{A}, \mathbf{T}, \mathbf{u}, \mathbf{r})$ are statistically close.

- $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapSamp}(1^n, 1^m, q)$; $\mathbf{u} \leftarrow \mathbb{Z}_q^n$; $\mathbf{r} \leftarrow \mathsf{SampleD}(\mathbf{A}, \mathbf{T}, \mathbf{u}, \sigma)$.
- $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapSamp}(1^n, 1^m, q)$; $\mathbf{r} \leftarrow \chi_\sigma^m$; $\mathbf{u} \leftarrow \mathbf{Ar}$.

Also, $\mathbf{A}$ is statistically close to uniformly random.

2.6 Generic Models and Applications

In cryptography it is sometimes useful to model certain cryptographic primitives as an oracle, which implements an idealized version of the given primitive. This allows to prove certain results with regards to the idealized version that one might not be able to prove by using the actual primitive.

When proving a result with respect to an oracle the hope is that this translates naturally into a protocol in the real world by just replacing the oracle by the primitive it is trying to replace. This heuristic can run into issues, some of which we are going to address later in this work.

Here, we detail some of the oracle we use and some of their applications.

2.6.1 Random Oracle Model

The random oracle [BR93] is the most popular idealized model in cryptography. It is often used in practice as it allows for efficient design of signatures [FS87], non-malleable encryption [FO99] and much more [Mic94, DN93].

The random oracle is an idealization of a hash function as a random function.

Definition 2.6.1. The random oracle model provides access to a random function $\mathsf{H} : \{0,1\}^* \to \{0,1\}$.

Remark 2.6.2. By subdividing the input space correctly the ROM can also me made to output a string instead of just a bit.

2.6.2 Ideal Cipher Model

In our constructions we use the ideal cipher model (ICM), which can be dated back to Shannon [Sha49, HKT11, RSS11]. As the name suggests it is supposed to idealize the properties of a perfect block cipher (a keyed permutation).

Definition 2.6.3. For any $n \in \mathbb{N}$ the ideal cipher model provides oracle access a keyed permutation $\mathsf{P} : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$, which for each key k we have P_k is an independent random permutation.

2.6.3 Generic Group Model

The generic group model (GGM) is a model that idealizes a cryptographically hard group [Nec94]. There are different variants of the GGM. We use Shoup's [Sho97] version of the generic group model.

Definition 2.6.4. The generic group model models cryptographic group operations via an oracle. For a prime p' the oracle holds a permutation $f : \mathcal{L} \mapsto \mathbb{Z}_{p'}$ that maps from the label space $\mathcal{L}$ which are just binary representations of the numbers $0, \ldots, p'-1$, to the group $(\mathbb{Z}_{p'}, +)$. At the beginning of a security game f is sampled uniformly at random from all permutations and all parties are provided with the label $g \leftarrow f^{-1}(1)$. Throughout security games the parties have oracle access to the oracle $\mathcal{G}$, which take two labels χ_1, χ_2 as input and responds with $f^{-1}(f(\chi_1) + f(\chi_2))$. We denote by $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$ the act of sampling a random GGM oracle of size $\geq 2^\lambda$.

Remark 2.6.5. Our constructions in Construction 4.3.9 and Corollary 4.6.2 additionally use a random oracle H, which is a uniformly random function with specified output length χ bits, sampled at the same time as the GGM oracle. For simplicity, in our constructions we always set $\chi = \lambda$ and always count total oracle calls to both oracles.

If an algorithm P has access to an oracle $\mathcal{O}$ we denote this by $P^\mathcal{O}$. Further, $y \leftarrow_{\mathsf{tr}} P^\mathcal{O}(x)$ means y is the output of evaluating the algorithm $P^\mathcal{O}$ on x and tr is the trace of P's interactions with the oracle $\mathcal{O}$, i.e., it contains all the queries to the oracle and its responses.

2.6.4 Distributed Discrete Logarithm

In our work we use the distributed discrete log algorithm:

Lemma 2.6.6 ([BGI16, BGI17])**.** *Let $\delta > 0$, $B \in \mathbb{N}$, p' be a prime with $B, T < p'$, and $\delta = 4B/T$. There exists an algorithm DDL that does T GGM queries such that for all GGM labels $h \in \mathcal{L}$,*

$$\Pr\left[\forall x \in [-B, B], \begin{array}{l}\mathsf{DDL}^\mathcal{G}_{B,\delta}(h) \\ \quad - \mathsf{DDL}^\mathcal{G}_{B,\delta}(h \cdot g^x) = x\end{array} \;\middle|\; \mathcal{G} \leftarrow \mathsf{GGM}(p')\right] \geq 1 - \delta \ .$$

Further, $\forall x \notin [-T, T]$ we have

$$h \cdot g^{\mathsf{DDL}^\mathcal{G}_{B,\delta}(h)} \neq h \cdot g^{x + \mathsf{DDL}^\mathcal{G}_{B,\delta}(h \cdot g^x)}.$$

Remark 2.6.7. Previous work on distributed discrete logarithms are in the plain model. They require a large group and a function that maps elements of this group to random bit-strings. The generic group model provides both of these properties.

2.6.5 Designated-Verifier SNARGs

Succinct non-inteactive arguments (SNARGs) allow a prover to succinctly convince a verifier of a statement. Here, we consider a variant, where only a verifier with specific information can verify these statements. These prove systems are called designated-verifier SNARGs.

Definition 2.6.8. A designated-verifier succinct non-interactive argument (dv-SNARG) in the generic group model, $(\mathsf{Setup}, \mathbf{P}, \mathbf{V})$ for a relation $\mathcal{R} = \{(x, w)\}$ is defined as follows:

- *Syntax.* We describe a dv-SNARG with message length ℓ:
  - The setup algorithm Setup receives an input size parameter 1^n. It outputs a common reference string crs and a verification state st.
  - The (honest) prover algorithm $\mathbf{P}$ receives as input a common reference string crs, instance $x \in \{0,1\}^n$, and witness $w \in \{0,1\}^m$. It outputs a proof $\mathsf{pf} \in \{0,1\}^\ell$.
  - The verifier algorithm $\mathbf{V}$ receives as input a verification state st, and instance $x \in \{0,1\}^n$, and a proof $\mathsf{pf} \in \{0,1\}^\ell$. It outputs a bit $b \in \{0,1\}$.
- *Completeness.* A dv-SNARG has completeness error α if for all $(x, w) \in \mathcal{R}$ and $\lambda \in \mathbb{N}$:

$$\Pr\left[\mathbf{V}^{\mathcal{G}}(\mathsf{st}, x, \mathsf{pf}) = 1 \;\middle|\; \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}^{\mathcal{G}}(1^{|x|}) \\ \mathsf{pf} \leftarrow \mathbf{P}^{\mathcal{G}}(\mathsf{crs}, x, w) \end{array}\right] \geq 1 - \alpha(\lambda, |x|) \ .$$

- *Soundness.* A dv-SNARG has (adaptive) soundness with error δ if for every $\lambda \in \mathbb{N}$, $n \in \mathbb{N}$, and prover $\mathbf{P}'$ that makes at most t queries to the GGM oracle:

$$\Pr\left[\begin{array}{c} x \notin \mathcal{L}(\mathcal{R}) \\ \wedge\, \mathbf{V}^{\mathcal{G}}(\mathsf{st}, x, \mathsf{pf}) = 1 \end{array} \;\middle|\; \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}^{\mathcal{G}}(1^{|x|}) \\ (x, \mathsf{pf}) \leftarrow \mathbf{P}'^{\mathcal{G}}(\mathsf{crs}) \end{array}\right] \leq \delta(\lambda, |x|, t) \ .$$

- *Succinctness.* For every large enough λ, GGM oracle $\mathcal{G}$ in the image of $\mathsf{GGM}(\lambda)$, $(x, w) \in \mathcal{R}$ and $(\mathsf{crs}, \mathsf{st})$ in the image of $\mathsf{Setup}^{\mathcal{G}}(1^{|x|})$, we have $|\mathsf{pf}| = o(w)$ for $\mathsf{pf} = \mathbf{P}^{\mathcal{G}}(\mathsf{crs}, x, w)$.

A SNARG with the following additional knowledge property is known as a SNARK:

- *(Straight-line) Knowledge soundness.* A dv-SNARG has (adaptive) knowledge soundness (in which case we refer to it as a dv-SNARK) with knowledge κ if there exists an expected polynomial time PPT extractor Ext so that for every $\lambda \in \mathbb{N}$, $n \in \mathbb{N}$, and prover $\mathbf{P}'$ that makes at most t queries to the GGM oracle,

$$\Pr\left[\begin{array}{c} (x, w) \notin \mathcal{R} \\ \wedge\, \mathbf{V}^{\mathcal{G}}(\mathsf{st}, x, \mathsf{pf}) = 1 \end{array} \;\middle|\; \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}^{\mathcal{G}}(1^n) \\ (x, \mathsf{pf}) \leftarrow_{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{crs}) \\ w \leftarrow \mathsf{Ext}(x, \mathsf{pf}, \mathsf{tr}) \end{array}\right] \leq \kappa(\lambda, n, t) \ .$$

Chapter 3

Lower-Bounds on Public-Key Operations in PIR

3.1 Introduction

Secure Multi-Party Computation (MPC), allows two parties to jointly evaluate a function f while leaking nothing about their input to the other party beyond the output of f. A central goal of modern cryptography is to construct efficient MPC protocols. This goal is important not only from a theoretical but also from a practical viewpoint. The computational efficiency of all MPC protocols is typically bottlenecked by public-key operations (e.g., group operations, oblivious transfers (OT)). OT extension is a technique toward reducing public-key operations [Bea96, IKNP03], allowing one to get the results of many OTs at the cost of performing only a few OTs and some symmetric-key operations. This technique has revolutionized the practical development of MPC, leading to protocols which employ a small number of public-key operations for sophisticated tasks.

A significant limitation of existing OT extension techniques is their high communication cost: for performing ℓ 1-out-of-2 bit OTs, the sender communicates at least 2ℓ bits. This severely limits the use of OT extension in MPC settings where a low amount of communication is required 'by design' (e.g., Private-Information Retrieval (PIR)). The overarching goal of our paper is to understand communication-computation tradeoffs in such MPC settings. We show that in many such situations, performing many public-key operations is provably unavoidable. To put our results in context, let us illustrate how communication efficient OT would impact private information retrieval.

PIR. Private information retrieval (PIR) [CGKS95, KO97] is a fundamental cryptographic primitive that allows a user to fetch a database entry without revealing to the server which database entry it learns. PIR becomes non-trivial if the server-to-user communication is strictly less than the database size (and ideally growing sub-linearly or even polylogarithmically in the database size). In some applications, one may need extra properties, such as an overall (as opposed to server-to-user) sub-linear communication or server privacy. Throughout the paper, we require neither of these unless otherwise stated. Since we prove lower bounds, this makes our results stronger. A truly efficient PIR protocol has significant real-world applications such as private certificate retrieval or private DNS lookups. By now, we know how to build PIR with communication complexity polylogarithmic in n from a wide

range of assumptions [CMS99, IP07, DGI+19, CGH+21, HHC+23]. While the amount of communication is attractively low, the computation overhead leaves much to be desired.

Computational complexity of PIR. In (single-server) PIR protocols, the running time of the server cannot be sub-linear in n, the database size, without preprocessing [BIM00]. If it was sub-linear the server could not read all the entries, leaking information about the user's index i. Faced with this lower-bound, and the fact that PIR requires public-key assumptions [DMO00], one may wish to settle for the next best thing: making the number of public-key operations independent of n. Somehow curiously, in all existing PIR protocols based on Diffie-Hellman or OT related assumptions [IP07, DGI+19, CGH+21], not only the server's running time, but the number of public-key operations performed by the server grows at least linearly with n. There is no evidence, however, if this is inherent, and in fact, it has remained an open problem whether one can build a PIR protocol where the number of public-key operations is sub-linear in n.

Is it possible construct a single-server non-preprocessing PIR with a sub-linear amount of public-key operations and an arbitrarily large number of symmetric-key operations?[1]

OT Extension. A major tool used for minimizing computation is OT extension. Existing OT extension techniques induce at least a linear amount of communication for the sender, making them unsuitable for PIR applications. Specifically, under existing constructions, an extended OT sender needs to communicate at least as many bits as its total input length. Beaver's seminal construction [Bea96] works by encoding all the sender messages into a garbled circuit, which the receiver can evaluate only on the labels that correspond to her choice bits; the IKNP protocol [IKNP03] establishes correlated randomness between the sender and the receiver, allowing the sender to XOR his messages with the corresponding masks such that the receiver can only de-mask the correct messages. In both these protocols, the sender's outgoing protocol messages information-theoretically determine the entire sender's input, causing the communication overhead. This state of affairs raises the following natural question.

Is it possible construct OT extension where the sender communication is close to optimal?

In the above question, by 'optimal sender communication' we mean the best information-theoretically achievable communication: which is ℓ bits for the sender for performing ℓ 1-out-2 single-bit OTs. Since OT extension is crucially used in many MPC protocols, understanding its communication complexity is of both practical and theoretical value.

Having optimal sender communication for OT extension is reminiscent of rate-1 string OT: building 1-out-of-2 string OTs for a pair of ℓ-bit strings, where (roughly speaking) the sender communication grows as $\ell + \lambda$ (as opposed to $2\ell + \lambda$), where λ is the security parameter. Two-round rate-1 OT has found a number of applications, notably in the construction of PIR protocols with polylogarithmic communication [IP07, DGI+19, GHO20, CGH+21, ADD+22, BBDP22]. We know how to build rate-1 OT from a wide variety of assumptions [IP07, DGI+19, GHO20, CGH+21, ADD+22], but all these constructions make at least a linear number of public-key operations. In particular, computational efficiency (e.g., a sub-linear number of public-key operations) and communication efficiency (e.g., sub-linear communication) seem to have largely been in conflict with each other — for reasons we have not been to justify so far. The goal of our paper is to elucidate this conflicting situation.

[1] Throughout this paper, when we say PIR, we mean a single-server non-preprocessing PIR.

3.1.1 Our Results

We answer both of the above questions, and several other related ones, negatively. In particular, we give a lower-bound on the number of public-key operations that need to be performed by servers in PIR protocols, and use this lower-bound to derive similar results for related primitives. Our core idea is based on a compilation technique that allows one to remove public-key operation queries from a PIR protocol at the cost of proportionally increasing the communication complexity in the public-key operation free protocol. As applications of our main theorem, we obtain results that settle several open problems in MPC.

In the statement below, by an SO oracle we mean a *simulatable oracle*: roughly speaking, one that can be simulated via lazy sampling (aka, on the fly). Examples of such oracles include generic-group oracles, public-key encryption oracles, etc. See Section 3.2 for more details. Also, we use the term a "party's SO bit complexity" to indicate the total bit size of all SO queries made by the party.

Theorem 3.1.1 (Informal Main Theorem)**.** *If there exists a PIR for n-bit databases (denoted as n-bit PIR) with oracle access to simulatable oracle SO, arbitrary oracle O, server communication of $\eta < cn$ for $c < 1$, $r \in o(n)$ rounds of interaction with the user, and $q \in o(n)$ bits of communication with the SO oracle, then there exists a PIR with oracle access to O, server communication $\overline{\eta} \leq \overline{c}n$ for $\overline{c} < 1$, and no calls to SO.*

We derive the following corollary.

Corollary 3.1.2. *There exists no n-bit PIR protocol built solely[2] from a simulatable oracle SO and a random oracle O with $o(n)$ round complexity, with $o(n)$ server's SO bit complexity and with $\eta \leq cn$ server's communication for $c < 1$.*

For example, letting SO be a generic group oracle (GGM), we rule out all n-bit PIR protocols that have $o(n)$ rounds and where the server's communication and the server's total number of GGM queries are, respectively, cn and $o(n)$ for $c < 1$. This holds irrespective of the number of random oracle (RO) queries the protocol is allowed to make. This closely matches the known upper-bounds, as [OS07, DGI+19] give n-bit PIR protocols based on the DDH assumption with server communication of $O(\lambda)$ and with the sever making $O(n)$ group operations.

The strength of the main theorem lies in its flexibility in instantiating the oracle SO: for example, one may let SO be an FHE oracle, and obtain similar results as long as the amount of server's communication with the FHE oracle respects the bounds. The work of [OS07] shows how to obtain PIR generically from (additively) homomorphic encryption, where the sever performs $O(n)$ homomorphic additions. Our work shows that this is close to optimal.

We will show that our computational lower-bounds for PIRs give rise to communication lower-bounds for OT extension.

Corollary 3.1.3 (OT Extension: Communication Lower-Bounds)**.** *There exists no ℓ-batch k-bit OT extension protocol with round complexity $r \in o(k\ell)$ and with server communication $\eta < c2k\ell$ for $c < 1$.*

In the above corollary, by ℓ-batch k-bit OT we mean performing ℓ OTs for pairs of k-bit strings. The IKNP protocol [IKNP03] in the ℓ single-bit OT case achieves sender

[2]By "solely" we mean that the parties also have access to a PSPACE-complete oracle. This stops the party from using any hardness assumptions other than the ones provided by the oracles.

communication of $> 2\ell$. Our result shows that the IKNP's sender communication complexity is close to optimal.

Finally, we relate PIR to other MPC protocols such as rate-1 OT to arrive at the following corollary. For brevity, we describe the statements when O is the GGM oracle, and only for rate-1 OT. In fact, we can show that achieving any rate strictly greater than 1/2 (measured as the information-theoretically optimal sender communication size divided by the sender's communication size in the actual protocol) requires making an almost linear number of group operations.

Corollary 3.1.4 (String OT Corollary)**.** *There exists no ℓ-bit string OT protocol in the GGM+RO model with sender communication of $\eta \leq c2\ell$ and $o(\ell)$ calls to the generic group for $c < 1$.*

Similar results can be proven for unbalanced PSI and in general for MPC with unbalanced inputs (aka, asymmetric MPC); see Section 3.4 for details.

Pre-Processing PIR. Our techniques also apply to the setting of single-server pre-processing PIR (for example, offline/online [CK20], doubly efficient PIR [LMW23]) in the sense that by merging the offline and online phases together, we will get a PIR without pre-processing. So, our results will imply non-trivial lower-bounds in the pre-processing setting as well.

In particular, for any constant $c < 1$ and for n-bit databases, our results rule out pre-processing PIRs with $O(n^c)$ "total" public-key operations for the server and a total cn server-side communication. Here by total we mean offline+online together. Again this holds irrespective of the number of symmetric-key operation calls. Such pre-processing PIRs will imply single-server PIR without pre-processing and with the same properties, which we have already ruled out.

For example, the pre-processing single-server PIR of Corrigan-Gibbs and Kogan [CK20, Theorem 20] induces $O(n^{2/3})$ total server communication, with the server performing, respectively, $O(n)$ and zero public-key operations in the offline and online phases. Our lower-bounds show that the number of public-key operations of [CK20] is close to optimal.

3.2 Technical Overview

First, we will give a quick example of how to simulate a generic group efficiently to illustrate the concept of simulatable oracles. Then, we sketch the proof of the main theorem. We proceed by showing why the main theorem is useful by demonstrating a few MPC protocols which imply non-trivial PIR, allowing us to apply our lower-bounds to them.

3.2.1 Generic Group Model

The generic group model models cryptographic group operations via an oracle. For a prime p' the oracle holds a permutation $f : \mathcal{L} \mapsto \mathbb{Z}_{p'}$ that maps from the label space $\mathcal{L}$ which are just binary representations of the numbers $0, \ldots, p' - 1$, to the group $(\mathbb{Z}_{p'}, +)$. At the beginning of a security game f is sampled uniformly at random from all permutations and all parties are provided with the label $g \leftarrow f^{-1}(1)$. Throughout security games the parties have oracle access to the oracle $\mathcal{G}$, which take two labels χ_1, χ_2 as input and responds with $f^{-1}(f(\chi_1) + f(\chi_2))$. We denote by $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$ the act of sampling a random GGM oracle of size $\geq 2^\lambda$.

A generic group of order p is the group $\mathbb{Z}_p$ together with a random injective encoding function $f\colon \mathcal{L} \to \mathbb{Z}_p$, where $\mathcal{L} = \{0, \ldots, p-1\}$. The algorithms can access this group via the oracle Add which decodes two encoded elements, computes a linear combination of them and gives back the encoded result. More formally, $\mathsf{Add} : \mathbb{Z}_p^2 \times \mathcal{L}^2 \to \mathcal{L}$, $(a_1, a_2, \ell_1, \ell_2) \mapsto f^{-1}(a_1 v_1 + a_2 v_2)$, where $v_i = f(\ell_i)$ for $i \in \{1, 2\}$.[3] This way the algorithms interacting with the oracle can manipulate the encodings of group elements via the oracle Add.

To simulate a GGM oracle efficiently, the simulator dynamically generates the encoding function f. More specifically, it maintains a partial set L of $\mathbb{Z}_p$-label pairs sampled by the simulator so far. (It is initially empty.) Whenever a query $(a_1, a_2, \ell_1, \ell_2)$ is made, the simulator checks if $(*, \ell_1) \in L$ (meaning that if for some v_1, $(v_1, \ell_1) \in L$); if not, the simulator samples a random v_1 from $\mathbb{Z}_p$ subject to $(v_1, *) \notin L$, and adds (v_1, ℓ_1) to L. The simulator does the same thing for ℓ_2. Now assuming $(v_1, \ell_1) \in L$ and $(v_2, \ell_2) \in L$, letting $a_3 = a_1 v_1 + a_2 v_2$ if $(a_3, \ell_3) \in L$ for some ℓ_3, the simulator responds to the query with ℓ_3; else, the simulator samples a random ℓ_3 subject to $(*, \ell_3) \notin L$, adds (a_3, ℓ_3) to L, and responds to the query with ℓ_3. L can be thought of a lazily sampled version of f.

Other simulatable oracles that are useful in our main theorem are black-box oblivious transfer [GKM+00], black-box public-key encryption [GKM+00], and ideal obfuscation [JLLW23].

3.2.2 Proof Sketch of Main Theorem

The observation that leads to the main theorem is that in PIR protocols the server does not need to have privacy. This means the protocol would still be secure if the user learned all of the server's oracle queries. Also, the user knows all of its own oracle queries. Therefore, the user can just simulate the oracle for both of the parties. For this the server just has to send all of its queries to the user. This modification increases the server communication roughly by the amount that the server would have communicated to the oracle. We will prove that this transformation preserves user security. Since we require the server's 'oracle communication' to be $o(n)$ and the server's protocol communication to be $< cn$ for some $c < 1$, the modified protocol will have no oracle queries and the server communication will be $< \overline{c}n$ for $\overline{c} < 1$. Moreover, in the actual compiled protocol, to enable the compiled user to distinguish query messages from normal protocol messages, we append a flag bit to the end of each server's protocol messages — causing a dependency on r, the number of rounds, in Theorem 3.1.1.

3.2.3 PIR Related Protocols

Now we exhibit a few protocols which imply non-trivial PIR, allowing us to apply our PIR impossibility results to these protocols.

Low Sender-Communication OT We show that an ℓ-batch k-bit OT protocol with sender communication $< c2k\ell$ for $c < 1$ implies a PIR. The folklore transformation works as follows: suppose w.l.o.g the database size is $2k\ell$. The server runs the OT protocol and encodes the first half of the database into the messages $(m_i^{(0)})_{i\in[\ell]}$ and the second half into $(m_i^{(1)})_{i\in[\ell]}$. Now, if a user wants to look up the j-th element of the database, it acquires $(m_i^{(0)})_{i\in[\ell]}$ if $j \le k\ell$ and $(m_i^{(1)})_{i\in[\ell]}$ otherwise. The database entry that the client wants to

[3] One may set S to be a random subset of size p of a larger set $\{0,1\}^u$ for $u > \log p$. Our analysis will remain unchanged, so we simply assume $S = \{0, \ldots, p-1\}$. See [Zha22] for differences between various models.

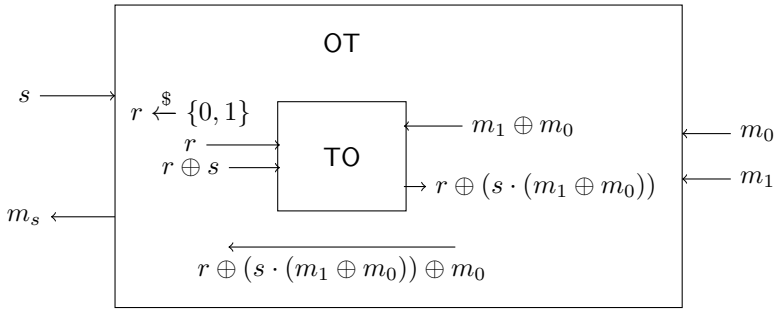

Figure 3.1: Similar to [WW06] a visual representation of how to build an oblivious transfer OT from an oblivious transfer TO that goes in the opposite direction.

learn is contained in the OT output. The server communication is $< c2k\ell$ and the user's input j is hidden from the server by the OT's receiver security.

Low Total Communication OT. We next attempt to prove lower-bounds for the case where the OT protocol has low total communication (as opposed to low sender communication). For convenience we focus on ℓ-batch single-bit OT. By low total communication we mean an amount that is close to the information-theoretically optimal communication, which is 2ℓ bits. The techniques for low sender-communication as above do not apply outright because the sender might cause the bulk of communication itself, an amount close to 2ℓ bits (e.g., suppose the sender's communication is $2\ell + \lambda$ bits and the receiver's communication is λ bits). We get around this issue via the following intuitive idea: When an OT protocol has low total communication it must have either low sender communication, from which we already showed how to obtain a non-trivial PIR, or it must have low receiver communication. In the latter case, we swap the roles of the two different parties with a role-flipping trick, implicitly used in [IKNP03] and explicitly in [WW06]. This role flipping trick turns the low receiver communication into low sender communication, which we can then turn into a non-trivial PIR. Figure 3.1 depicts the construction for $\ell = 1$. In essence, we show how to turn a communication-efficient OT into a sender-communication-efficient OT by introducing an additioal round.

OT Extension. The above results immediately imply communication lower bounds for OT extension: showing that performing OT extension for ℓ-batch k-bit OTs with $c2k\ell$ bits of sender communication for $c < 1$, and with an $O(\lambda)$ (and even $o(k\ell)$) number of public-key operations is impossible.

Unbalanced PSI. Private set intersection (PSI) is an MPC protocol between two parties each holding a set and the party called the receiver learns the intersection of the two sets. No other information should be revealed to is to any of the parties. In Unbalanced PSI, a special case of PSI, the receiver set is much smaller than the sender set and the communication should only scale with the receiver set. To build a non-trivial PIR from such a protocol, for a client index i, the client sets $x := i$ (padding it out if necessary), and for a database DB, the server forms the set $\{i \mid \mathsf{DB}[i] = 1\}$. An answer to $x \in^? S$ reveals $\mathsf{DB}[i]$. This observation allows us to prove that in unbalanced PSI with sub-linear communication, the sender should perform close to linear public-key operations. This shows that the large number of public-key

operations used in unbalanced PSI protocols of [DGI+19, GHO20, CGH+21] is inherent.

Non-Trivial PIR implies Oblivious Transfer We know that non-trivial PIR implies oblivious transfer [DMO00], and this is used to get our final impossibility results. The transformation utilizes the user security of the PIR protocol and deploys a compression argument to argue information loss. The entropy garnered from the information loss is then fed into a randomness extractor, the output of which can be used to guarantee sender security in the resulting oblivious transfer protocol.

3.2.4 Oracles

Notice that all these transformations only make black-box use of the protocols they transform. This means that if the starting protocol uses some oracle other than the one we want to remove, say the random oracle [BR93], then the resulting protocol will also use the random oracle even if we remove other oracle queries.

3.3 Related Work

Techniques Our 'compiling-out' techniques bear some similarities to ideas used by Gennaro and Trevisan [GT00] for giving lower-bounds on the query complexity of PRGs from OWPs. Essentially, they showed that if the number of queries is 'small', they can be encoded as part of the input, hence getting rid of OWP calls in the construction of a PRG. Gennaro et al. [GGK03] built on that idea to give lower-bounds on the efficiency of various cryptographic primitives. These works mostly deal with non-interactive primitives. Our techniques are used in a different way in that we leverage the lack of security requirements for a party to get rid of oracle calls of an interactive protocol.

Private-Information Retrieval. In all but this section of the paper we talk about non-trivial single-server private information retrieval, which is why we will sometimes leave out the descriptor "single-server". Traditionally, PIR [CGKS95, KO00] is a protocol between one user and possibly multiple servers. Just like in the non-trivial single-server case the user with an index $i \in [n]$ learns the i-th element of a database $\mathsf{DB} \in \{0,1\}^n$ held by all the servers without disclosing i to the servers. If the caveat of non-triviality is not made, then not only the server communication needs to be sub-linear in n, but also the total communication. Thus, a non-trivial PIR (requiring only the server-to-user communication to be small, as ruled out in our lower-bounds) is a weaker primitive than PIR. Multi-server PIR protocols assume some kind of non-collusion between the servers, which allows them to achieve statistical security as opposed to computational security.

By now PIR is a well studied primitive; here we focus on the single server setting. We know how to build PIR with communication complexity polylogarithmic in n from a wide range of assumptions [CMS99, IP07, DGI+19, CGH+21]. In the last few years, we have also made progress towards practically efficient PIR [CK20, KC21, SACM21, CHK22, ZLTS23, ZPZS24, HHC+23] and asymptotically efficient PIR [CHR17, BIPW17, LMW23] when the server and the client (or sometimes only the server) are allowed to preprocess the database. We even know some lower bounds for different preprocessing settings [BIM00, CK20, CHK22, PY22, Yeo23].

OT Extension. The intuition behind OT extension is that it only uses very few calls to an OT functionality to implement many more OTs. An equivalent description is that an OT extension protocol is an OT protocol that can make calls to an OT functionality. The protocol becomes valuable if the number of OT calls in the protocol is much less than the 'size' of the OT being implemented. OT calls are typically modelled as oracle calls to an OT functionality or the OT hybrid model.

Beaver [Bea96] constructed the first OT extension protocol, which makes non-black-box use of pseudorandom generators and which has two rounds. Ishai et al. [IKNP03] give the first OT extension protocol only making black-box use of symmetric-key cryptography while increasing the rounds to three. Garg et al. [GMMM18] show that three rounds are necessary in the OT hybrid model when only making black-box use of symmetric-key cryptography.

Rate-1 String OT. The notion of rate-1 OT has applications beyond the construction of PIR with polylog communication. In particular, a generalization of this notion, called trapdoor hash, has been used as a building block to build non-interactive zero knowledge for NP [BKM20]. This has made the notion of rate-1 OT appealing from both a theoretical and practical points of view.

Unbalanced Private-Set Intersection (PSI). Private keyword search allows a receiver, with a single element x, to learn whether x is a member of a large set S held by a sender, or sometimes called PIR for keywords [CGN98]. This is an instance of the so-called unbalanced PSI problem, defined earlier. A desirable feature of such unbalanced PSI protocols is sub-linear communication: the total amount of communication must be sub-linear the larger set size. We have protocols, from a wide variety of cryptographic assumptions, for unbalanced PSI whose communication complexity grows only polylogarithmically with the larger set size [IP07, DGI$^+$19, GHO20, CGH$^+$21].

Again, the Diffie-Hellman-based protocols come with a high sender computation cost: the number of group operations grows at least linearly in the bigger set size. As in PIR, one can prove that the strict running time of the sender in unbalanced PSI cannot be sub-linear in $|S_1|$, but that does not mean the number of public-key operations must also grow with n — especially, if the sender is allowed to make an arbitrarily-large number of symmetric-key operations. In fact, while the protocols in [DGI$^+$19, GHO20, CGH$^+$21] induce little communication, the large number of public-key operations involved is a major bottleneck.

In the absence of the sub-linear communication requirement, one may use oblivious-transfer (OT) extension techniques [Bea96, IKNP03] to design unbalanced PSI protocols with a number of public-key operations independent of $|S_1|$. These protocols can be made concretely efficient as well (e.g., [CM20]). However, all these OT-extension-based protocols fail to achieve sub-linear communication, and our lower-bound results explain this situation.

Subsequent Work In subsequent work, [LMW25] built upon our techniques to prove that black-box use of cryptography does not help with building doubly efficient PIR. This provides a strong indication that the heavy non-black box use of Ring-LWE in [LMW23] was necessary.

3.4 Protocols that Imply Non-Trivial PIR

We substantiate the relevance of non-trivial PIR by proving that communication-efficient versions of some popular MPC protocols can be transformed into non-trivial PIR in a black-

box manner. These transformations later let us transfer the lower-bounds regarding PIR to these protocols.

In the following, we focus on different variants of oblivious transfer and unbalanced private set intersection to demonstrate the concept. The same ideas apply to many other protocols such as vector oblivious linear evaluation and oblivious polynomial evaluation.

3.4.1 Oblivious Transfer

We show how to transform a protocol for k-bit string oblivious transfer $\mathsf{OT} = (\mathsf{OTR}, \mathsf{OTS})$ that makes calls to an oracle $\mathcal{O}$ into a PIR protocol $(\mathsf{PIRU}, \mathsf{PIRS})$ with database size $n = 2k$ that makes calls to the same oracle $\mathcal{O}$ in a black-box manner. The construction is folklore and works by splitting the database in half, using each half as one of the two strings, and choosing the OT choice bit based on the PIR client's index accordingly.

- $\mathsf{PIRU}^{\mathcal{O}}(1^\lambda, 1^n, i)$: For $n = 2k$, set the choice bit $b \leftarrow \lfloor (i-1)/k \rfloor$. Run the OT receiver $m_b \leftarrow \mathsf{OTR}^{\mathcal{O}}(1^\lambda, b)$ to get the chosen string m_b. Return $m_b[i - kb]$
- $\mathsf{PIRS}^{\mathcal{O}}(1^\lambda, 1^n, \mathsf{DB})$: Let strings $m_0 \leftarrow \mathsf{DB}[1, \ldots, k]$ and $m_1 \leftarrow \mathsf{DB}[k+1, \ldots, 2k]$. Run the OT sender $\mathsf{OTS}^{\mathcal{O}}(1^\lambda, m_0, m_1)$.

Lemma 3.4.1 (Folklore)**.** *The PIR protocol* $(\mathsf{PIRU}, \mathsf{PIRS})$ *has the same correctness error, the same sender/receiver query complexity, and the same sender/receiver communication as those of the OT protocol* OT*.*

That means if the expected sender communication in OT *is less than* $n = 2k$*, then* $(\mathsf{PIRU}, \mathsf{PIRS})$ *is a non-trivial PIR protocol.*

Proof of Correctness. Correctness follows from the correctness of the OT protocol and for $b = \lfloor (i-1)/k \rfloor$ we have $m_b[i - kb] = \mathsf{DB}[kb + 1, \ldots, kb + k][i - kb] = \mathsf{DB}[i]$. □

Proof of Client Security. Suppose there exists $i, i' \in [N]$, $\mathsf{DB} \in \{0,1\}^N$ such that an adversary $\mathcal{A}$ can distinguish $\mathsf{view}_s^{\mathsf{PIR}}(i, \mathsf{DB})$ from $\mathsf{view}_s^{\mathsf{PIR}}(i', \mathsf{DB})$ with non-negligible probability. Then $\lfloor (i-1)/k \rfloor \neq \lfloor (i'-1)/k \rfloor$ else $\mathsf{view}_s^{\mathsf{PIR}}(i, \mathsf{DB})$ and $\mathsf{view}_s^{\mathsf{PIR}}(i', \mathsf{DB})$ follow the exact same distribution. The same adversary $\mathcal{A}$ distinguishes between $\mathsf{view}_s^{\mathsf{OT}}(\lfloor (i-1)/k \rfloor, (\mathsf{DB}[1, \ldots, k], \mathsf{DB}[k+1, \ldots, 2k]))$ and $\mathsf{view}_s^{\mathsf{OT}}(\lfloor (i'-1)/k \rfloor, (\mathsf{DB}[1, \ldots, k], \mathsf{DB}[k+1, \ldots, 2k]))$ with the same non-negligible probability since the views are exactly the same as $\mathsf{view}_s^{\mathsf{PIR}}(i, \mathsf{DB})$ and $\mathsf{view}_s^{\mathsf{PIR}}(i', \mathsf{DB})$ respectively. □

Remark 3.4.2. One can transform any ℓ-batch k-bit OT protocol into a $k\ell$-bit string OT protocol by reusing the same choice bit across all the ℓ batches. This works without any issues because we only talk about semi-honest security.

OT With Low Total Communications. Using the symmetric nature of OT we transform an OT protocol with low communication (not just low sender communication) into a low sender communication OT protocol in a black-box manner. This allows us to apply our PIR lower-bounds to OT with low expected communication. Our transformation works by noting that every communication efficient OT protocol has either low sender or low receiver communication; if the receiver communication is low, our transformation will swap the roles of the sender and receiver, to obtain an OT protocol with low sender communication, as desired.

The following transformation was implicitly used in [IKNP03] and explicitly in [WW06]. The transformation works as follows: Let $\mathsf{OT} = (\mathsf{OTR}, \mathsf{OTS})$ be an ℓ-batch single-bit OT with expected total communication $t(\lambda, \ell)$, expected download communication $d(\lambda, \ell)$, expected upload communication $u(\lambda, \ell)$, and oracle accesses to $\mathcal{O}$. We define a $\mathsf{OT}' = (\mathsf{OTR}', \mathsf{OTS}')$ as follows

$\mathsf{OTR}'^{\mathcal{O}}(1^\lambda, 1^\ell, s)$:

1. If the expected download communication of OT is $d(\lambda, \ell) < u(\lambda, \ell) + \ell$:
   (a) Run $(m'_1, \ldots, m'_\ell) \leftarrow \mathsf{OTR}^{\mathcal{O}}(1^\lambda, 1^\ell, s)$, the ℓ-batch OT receiver on the choice string s.
   (b) Return $(m'_1, \ldots, m'_\ell)$
2. Else:
   (a) Sample $r \xleftarrow{\$} \{0,1\}^\ell$ uniformly at random.
   (b) Run $\mathsf{OTS}^{\mathcal{O}}(1^\lambda, 1^\ell, r, s \oplus r)$, the ℓ-batch OT sender on messages $m_0 = r$ and $m_1 = r \oplus s$.
   (c) Receive v in the round after OTS is done.
   (d) Return $v \oplus r$

$\mathsf{OTS}'^{\mathcal{O}}(1^\lambda, 1^\ell, m^{(0)}, m^{(1)})$:

1. If the expected download communication of OT is $d(\lambda, \ell) < u(\lambda, \ell) + \ell$:
   (a) Run $\mathsf{OTS}^{\mathcal{O}}(1^\lambda, 1^\ell, m^{(0)}, m^{(1)})$, the ℓ-batch OT sender on messages $m_0 = m^{(0)}$ and $m_1 = m^{(1)}$.
   (b) Return
2. Else:
   (a) Run $z \leftarrow \mathsf{OTR}^{\mathcal{O}}(1^\lambda, 1^\ell, m^{(1)} \oplus m^{(0)})$, the ℓ-batch OT receiver on the choice string $m^{(1)} \oplus m^{(0)}$ to receive the string z.
   (b) Send $z \oplus m^{(0)}$ in the round after OTR is done
   (c) Return

Lemma 3.4.3. *The constructed OT protocol* $\mathsf{OT}' = (\mathsf{OTR}', \mathsf{OTS}')$ *has the same correctness as the base OT* $\mathsf{OT} = (\mathsf{OTR}, \mathsf{OTS})$. *Moreover,* $\mathsf{OT}' = (\mathsf{OTR}', \mathsf{OTS}')$ *is secure if* $\mathsf{OT} = (\mathsf{OTR}, \mathsf{OTS})$ *is secure.*

Assuming $\mathsf{OT} = (\mathsf{OTR}, \mathsf{OTS})$ *has expected overall rate* $r > 2/3$*, the constructed OT has expected download rate* $w > 1/2$.

Correctness. If $d(\lambda, \ell) < u(\lambda, \ell) + \ell$ then both parties behave exactly like OT and therefore correctness is inherited.

If $d(\lambda, \ell) \geq u(\lambda, \ell) + \ell$ the sender OTS' learns

$$\left(r_1 \oplus s_1 \cdot (m_1^{(1)} \oplus m_1^{(0)}), \ldots, r_\ell \oplus s_\ell \cdot (m_\ell^{(1)} \oplus m_\ell^{(0)})\right)$$

it then sends back

$$\left(r_1 \oplus s_1 \cdot (m_1^{(1)} \oplus m_1^{(0)}) \oplus m_1^{(0)}, \ldots, r_\ell \oplus s_\ell \cdot (m_\ell^{(1)} \oplus m_\ell^{(0)}) \oplus m_\ell^{(0)}\right)$$

then the receiver OTR' computes

$$\begin{aligned}&\left(s_1 \cdot (m_1^{(1)} \oplus m_1^{(0)}) \oplus m_1^{(0)}, \ldots, s_\ell \cdot (m_\ell^{(1)} \oplus m_\ell^{(0)}) \oplus m_\ell^{(0)}\right)\\ =&\left(m_1^{(s_1)}, \ldots, m_\ell^{(s_\ell)}\right)\end{aligned}$$

□

Security. The security in the case that $d(\lambda, \ell) < u(\lambda, \ell) + \ell$ directly follows from the security of OT.

In the other case it follows from the security of OT and the work of [WW06] which proves that this exact construction is secure. For receiver security we have that the sender (according to sender security of OT) only learns z. Each bit z_i is either r_i or $s_i \oplus r_i$, in both cases it is uniformly random because r is uniformly random. Sender security of OT' follows because the execution of OT leaks nothing to the receiver (according to the receiver security of OT). That means all the receiver learns is $z \oplus m^{(0)}$. By correctness of OT' this is exactly $(m_1^{(s_1)} \oplus r_1, \ldots, m_\ell^{(s_\ell)} \oplus r_\ell)$ and therefore contains no information about $(m_1^{(1-s_1)}, \ldots, m_\ell^{(1-s_\ell)})$. □

Expected Download Communication. Let r be the rate, $t(\lambda, n)$ be the expected total communication which is the sum of the expected receiver-to-sender communication $u(\lambda, \ell)$ and the expected sender-to-receiver communication $d(\lambda, \ell)$. Then for all but finitely many ℓ we have $t(\lambda, \ell) < \frac{2\ell}{r}$. In the following, the expected sender-to-receiver communication of OT' will be called $d'(\lambda, \ell)$.

If $d(\lambda, \ell) < u(\lambda, \ell) + \ell$ then

$$\begin{aligned}d'(\lambda, \ell) = d(\lambda, \ell) = t(\lambda, \ell) - u(\lambda, \ell) \leq t(\lambda, \ell) - d(\lambda, \ell) + \ell \qquad &\Leftrightarrow\\ d'(\lambda, \ell) \leq \frac{t(\lambda, \ell) + \ell}{2}&\end{aligned}$$

Else the new expected sender-to-receiver communication is

$$\begin{aligned}d'(\lambda, \ell) = u(\lambda, \ell) + \ell = t(\lambda, \ell) - d(\lambda, \ell) + \ell \leq t(\lambda, \ell) - u(\lambda, \ell) - \ell + \ell \qquad &\Leftrightarrow\\ d'(\lambda, \ell) \leq \frac{t(\lambda, \ell) + \ell}{2}&\end{aligned}$$

Either way, $d'(\lambda, \ell) \leq \frac{t(\lambda,\ell)+\ell}{2}$ which means that for all but finitely many ℓ we have $d'(\lambda, \ell) < (\frac{1}{r} + \frac{1}{2})\ell$. Therefore, the expected download rate is $\frac{2}{(\frac{1}{r}+\frac{1}{2})}$ which is $> 1/2$ for $r > 2/3$. □

3.4.2 Unbalanced Private-Set Intersection

In unbalanced private set intersection we have a set A of n λ-bit messages, held by a sender $\mathsf{PSIS}(1^\lambda, 1^n, A)$, and a singleton set B, held by a receiver $\mathsf{PSIR}(1^\lambda, 1^n, B)$. The goal is for the receiver to learn $A \cap B$ while the sender should learn nothing. Semi-honest receiver security can be defined along the lines of receiver (client) security of PIR (Definition 2.4.7).

We show how to transform a protocol for unbalanced private-set intersection $\mathsf{PSI} = (\mathsf{PSIS}, \mathsf{PSIR})$ that makes calls to oracle $\mathcal{O}$ into a PIR protocol $(\mathsf{PIRU}, \mathsf{PIRS})$ that makes calls to the same oracle $\mathcal{O}$ in a black-box manner. We do this by simply encoding the PIR-database and the PIR-query as sets.

- $\mathsf{PIRU}^{\mathcal{O}}(1^\lambda, 1^n, i)$: Set $A := \{i\}$. Run the PSI receiver $I \leftarrow \mathsf{PSIR}^{O}O(1^\lambda, A)$ to get the intersection I. Return 1 if $\{i\} = I$ and 0 otherwise.
- $\mathsf{PIRS}^{\mathcal{O}}(1^\lambda, 1^n, \mathsf{DB})$: Form the set $B := \{x \mid \mathsf{DB}[x] = 1\}$. Run the PSI sender $\mathsf{PSIS}^{\mathcal{O}}(1^\lambda, B)$.

Lemma 3.4.4 (Folklore)**.** *The PIR protocol* $(\mathsf{PIRU}, \mathsf{PIRS})$ *has the same correctness error, sender and receiver communication and sender and receiver query complexity as those of* PSI*. Moreover, the resulting PIR protocol has client security if* PSI *provides receiver security.*

That means if the sender communication in PSI *is less than* n*, then the protocol* $(\mathsf{PIRU}, \mathsf{PIRS})$ *is a non-trivial PIR protocol.*

Proof of Correctness. Correctness follows from the correctness of the PSI protocol and the intersection of $\{i\}$ and B being $\{i\}$ if $i \in B \Leftrightarrow \mathsf{DB}[i] = 1$ and $\emptyset$ otherwise.

Proof of Client Security. Suppose there exists $i, i' \in [n]$, $\mathsf{DB} \in \{0,1\}^n$ such that an adversary $\mathcal{A}$ can distinguish $\mathsf{view}_s^{\mathsf{PIR}}(i, \mathsf{DB})$ from $\mathsf{view}_s^{\mathsf{PIR}}(i', \mathsf{DB})$ with non-negligible probability. The same adversary $\mathcal{A}$ distinguishes between $\mathsf{view}_s^{\mathsf{PSI}}(\{i\}, B)$ and $\mathsf{view}_s^{\mathsf{PSI}}(\{i'\}, B)$ with the same non-negligible probability since the views are exactly the same. □

3.5 Lower-Bounds on the Oracle Queries in PIR

In this section we show how to transform a private information retrieval (PIR) protocol with access to some *simulatable* oracle SO into one that does not query that oracle. To have something concrete in mind one may imagine SO being the generic group model, though the technique is more general. We will later go into common instantiations of the oracle SO. This transformation allows us to transfer lower-bounds from PIR without oracle access to PIR with oracle access.

Simulatable Oracles. A simulatable oracle is an oracle SO which can efficiently be simulated by a stateful simulator Sim. More formally, a computationally unbounded adversary $\mathcal{A}$ cannot win the following game with a non-negligible advantage in polynomially many rounds r, where Sim is a PPT algorithm:

1. Sample random bit $b \xleftarrow{\$} \{0,1\}$.
2. Initialize the state of the oracle as $\mathsf{st} \leftarrow \bot$.
3. Initialize the state of the adversary $\mathsf{ast} \leftarrow \bot$.
4. The adversary produces a first query qu.
5. For $i \in [r]$:
   (a) If $b = 0$:
      - Let the response be $\mathsf{resp} \leftarrow \mathsf{SO}(\mathsf{qu})$
   (b) Else:
      - Let response and new oracle state be $(\mathsf{resp}, \mathsf{st}) \leftarrow \mathsf{Sim}(\mathsf{qu}, \mathsf{st})$
   (c) Let new query and adversary state be $(\mathsf{qu}, \mathsf{ast}) \leftarrow \mathcal{A}(\mathsf{ast}, \mathsf{resp})$

6. Let the adversary output its guess $b' \leftarrow \mathcal{A}(\mathsf{ast})$.

7. The adversary wins if $b = b'$.

Typical examples of simulatable oracles include the random oracle and the generic group oracle.

Construction 3.5.1. Let $\mathsf{PIR} := (\mathsf{PIRU}^{\mathsf{SO},\mathsf{O}}, \mathsf{PIRS}^{\mathsf{SO},\mathsf{O}})$ be a bit PIR protocol that uses a simulatable oracle SO and another oracle O. We show how to compile out the SO-calls of $(\mathsf{PIRU}^{\mathsf{SO},\mathsf{O}}, \mathsf{PIRS}^{\mathsf{SO},\mathsf{O}})$, obtaining an SO-free PIR protocol $\overline{\mathsf{PIR}} := (\overline{\mathsf{PIRU}}^{\mathsf{O}}, \overline{\mathsf{PIRS}}^{\mathsf{O}})$.

For notational convenience, in the following whenever calling PIRU or PIRS, we omit the private-state part of the input.

The protocol messages sent from $\overline{\mathsf{PIRU}}$ to $\overline{\mathsf{PIRS}}$ are tagged with either 'protocol' (or bit zero) signifying a normal protocol message, or with 'query' (or bit 1) signifying a query message.

$\overline{\mathsf{PIRU}}^{\mathsf{O}}(1^\lambda, 1^n, i)$:

- Initialize the state of the simulatable oracle $\mathsf{st} \leftarrow \bot$.
- Run the interactive PPT $\mathsf{PIRU}^{\mathsf{SO},\mathsf{O}}$ with the following interactions:
  1. When $\mathsf{PIRU}^{\mathsf{SO},\mathsf{O}}$ calls O on a query qu forward the query to O and respond with the received response.
  2. When $\mathsf{PIRU}^{\mathsf{SO},\mathsf{O}}$ calls SO on a query qu, simulate the response and update the oracle simulators state $(\mathsf{resp}, \mathsf{st}) \leftarrow \mathsf{Sim}(\mathsf{qu}, \mathsf{st})$.
  3. Upon $\overline{\mathsf{PIRU}}$ receiving a message of the form $(\text{`}query\text{'}, \mathsf{msgs})$, interpret msgs as a query qu, simulate the oracle response and update the oracle simulators state as $(\mathsf{resp}, \mathsf{st}) \leftarrow \mathsf{Sim}(\mathsf{qu}, \mathsf{st})$ and return resp to the sender $\overline{\mathsf{PIRS}}$. If the message has the form $(\text{`}protocol\text{'}, \mathsf{msgs})$, run $\mathsf{PIRU}^{\mathsf{SO},\mathsf{O}}$ on the protocol message msgs until it produces the next message msgr and send that to $\overline{\mathsf{PIRS}}$. The oracle queries are handled as described above.

$\overline{\mathsf{PIRS}}^{\mathsf{O}}(1^\lambda, 1^n, \mathsf{DB})$:

- Run the interactive PPT $\mathsf{PIRS}^{\mathsf{SO},\mathsf{O}}$ with the following interactions:
  1. When $\mathsf{PIRS}^{\mathsf{SO},\mathsf{O}}$ calls O on a query qu, forward the query to O and respond with the received response.
  2. When $\mathsf{PIRS}^{\mathsf{SO},\mathsf{O}}$ calls SO on a query qu, send a tagged query pair $(\text{`}query\text{'}, \mathsf{qu})$ to $\overline{\mathsf{PIRU}}$ and use the response msgr as a query response for qu to PIRS.
  3. Else, run $\mathsf{PIRS}^{\mathsf{SO},\mathsf{O}}$ until it produces a message msgs and sends the tagged message $(\text{`}protocol\text{'}, \mathsf{msgs})$ to $\overline{\mathsf{PIRU}}$, then wait for the response msgr and continue.

Theorem 3.5.2. *If* PIR *is a non-trivial private information retrieval with server communication of* $\eta < cn$ *for* $c < 1$, $r \in o(n)$ *rounds of interaction with the user, and* $q \in o(n)$ *bits of communication with the* SO *oracle then* $\overline{\mathsf{PIR}}$ *is a non-trivial private information retrieval with server communication* $\overline{\eta} \leq \overline{c}n$ *for* $\overline{c} < 1$ *and no calls to* SO.

Server communication. The server's additional communication overhead includes 1 bit per round as well as a total of $O(q)$ bits. Since the number rounds is $o(n)$, the total server communication complexity becomes $cn + o(n)$, which is less than $\overline{c}n$ for some $\overline{c} < 1$. □

Correctness. Notice that the above protocol will have different output from an execution of PIR either

1. if a with Sim simulated oracle behave differently from the real oracle behaviour or
2. if a message in the execution of PIR happens to start with t.

Both of these events happen with negligible probability. Therefore, if PIR has statistical correctness then so does $\overline{\mathsf{PIR}}$. □

Client Security. Suppose there exists $i, i' \in [n]$, $\mathsf{DB} \in \{0,1\}^n$ such that an adversary $\overline{\mathcal{A}}$ can distinguish $\mathsf{view}_s^{\overline{\mathsf{PIR}}}(i, \mathsf{DB})$ from $\mathsf{view}_s^{\overline{\mathsf{PIR}}}(i', \mathsf{DB})$ with non-negligible probability. We construct a new adversary $\mathcal{A}$ to distinguish between $\mathsf{view}_s^{\mathsf{PIR}}(i, \mathsf{DB})$ and $\mathsf{view}_s^{\mathsf{PIR}}(i', \mathsf{DB})$. The new adversary $\mathcal{A}$ gets as input a view v either from $\mathsf{view}_s^{\mathsf{PIR}}(i, \mathsf{DB})$ or $\mathsf{view}_s^{\mathsf{PIR}}(i', \mathsf{DB})$ and does the following:

1. Generate an empty view $\overline{v}$.
2. Copy all O-oracle calls from v to $\overline{v}$.
3. Run PIRS on the randomness and DB as defined in the view v and simulate its interaction as follows:
   (a) For PIRS's calls to the SO oracle with query qu and gets response resp enter $('query', \mathsf{qu})$ as a server message into $\overline{v}$ and resp as a user message.
   (b) For PIRS's messages msgs enter $('protocol, \mathsf{msgs})$ in the transcript $\overline{v}$ as a server message and enter the users response msgr as a users message.
4. Run $b \leftarrow \overline{\mathcal{A}}(\overline{v})$, on the view $\overline{v}$ produced by $\overline{\mathsf{PIRS}}$
5. Return b

$\mathcal{A}$ will distinguish $\mathsf{view}_s^{\mathsf{PIR}}(i, \mathsf{DB})$ and $\mathsf{view}_s^{\mathsf{PIR}}(i', \mathsf{DB})$ with negligibly close to the probability as $\overline{\mathcal{A}}$ can distinguish $\mathsf{view}_s^{\overline{\mathsf{PIR}}}(i, \mathsf{DB})$ from $\mathsf{view}_s^{\overline{\mathsf{PIR}}}(i', \mathsf{DB})$. This is because $\overline{v}$ follows the same distribution as $\mathsf{view}_s^{\overline{\mathsf{PIR}}}(i, \mathsf{DB})$ (except that the SO queries are produced by the real oracle, not the simulator) if v was from $\mathsf{view}_s^{\mathsf{PIR}}(i, \mathsf{DB})$ and $\overline{v}$ follows the same distribution as $\mathsf{view}_s^{\overline{\mathsf{PIR}}}(i', \mathsf{DB})$ (same caveat here) if v was from $\mathsf{view}_s^{\mathsf{PIR}}(i', \mathsf{DB})$. If the adversary could notice the simulation of SO then it would break its simulatability. □

Remark 3.5.3. Theorem 3.5.2 is applicable to any two-PC protocol with one-sided receiver security. Of course, in the absence of further restrictions, such protocols are trivial to realize (e.g., by the sender sending its input in the clear to the receiver). One restriction that makes the problem non-trivial is to require the sender-to-receiver communication to be sub-linear in the sender's input size, as in PIR.

The utility of Theorem 3.5.2, beyond PIR itself, becomes apparent when one considers other protocols that imply non-trivial PIR while instantiating their underlying oracles via ideal forms of powerful primitives. We first discuss the implications of the theorem in terms of particular instantiations of the oracle, and in the next section we consider protocols that imply PIR.

Theorem 3.5.2 allows us to also rule out powerful non-black-box techniques for building PIR. We demonstrate this by letting SO include an OT oracle and an ideal obfuscation oracle that can obfuscate circuits with generic OT gates and random oracle gates. (See [AS15b, GHMM18] for capturing similar non-black-box techniques via oracle-aided circuits.)

Corollary 3.5.4. *For any constants $c < 1$, there exists no n-bit PIR protocol with server communication $\eta \leq cn$, round complexity $r \in o(n)$, and with oracle access to a PSPACE-complete oracle, a random oracle, a generic OT oracle, and an obfuscation oracle for circuits with OT and random oracle gates, and where the server only communicates $q \in o(n)$ bits to the ideal obfuscation and OT oracles.*

Proof. In Lemma 3.4.1 we show an OT protocol with the above mentioned characteristics implies a non-trivial PIR. Let SO consist of an OT oracle [GKM$^+$00] and an ideal obfuscation oracle [AS15b, JLLW23] for obfuscating circuits with OT/RO gates. (Such an SO oracle is simulatable.) By invoking Theorem 3.5.2 one gets a non-trivial PIR with oracle access to the random oracle and an PSPACE-complete oracle. This in turn can be transformed into an OT protocol (while retaining the O oracles) via [DMO00]. The existence of such an object however was ruled out by [GKM$^+$00]. □

Back to the black-box setting, other illustrative examples include the use of GGMs for building non-trivial PIR.

Corollary 3.5.5. *For any constants $c < 1$, a non-trivial n-bit PIR protocol with server communication of cn, round complexity $r \in o(n)$ and where the server makes sublinear in n many generic group queries requires MPC-hard assumptions, beyond the generic group.*

Proof. In Theorem 3.5.2, if one instantiates SO by a generic group [Sho97] and let the O oracle be empty, then one gets a non-trivial PIR without any oracle calls. This in turn can be transformed into an OT protocol without any oracles via [DMO00]. OT is an MPC-complete protocol. □

The above corollary is almost tight as there exists GGM-based PIR protocols with a linear number of GGM queries.

Lemma 3.5.6 ([DGI$^+$19])**.** *Based on the DDH assumption, there exists a non-trivial n-bit PIR protocol with server communication of $O(\lambda)$ and with the sever making $O(n)$ group operations.*

Finally, we may derive a statement for FHE oracles.

Corollary 3.5.7. *For any constants $c < 1$, a non-trivial n-bit PIR protocol with server communication of cn, round complexity $r \in o(n)$ where the server makes $q \in o(n)$ black-box use of fully homomorphic encryption[4] requires MPC-hard assumptions, even beyond the fully homomorphic encryption.*

Proof. Let SO be an FHE oracle [GMM17] (defined similarly to a generic PKE oracle of [GKM$^+$00]). Let the O oracle be empty. Invoking Theorem 3.5.2 one gets a non-trivial PIR without any oracle calls. This in turn can be transformed into an OT protocol without oracles via [DMO00]. OT is an MPC-complete protocol. □

3.6 Communication Lower-Bounds for OT Extension

Theorem 3.5.2 provides lower-bounds on the computational complexity of PIR protocols. In this section, we show that these computational lower-bounds give rise to communication lower-bounds for OT extension (i.e., the number of bits that an extended OT sender needs to communicate). The result of this section implies that the communication complexity of the sender in the IKNP OT extension protocol [IKNP03] is close to optimal.

[4]This means the server communicates at most q bits to the FHE oracle

Corollary 3.6.1 (OT Extension: Sender Communication Lower-Bound)**.** *For any constants $c < 1$, there exist no ℓ-batch k-bit OT extension protocol with sender communication $\eta < c2k\ell$, round complexity $r \in o(k\ell)$, and with the sender making $q \in o(k\ell)$ OT calls.*

Proof. OT extension is just an OT protocol that makes use of only a black-box OT and a random oracle. An ℓ-batch k-bit OT naturally gives rise to a $k\ell$-bit string OT. In Lemma 3.4.1 we show that such an OT protocol implies a non-trivial PIR for databases of $2\ell k$ bits. Under the resulting PIR protocol, the server communication is $\eta < c2k\ell$, round complexity $r \in o(k\ell)$, and the server communicates a total of $o(k\ell)$ bits with the OT oracle. Invoking Theorem 3.5.2, by instantiating SO with a generic OT oracle [GKM$^+$00] and O with the random oracle [BR93] and by also including a PSPACE-complete oracle, we get a non-trivial PIR with oracle access to the random oracle and a PSPACE-complete oracle. This in turn can be transformed into an OT protocol (while retaining the O oracles) via [DMO00]. The existence of such an object however was ruled out by [GKM$^+$00]. □

Corollary 3.6.2 (OT Extension: Total Communication Lower-Bound)**.** *For any constants $c < 1$, there exist no ℓ-batch k-bit OT extension protocol with total communication $\eta < \frac{3}{2}c(\ell + k\ell)$, round complexity $r \in o(k\ell)$, and with the sender making $q \in o(k\ell)$ OT calls.*

Proof. Follows from Lemma 3.4.3 and Corollary 3.6.1. □

Chapter 4

Designated-Verifier SNARGs

4.1 Introduction

Interactive proofs [GMR89] is a central tool in cryptography and complexity. In an interactive proof system for an NP language L, a prover holding an instance-witness pair (x, w) wants to convince a polynomial-time verifier that $x \in L$, and they do so over multiple rounds of back-and-forth communication. If the statement is true, the verifier should accept; if false, it should accept with probability at most $2^{-\tau}$, no matter what the prover does.

Here, we consider such proof systems with two natural relaxations. First, we only require soundness against computationally bounded provers [BCC88]. Such proof systems are often referred to as *arguments*. Second, we allow the prover and the verifier to engage in an (instance-independent) *preprocessing* protocol. In this setting, we ask the following basic question:

How succinct can a practical proof system be?

By *succinctness*, we refer to the total number of bits exchanged between the prover and the verifier, excluding the preprocessing phase. Originating from the works of Kilian [Kil92] and Micali [Mic94], an enormous body of works studied succinct proof systems that have sublinear communication in the length of the NP-witness.

By *practical*, we loosely refer to proof systems that are practically feasible in the sense that they can run in a "reasonable" amount of time for non-trivial NP-statements, such as proving the satisfiability of a circuit with a few thousand gates. This excludes optimally-succinct proof systems based on general-purpose obfuscation techniques [SW21], which are not yet practical in this sense. To give a more precise notion of practicality, we consider proof systems that can be cast in simple generic models, such as the random oracle model (ROM) [BR93], the generic group model (GGM) [Sho97], or a generic bilinear group model. This is general enough to capture the most succinct proof systems from the literature that are practical in the above informal sense.[1]

Finally, in the above generic models one can typically obtain a *non-interactive* proof system with little to no loss of succinctness. We will therefore restrict our attention to such proof systems, referred to as *succinct non-interactive arguments* (SNARGs) [Mic94, GW11].

[1]The generic models we consider exclude practical *lattice-based* proof systems, e.g., [BISW17, BS23, SSE+24, AFLN24] and many more. However, such proof systems are not competitive with group-based systems in terms of concrete succinctness.

Here, the preprocessing phase may produce a (possibly long and structured) *common reference string* (CRS), which can be used by any prover. SNARGs where the verifier generates the CRS and may keep a secret verification key are referred to as *designated-verifier* SNARGs (dv-SNARGs).

Concrete succinctness of known practical SNARGs. We briefly summarize the sate of the art on practical succinctness and defer a more detailed overview to Section 4.1.3. When referring to concrete proof size, we assume 2^{-80} soundness at a 128-bit security level.[2]

- Using generic bilinear groups, there are SNARGs with 2 $\mathbb{G}_1$-elements and 2 field elements [Gro16, Lip24, DMS24]. When instantiated, these yield 1280-bit proofs.
- In the GGM, [BIOW20] obtain a dv-SNARG with 2 $\mathbb{G}$-elements that yields 512-bit proofs, but whose soundness error is inverse-polynomial in the verifier's running time. Obtaining negligible soundness using the [BIOW20] construction requires a super-constant number of $\mathbb{G}$-elements.

This leaves open two questions: Can we obtain negligible soundness in the GGM with a constant number of group elements? Can we obtain *any practical proof system* that meets the above concrete soundness level using fewer than 1280 bits?

4.1.1 Our Results

We answer both questions in the affirmative. We construct the first designated-verifier SNARGs in the generic group model with negligible soundness error and proof size equating to a constant number of group elements. In fact, our proofs consist of a single group element and an additive term that depends on the soundness error. In concrete terms, we can obtain 2^{-80} soundness at a 128-bit security level using 695 bits, almost a 2x improvement over the best pairing-based SNARGs [Gro16, Lip24, DMS24].

Settling for a constant soundness error (say, 1/2), which may be good enough for some practical use cases, the total proof size is close to a *single* group element, almost a 2x improvement over the best previous dv-SNARGs in this setting [BIOW20].

The above results are obtained via two variants of the same blueprint. We begin by describing a SNARG with one group elements and $O(\tau)$ additional bits, and then discuss its construction.

Theorem 4.1.1 (1 $\mathbb{G}$ + $O(\tau)$ bit dv-SNARG, informal)**.** *Let $\mathbb{G} = \mathbb{G}_\lambda$ be a generic group*[3] *and τ a soundness parameter. There exists a* dv-SNARG *for proving the satisfiability of a Boolean circuit of size s with the following features:*

- *Soundness error: $2^{-\tau} + O(t^2 \cdot 2^{-\lambda})$ against t-query adversaries;*
- *Proof size: 1 $\mathbb{G}$-element (λ bits) and $O(\tau)$ additional bits;*
- *CRS size: $O(\tau s)$ $\mathbb{G}$-elements.*

See Corollary 4.6.1 for a formal statement.

[2] By 2^{-80} soundness at a 128-bit security level we refer to provable soundness error of 2^{-80} against polynomial-time malicious provers in the standard GGM, where we instantiate the group to have 256-bit elements. A similar convention is used in prior related works, see Appendix B.1 for further discussion.

[3] Here $\mathbb{G}_\lambda$ refers to a generic group of size $\approx 2^\lambda$ and whose elements are described using λ bits.

In fact, we prove that our dv-SNARG has the stronger notion of knowledge soundness. While the above $O(\tau)$ term in proof size hides a large constant, we show that if we relax the CRS size to $O(\tau s^2)$, the proof can include 1 $\mathbb{G}$-element and 56τ bits. Thus, for use-cases where only a constant soundness error is required, the proof size in the above dv-SNARG is *smaller than* 2 *group elements*. Finally, if we only require *some constant* soundness error $\delta < 1$, then we can get all the way down to 1 $\mathbb{G}$-element and only 7 additional bits.

We note that even a quadratic CRS size may be tolerable when using our dv-SNARK as an "inner system" for proving the correctness of a fast to verify proof generated by an "outer" SNARG, such as Groth16 [Gro16].

Our prover and verifier both make $\tilde{O}(\lambda\tau s)$ group operations (the prover time becomes $\tilde{O}(\lambda\tau s^2)$ when considering the quadratic CRS variant). We leave a more refined optimization of asymptotic and concrete efficiency to future work, and discuss some possible routes for improvement in Section 4.1.2.

Theorem 4.1.1 is proved by extending the BCIOP compiler [BCI$^+$13], which combines a "linear-only" encryption scheme and a linear PCP to construct dv-SNARGs, to *compressible* encryption schemes, where ciphertexts can be compressed following homomorphic evaluations. Specifically, we consider the *packed* ElGamal encryption scheme implied by techniques developed in [BGI16, DGI$^+$19, BBD$^+$20]. We analyze the malleability of this encryption scheme and design (variants of) linear PCPs that support instantiating the compiler with the packed ElGamal scheme. In more detail, we show that packed ElGamal in the generic group model is *isolated homomorphic*, which is a form of limited homomorphism which allows the adversary more ability than in the linear-only definition of [BCI$^+$13]. We then use packed ElGamal to construct a dv-SNARG using the compiler along with a *strong linear multi-prover interactive proof* (strong LMIP). While in a standard MIP, the verifier interacts with multiple provers which are unable to share information about the verifier's queries, a strong LMIP additionally requires the honest prover strategy to be linear while retaining soundness against malicious provers with arbitrary strategies. To construct strong LMIPs, we use a linear PCP (either the 1-query LPCP of [BHI$^+$24] for linear CRS size or, for better concrete succinctness with quadratic CRS, the Hadamard-based 2-query LPCP from [BIOW20]) and transform it into a strong LMIP. Such a transformation was first used in [IKO07], and we give an alternate construction (for 2-query LPCPs) which achieves better concrete parameters. See Section 4.2 for more details.

Improving concrete proof size via hashing. The primary difficulty in constructing the dv-SNARG of Theorem 4.1.1 is to analyze precisely what power an adversary has in the malleability of the packed ElGamal scheme, and this is the main limitation for achieving smaller proof size. We show that by utilizing a random oracle (which we use only for its collision-resistant properties) it is possible to restrict the adversary's range of actions to a more limited set of malleability attacks. By using this this variant of the packed ElGamal scheme, we design a SNARG whose length is one group element, one output of the random oracle, and a number of bits that tends towards 2τ:

Theorem 4.1.2 (1 $\mathbb{G}$ + 1 $\mathbb{H}$ + $\sim 2\tau$ bit dv-SNARG, informal)**.** *Let $\mathbb{G} = \mathbb{G}_\lambda$ be a generic group, $\mathbb{H} = \mathbb{H}_\lambda$ be a random oracle with λ output bits, τ a soundness parameter, and $p > 2$ be a prime. There exists a* dv-SNARG *for proving the satisfiability of a Boolean circuit of size s with the following features:*

- *Soundness error: $2^{-\tau} + O(t^2 \cdot 2^{-\lambda})$ against t-query adversaries;*
- *Proof size: 1 $\mathbb{G}$-element (λ bits), 1 $\mathbb{H}$ output (λ bits), and $\lceil \frac{2\tau \log p}{\log p - \Theta(1)} \rceil$ additional bits;*

- *CRS size:* $O(\tau s \cdot \mathsf{poly}(p))$ $\mathbb{G}$*-elements.*

See Corollary 4.6.2 for a formal statement.

If we allow the CRS size to be quadratic in s, we can make the constant $\Theta(1)$ in the proof size equal to 1. Thus, for soundness error 2^{-80} at a 128-bit security level (i.e., setting $\tau = 80$, $\lambda = 256$) and choosing p to be a 256-bit prime, the proof length is only $2\lambda + \lceil \frac{2\tau \log p}{\log p - 1} \rceil = 695$ bits, almost a 2x improvement over the best pairing-based SNARGs.

As in Theorem 4.1.1, we rely on linear PCPs (again, using the linear PCPs of [BHI+24] and [BIOW20]). However, due to the reduced malleability of the encryption scheme, we do not have the additional overhead of compiling to strong LMIP. We still need to slightly adapt the PCPs, but this adaptation significantly more efficient. It is due to this that Theorem 4.1.2 achieves better concrete parameters than Theorem 4.1.1, at the cost of the added random oracle output. See Section 4.2.3 for more details.

4.1.2 Open Problems and Future Directions

In this work, we establish the practical feasibility of dv-SNARGs in the GGM whose proof size contains a single group element and a small number of additional bits that depends on the level of soundness. While this proof size is not too far from optimal, our results leave room for three kinds of improvement: (1) further improving succinctness, (2) improving prover and verifier runtimes, and (3) making the CRS fully reusable. We elaborate on each goal separately below.

Succinctness. There are two plausible approaches for further improving the proof size.

- *Tighter analysis.* In our analysis of packed ElGamal, we give a bound on the possible malleability attacks a malicious party may do. However, we believe that our analysis is quite loose, and conjecture that an even a slightly simpler construction can achieve a better level of soundness. See Section 4.2.3 for further discussion, as well as an explicit proposal for a dv-SNARG that we conjecture to achieve soundness $2^{-\tau}$ with proofs consisting of only 1 group element and $\approx 2\tau$ additional bits. For a soundness error of 2^{-80} at a 128-bit security level, this would amount to a proof size of ≈ 420 bits.

- *Better PCPs.* As with prior constructions [BCI+13, BIOW20, BHI+24], our dv-SNARGs rely on different flavors of linear PCPs. However, unlike these previous constructions, our apporach is less sensitive to the number of queries and depends mainly on the ratio μ between the *total bit-length* of the LPCP answers and the soundness level τ. The LPCPs we use have $\mu \approx 2$, which is why the SNARG described in Theorem 4.1.2 tends to 2τ, and explains the 2τ additive term in our proof length. As noted in [BHI+24], known hardness of approximation results for MAXLIN [Hås01, FJ12, ABCH19] imply 1-query LPCPs with $\mu \approx 1$. Using such a linear PCP would result in a proof where the additive term is improved from $\approx 2\tau$ to $\approx \tau$. However, these LPCPs have a non-negligible completeness error and seem practically infeasible. The completeness error can potentially be eliminated by allowing more queries, combining PCPs with optimal amortized query complexity [HK05] with the universal factor graph technique from [ABCH19] to make the query distribution input-independent. However, this approach too seems practically infeasible. We leave open the question of designing practical LPCPs with $\mu \approx 1$.

The above two potential improvements could lead to the following dv-SNARG.

Conjecture 4.1.3. Let $\mathbb{G} = \mathbb{G}_\lambda$ be a generic group and τ a soundness parameter. There exists a dv-SNARG for proving the satisfiability of a Boolean circuit of size s with the following features:

- Soundness error: $2^{-\tau+\log\log\lambda} + O(t^2/2^\lambda)$ against t-query adversaries;
- Proof size: 1 $\mathbb{G}$-element and $\tau + o(\tau)$ additional bits;
- CRS size: $O(\tau s)$ $\mathbb{G}$-elements.

For a soundness error of 2^{-80} at a 128-bit security level this would amount to a proof size of ≈ 340 bits, roughly half the proof size from Theorem 4.1.2.

Runtimes. Our new proof systems are practically feasible even for satisfiability problems involving thousands of constraints. However, the concrete runtime of the prover and (especially) the verifier still leave much to be desired, and improving these overheads is a major direction for future research.

Our packed ElGamal encryption is based on the distributed discrete logarithm algorithm from [BGI17], which helps us achieve perfect completeness. With a more careful analysis, one might be able to switch to the faster distributed discrete logarithm algorithm from [DKK18] to quadratically reduce verification time.

An orthogonal improvement is a tighter analysis of the SNARG verification time. Our analysis pessimistically assumes the magnitude of LPCP answers to scale linearly with the proof length. However, for natural linear PCPs, a quadratic improvement could be potentially obtained by using a concentration bound for a corresponding random walk. Similar ideas have been explored in [BIOW20]. In Claims 4.5.5 and 4.5.10, we show that this analysis is compatible with our transformations. See Conjecture 2.3.7 for a relevant conjecture.

Combining both of the above potential optimizations, the verifier's runtime can grow linearly with $s^{1/4}$ rather than linearly with s, potentially making our SNARGs practical for much larger circuits.

Finally, there is a lot of room for improving the concrete efficiency of the LPCPs we employ. In particularly, we rely on 1-query LPCPs from [BIOW20, BHI+24] that apply to Boolean constraints or arithmetic constraints over small fields. Extending them to natively accommodate arithmetic constraints over large fields remains open.

Reusability. In every dv-SNARG, the CRS setup can be safely reused an arbitrary number of times as long the prover does not learn (too many times) whether the verifier accepts badly formed proofs.[4] This may be good enough for many practical use cases, especially when there are long-term relations between the prover and the verifier. In particular, the verifier can replace the CRS after several rejections, or alternatively not reveal whether each individual proof is accepted.

However, the standard (strong) notion of reusability for SNARGs requires that the CRS can be safely reused even when a malicious prover can fully observe the verifier's accept/reject decisions. Our dv-SNARGs are not reusable in this sense, leaving the question of achieving full reusability open. We explain the source of the problem below.

The BCIOP compiler [BCI+13] shows that by combining an LPCP that has reusable soundness with a linear-only encryption scheme, one can obtain a dv-SNARG with (strong)

[4] In contrast, some *interactive* arguments in the preprocessing model, such as ones suggested in [IKO07, BHI+24], require an independent setup for each proof instance even when malicious provers cannot learn the verifier's decisions.

reusable soundness. However, the kinds of LPCPs and MIPs we use (concretely, "strong linear MIPs" and "modded linear PCPs") do not have reusable soundness. In the case of strong linear MIPs, where malicious provers can employ an arbitrary strategy, the lack of reusable soundness seems inherent. However, there is hope to construct reusably sound modded LPCPs. Indeed, [BHI+24, Corollary 5.24] realized reusably sound *bounded* 1-query LPCP over large fields, which is a strongly related notion.

4.1.3 Related Work

In this section, we give an overview of the concrete level of succinctness that can be achieved in each of the main generic models: the random oracle model (ROM), the generic bilinear group model, and the generic group model (GGM). For concreteness, we require here soundness error of 2^{-80} at a 128-bit security level for designated verifier SNARGs and 128-bit soundness for publicly verifiable ones. See Appendix B.1 for an extended discussion about the security notion and this choice of numbers for comparison.

Random Oracle Model. In the random oracle model, the most succinct hash-based SNARGs [BBHR18, ACFY24a, ACFY24b] have proofs with size roughly 40Kib (for instances of size 2^{12}). At a technical level, these (setup-free) SNARGs combine the blueprint of Kilian and Micali with an interactive variant of classical PCPs known as an IOP [BCS16, RRR16]. See [CY24] for further details. Using classical PCPs instead of IOPs, one could potentially obtain somewhat better succinctness at the expense of a much slower prover time. However, even in this case, proofs would have length in the thousands of bits.

Generic Bilinear Group Model. Bilinear group-based SNARGs can obtain a much better level of succinctness by incorporating a different relaxation of classical PCPs known as a *linear PCP* [IKO07]. The first practical SNARGs based on bilinear groups were given by Groth [Gro10]. Following a sequence of works [Lip13, GGPR13, BCI+13, DFGK14], the Groth16 SNARG [Gro16] was considered until recently to be the state of the art in succinctness. Built on asymmetric pairings, a Groth16 proof has size 2 $\mathbb{G}_1$-elements and 1 $\mathbb{G}_2$-element. For the popular group of choice, BLS12-381 (for 128-bit security), this corresponds to $2 \cdot 384 + 768 = 1536$ bits. Recently, [Lip24] improved on the size of Groth16, achieving a size of 3 $\mathbb{G}_1$-elements and 1 field element, which equates to $3 \cdot 384 + 256 = 1408$ bits. This was reduced in [DMS24] to 2 $\mathbb{G}_1$-elements and 2 field elements arriving at $2 \cdot 384 + 2 \cdot 256 = 1280$ bits. When instantiating [Mic94] with the linear map commitments of [LM19] and a 2-query linear Reed-Solomon PCP implicit in [DFGK14] (see [BHI+24, Corollary D.6]), one also gets a proof size of 2 $\mathbb{G}_1$-elements and 2 field elements. If one is willing to use the full PCP machinery instantiating [Mic94] with the subvector commitments of [LM19], one may get a slightly lower proof size. However, as discussed above, such general-purpose PCPs have poor concrete efficiency.

Generic Group Model. Most relevant to our work, another line of research [BCI+13, BCC+16, BBB+18, BIOW20, BHI+24] considers minimizing proof size using generic *pairing-free* groups, namely in the standard GGM. The simpler structure gives hope for more conservative group instantiations with better concrete parameters. Unlike pairing-based SNARGs, the most succinct GGM-based SNARGs apply only in the designated-verifier setting. Settling for inverse-polynomial soundness error, Barta et al. [BIOW20], obtained dv-SNARGs with proofs as short as 2 $\mathbb{G}$-elements, which corresponds to 512 bits for Curve 25519, a popular group of choice. However, applying this construction with our soundness target of 2^{-80}

would make verification practically infeasible, unless the proof size is increased drastically to amplify soundness.

4.1.4 Organization

The rest of this chapter is organized as follows. In Section 4.2, we give a high-level overview of the ideas and techniques used in our work. In Section 4.3, we define compressible encryption schemes, and introduce the packed ElGamal encryption scheme, along with a variant thereof. In Section 4.4, we define malleability security notions and prove that our compressible encryption schemes meet these security guarantees. In Section 4.5, we show how to transform linear PCPs to strong linear MIPs and to modded linear PCPs. In Section 4.6, we combine compressible encryption schemes and suitable linear PCPs (or MIPs) to construct dv-SNARGs.

4.2 Technical Overview

In this section, we give an overview of our results and the underlying techniques.

4.2.1 Designated-Verifier SNARGs Blueprint

We revisit a paradigm for constructing designated-verifier SNARGs by combining linearly homomorphic encryption schemes with linear PCPs and related objects (such as linear IPs) developed in [IKO07, BCI+13]. In a linear PCP (LPCP) over a field $\mathbb{F}_p$ the prover (whether honest or malicious) commits to a proof $\pi \in \mathbb{F}_p^\ell$, and the verifier chooses queries $\mathbf{a}_1, \ldots, \mathbf{a}_q \in \mathbb{F}_p^\ell$. The verifier then receives answers $(b_1, \ldots, b_q) \in \mathbb{F}_p$ to the queries, where $b_i = \langle \pi, \mathbf{a}_i \rangle \in \mathbb{F}_p$. To construct a designated-verifier SNARG from LPCPs, we have the verifier choose its queries first and put them into the common reference string. To have any chance of preserving soundness, we hide these queries from the prover by encrypting them, where only the verifier is given the decryption key. Intuitively, this way the prover has a hard time making its proof string depend on the queries. For completeness to still hold, we need this encryption to enable linear computation on the encrypted messages.

We describe in more detail the transformation given an LPCP and an encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ that is linearly homomorphic. For convenience of notation, for now, we consider only 1-query LPCPs.

- **Setup.** Generate keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$ for the encryption scheme, and PCP verifier query $\mathbf{a} \in \mathbb{F}_p^\ell$. Encrypt the queries, $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}, \mathbf{a}[i])$, where $\mathbf{a}[i]$ is the i-th entry of $\mathbf{a}$. The verifier private state is sk, and the public reference string contains the public key pk and ciphertexts $(\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.

- **Prover.** Given pk, and $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, the honest prover generates $\pi \in \mathbb{F}_p^\ell$ as in the linear PCP. It then homomorphically evaluates $\langle \pi, \mathbf{a} \rangle$ using the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, thus generating a new ciphertext ct'. The prover message is ct'.

- **Verifier.** Given the secret key sk, and ciphertext ct', decrypt

$$b = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}') \in \mathbb{F}_p$$

  and check that the PCP verifier accepts given b as the query answer.

Observe that in order to argue soundness, we need more properties from our encryption scheme. Indeed, if a malicious prover can do non-linear operations on the encrypted messages, then it can essentially launch a non-linear attack on the linear PCP, in which case we cannot rely on soundness of the PCP. Thus, we want the encryption scheme to be linearly homomorphic but the homomorphic operations to be restricted only to linear ones (or, more generally, affine ones). Encryption schemes with this property are referred to as being "linear-only" homomorphic.

Designated verifier SNARGs from ElGamal. Following the [BCI+13] paradigm, Barta et al. [BIOW20] construct dv-SNARGs by utilizing the ElGamal encryption scheme, which is linearly homomorphic for small messages. Recall that, given a suitable group $\mathbb{G}$ of order p' with generator g, the ElGamal encryption scheme is:

- KeyGen: Sample random $x \xleftarrow{\$} \mathbb{Z}_{p'}$. Set $\mathsf{pk} = g^x$ and $\mathsf{sk} = x$.
- Enc: Given public key $\mathsf{pk} = h$, to encrypt $m \in \mathbb{Z}_{p'}$ pick $r \xleftarrow{\$} \mathbb{Z}_{p'}$ and output $(g^r, h^r g^m)$.
- Dec: Given secret key $\mathsf{sk} = x$ and a ciphertext (c_1, c_2), compute the message $m = \mathsf{DLog}(c_2 \cdot c_1^{-x})$.

Decryption, here, only works if m is small (i.e., so that discrete log can be computed by polynomially bounded honest parties). [BIOW20] show that, in the generic group model (GGM), the ElGamal encryption scheme satisfies linear targeted malleability, a variant of linear-only encryption. They further design 1-query LPCPs over a field of size $\mathsf{poly}(\lambda)$ with soundness error $1/\mathsf{poly}(\lambda)$. By combining these two ingredients using the compiler described above, they construct a dv-SNARG in the GGM whose argument consists of 2 group elements and whose soundness error is $1/\mathsf{poly}(\lambda)$.

In the following, we explore how to push this idea to negligible soundness.

Adapting [BIOW20] for negligible soundness. The easiest way to achieve negligible soundness using the previous approach is to repeat the proof q times, where q is super-constant in λ. This, however, would increase size of the proof to $2q = \omega(1)$ group elements, which is too large.

Our first step to reduce the size is to reuse the ciphertext randomness of ElGamal with multiple secret keys in order to encrypt a vector of messages $m_1, \dots m_q$. In more detail:

- KeyGen: Sample random $x_1, \dots, x_q \xleftarrow{\$} \mathbb{Z}_{p'}$. Set the public key $\mathsf{pk} = (g^{x_1}, \dots, g^{x_q})$ and the secret key $\mathsf{sk} = (x_1, \dots, x_q)$.
- Enc: Given public key $\mathsf{pk} = (h_1, \dots, h_q)$, to encrypt $m_1, \dots, m_q \in \mathbb{Z}_{p'}$ pick $r \xleftarrow{\$} \mathbb{Z}_{p'}^n$ and output $(g^r, h_1^r g^{m_1}, \dots, h_q^r g^{m_q})$.
- Dec: Given secret key $\mathsf{sk} = (x_1, \dots, x_q)$ and a ciphertext $(c_0, \dots, c_q)$, output $(\mathsf{DLog}(c_1 \cdot c_0^{-x_1}), \dots, \mathsf{DLog}(c_q \cdot c_0^{-x_q}))$.

Observe that this change preserves linear homomorphism: given the two ciphertexts

$$(g^r, h_1^r g^{m_1}, \dots, h_n^r g^{m_q}) \text{ and } (g^{r'}, h_1^{r'} g^{m'_1}, \dots, h_q^{r'} g^{m'_q})$$

each encrypting a q-message vector we can compute a ciphertext

$$\begin{aligned}&(g^r g^{r'}, h_1^r g^{m_1} h_1^{r'} g^{m'_1}, \dots, h_t^r g^{m_q} h_n^{r'} g^{m'_q})\\ =&(g^{r+r'}, h_1^{r+r'} g^{m_1+m'_1}, \dots, h_q^{r+r'} g^{m_q+m'_q}),\end{aligned}$$

which decrypts to the sum of the message vectors. Thus, we can use it in the paradigm.

With this modification, we have already reduced our proof length from $2q$ to $q+1$ group elements, which is a significant decrease but still requires a super-constant number of group elements to achieve negligible soundness error. However, we have gained more power: the malicious prover is restricted to computing the same linear function over all elements of the vector. This allows us to move from a 1-query LPCP to a q-query one rather than repeat the 1-query LPCP q times. Multi-query LPCPs are significantly easier to design than their 1-query variant.[5] Revisiting our dv-SNARG construction, we use a q-query LPCP to generate q queries $\mathbf{a}_1, \ldots, \mathbf{a}_q$. The verifier then encrypts the queries $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i])$. As in the 1-query compiler described above, the common reference string contains all of these ciphertexts. The prover then homomorphically computes a ciphertext encrypting the value $\pi^{\mathsf{T}} \begin{pmatrix} \mathbf{a}_1 & \ldots & \mathbf{a}_q \end{pmatrix}$ and sends it to the verifier. The verifier decrypts this ciphertext and checks whether the LPCP verifier accepts given the decrypted values.

We have established that the paradigm can be made to work relatively efficiently for q-query linear PCPs. However, this does not suffice to achieve negligible soundness with a constant number of group elements, as we would need a linear PCP with constant query complexity and negligible soundness, which we only know how to construct over large fields, which is both incompatible with computing the discrete log, and would require a much larger generic group.

Smaller proofs using compressible encryption. We make two observations. The first is that because the messages need to be small to be able to compute the discrete logarithm, most of the group is unused. In other words, in an amortized sense, one group element of size $O(\lambda)$ only encodes $\mathrm{polylog}(\lambda)$ bits of information. Our second observation is that the homomorphic properties of the encryption scheme are used only once in the dv-SNARG. Indeed, after computing the query answers under the encryption, the prover simply sends the resultant ciphertexts to the verifier, who decrypts them immediately.

In order to utilize the above observations, we consider encryption schemes which are *compressible*. In a (linearly homomorphic) compressible encryption scheme, the encryption procedure Enc produces ciphertexts ct that are large and support homomorphism. The scheme additionally has a compression algorithm $\mathsf{Compress}$ takes a ciphertext ct and compresses it into a smaller ciphertext cct, which might lose the homomorphic capabilities held by ct.

We can now restate the transformation using the combined ideas of q-query LPCPs and compressible encryption schemes:

- **Setup.** Generate keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$ for the encryption scheme, and PCP verifier queries $\mathbf{a}_1, \ldots, \mathbf{a}_q$. Encrypt the queries,

$$\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]).$$

  The verifier private state is sk, and the public reference string contains the public key pk and ciphertexts $(\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.

- **Prover.**
  1. Given pk, and $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, the prover generates π as in the linear PCP. It then homomorphically evaluates $\langle \pi, \mathbf{a}_1 \rangle$ up to $\langle \pi, \mathbf{a}_q \rangle$ using the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, thus generating a new ciphertext ct'.

[5] See [BIOW20, BHI+24] for further discussion on the complications in designing 1-query LPCPs.

2. Compute a compressed ciphertext cct from ct' using $\mathsf{Compress}$. The prover message is cct.

- **Verifier.** Given the secret key sk, and compressed ciphertext cct, decrypt $(b_1, \ldots, b_q) = \mathsf{Dec}(\mathsf{sk}, \mathsf{cct})$ and check that the PCP verifier accepts given $b_1, \ldots, b_q$ as the query answers.

In the next sections, we explore instantiating this extension of the [BCI+13] paradigm.

4.2.2 Packed ElGamal

We consider the *packed* ElGamal scheme implied by techniques developed in [BGI17, DGI+19] and coined in [BBD+20]. The setup and encryption of the scheme are identical to multi-message ElGamal, but now we also consider a *compression* algorithm and subsequent decryption algorithm for compressed ciphertexts.

The main ingredient of the compression procedure is the "distributed discrete logarithm" (DDL) algorithm. The DDL algorithm allows two parties that have group elements h_1 and $h_2 = h_1 \cdot g^x$ (respectively), to convert these elements into integers y_1 and y_2 such that $y_1 = y_2 + x$, given that x is smaller than some fixed bound B. We describe a simple distributed discrete logarithm algorithm (more efficient algorithms exist but are unnecessary to understanding our results). Suppose the two parties are given access to a random function $\phi\colon \mathbb{G} \to \{0,1\}^\ell$. Now, each party computes its DDL share as follows: compute $h_i \cdot g^y$ for every $y < B'$, and output the smallest y such that $\phi(h_i g^y) = 0^\ell$. Now, if B' is much bigger than B with respect to x then with high probability both parties will arrive at the same element which maps to 0^ℓ, i.e., $h_1 g^{y_1} = h_2 g^{y_2} = h_1 g^{x+y_2}$, and so $y_1 = y_2 + x$.

We show how to use DDL to compress an ElGamal ciphertext that encrypts a message $m_1, \ldots, m_q \in \mathbb{Z}_p$:

- $\mathsf{Compress}$: Given a ciphertext $\mathsf{ct} = (c_0, c_1, \ldots, c_q)$, compute $v_i = \mathsf{DDL}(c_i) \mod p$ for all $i \in [q]$, and output $\mathsf{cct} = (c_0, v_1, \ldots, v_q)$.
- Dec: Given secret key $\mathsf{sk} = (x_1, \ldots, x_q)$ and a compressed ciphertext $(c, v_1, \ldots, v_q)$, compute

$$((\mathsf{DDL}(c^{x_1}) - v_1) \mod p, \ \ldots, \ (\mathsf{DDL}(c^{x_q}) - v_q) \mod p) \ .$$

Then, assuming the distributed discrete logarithm algorithm does not fail, computing the compression and then the decryption procedures becomes,

$$\mathsf{Dec}(\mathsf{Compress}(g^r, h_1^r g^{m_1}, ..., h_q^r g^{m_q})),$$

and for every $i \in [q]$ we get:

$$\begin{aligned} &\mathsf{DDL}(g^{x_i r}) \mod p - \mathsf{DDL}(g^{x_i r + m_i}) \mod p \\ =&(y_i - (y_i - m_i)) \mod p = m_i \mod p \end{aligned}$$

Failures of the DDL algorithm can be prevented by the compressing party resampling the randomness of the ciphertext. While the compressor cannot check whether a compression error has occurred since it does not the decryption key, it can test whether there is a possible value which leads to a failure (recall that we are considering here p which is small).

Is packed ElGamal linear only? Recall that in order to use the paradigm to construct dv-SNARGs we needed an encryption scheme that is linear-only (or linear targeted malleable), i.e., is capable of doing linear operations on the encrypted messages and nothing else. Since $\mathsf{Compress}$ is a postprocessing procedure to the standard ElGamal ciphertexts, one could naively expect that this encryption, too, is linear only. However, we show that this is, in fact, not the case by demonstrating that one can homomorphically evaluate non-linear functions by forcing a decryption error.

We demonstrate how to evaluate a non-linear function over a packed ElGamal encryption with a ciphertext that encrypts a message $m \in \{0,1,2\}$, and the compression happens modulo 3. The adversary gets a ciphertext $\mathsf{ct} = (c_0, c_1)$ and the public key (g, h). Further, it knows $h = g^x$, $c_0 = g^r$, and $c_1 = g^{rx+m}$ for some $r, x \in \mathbb{Z}_{p'}$ and $m \in \{0,1,2\}$. In order to attack the scheme, the adversary scales (c_0, c_1) by a large random number $s \in \mathbb{Z}_{p'}$. More specifically, it produces a new ciphertext $(c'_0 = c_0^s = g^{rs}, c'_1 = c_1^s = g^{rxs+sm})$. If the adversary chooses the (malformed) compressed ciphertext (c'_0, e) for $e \in \mathbb{Z}_3$, then the decryption procedure Dec will output $(\mathsf{DDL}(c_0'^x) - e) \mod 3$. Of course, the evaluator does not know $c_0'^x$, but it does know $c'_1 = c_0'^x / g^m$. Therefore, it knows that Dec will output the following:

- $(\mathsf{DDL}(c'_1) - e) \mod 3$ if $m = 0$;
- $(\mathsf{DDL}(c'_1/g^s) - e) \mod 3$ if $m = 1$;
- $(\mathsf{DDL}(c'_1/g^{2s}) - e) \mod 3$ if $m = 2$.

In other words, the adversary can homomorphically evaluate the function $f_{s,e}$ defined as:

$$m \mapsto \begin{cases} (\mathsf{DDL}(c'_1) - e) \mod 3 \text{ if } m = 0 \\ (\mathsf{DDL}(c'_1/g^s) - e) \mod 3 \text{ if } m = 1 \\ (\mathsf{DDL}(c'_1/g^{2s}) - e) \mod 3 \text{ if } m = 2 \end{cases}$$

Because c'_1, c'_1/g^s, and c'_1/g^{2s} are far apart, the value output of DDL given each as input is independent (as they choose a different zero point of ϕ), and random[6]. Therefore, for example, with a constant probability we will have the function $(\mathsf{DDL}(c'_1) - e) \mod 3 = (\mathsf{DDL}(c'_1/g^s) - e) \mod 3 = 0$ and $(\mathsf{DDL}(c'_1/g^{2s}) - e) \mod 3 = 1$, which is not a linear function over $\mathbb{Z}_3$.

Because s and e are chosen by the adversary, and it can compute $f_{s,e}$, it can also use rejection sampling until $f_{s,e}$ is whatever function it wants.

This attack generalizes to bigger message spaces and multiple ciphertexts. It additionally generalizes in the following sense to ciphertexts encrypting multiple messages: given a ciphertext encrypting a vector $(m_1, \ldots, m_n)$ the evaluator can evaluate non-linear functions $f_1, \ldots, f_n$ on the ciphertext, such it decrypts to $f_1(m_1), \ldots, f_n(m_n)$.

Isolated homomorphism of packed ElGamal. We prove in the generic group model that the above-mentioned attack is the most a malicious party can do. More specifically, we prove that for every adversary provided with ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$ encrypting vectors $\mathbf{m}_1, \ldots, \mathbf{m}_\ell \in \mathbb{F}_p^q$ that outputs a compressed ciphertext cct, there are functions $f_1, \ldots, f_q$ so that

$$\mathsf{Dec}(\mathsf{cct}) = (f_1(\mathbf{m}_1[1], \ldots, \mathbf{m}_1[\ell]), \ldots, f_q(\mathbf{m}_q[1], \ldots, \mathbf{m}_q[\ell])) \ .$$

In other words, the adversary can evaluate arbitrary functions, but it cannot share information between different "slots" of the vectors. We call this property of the encryption scheme *isolated homomorphism*.

[6]It is not uniformly random but the distribution has high enough entropy to make this attack work.

Strong linear MIPs. Isolated homomorphism of the packed ElGamal encryption scheme allows a malicious prover to apply an arbitrary function f_i to the veifier's i-th query $\mathbf{a}_i$. However, since these functions are isolated, they cannot "share" information about between different slots of the ciphertexts. Thus, isolated homomorphism translates to the soundness of a multi-prover interactive proof (MIP) rather than of a PCP. In an MIP, a set of provers $P_1, \ldots, P_q$ wants to convince a single verifier of a statement. The verifier sends a query $\mathbf{a}_i$ to each prover P_i and receives a response b_i. A malicious set of provers can answer with $(f_1(\mathbf{a}_1), \ldots, f_q(\mathbf{a}_q))$ for any arbitrary functions $f_1, \ldots, f_q$. This exactly matches the guarantees provided by the isolated homomorphism notion.

In fact, we need the additional property that the honest proof to be a single linear strategy. We call an MIP where the honest prover computes a single linear function, but the malicious provers are allowed to compute arbitrary (isolated) functions a *strong linear MIP*[7]. Strong linear MIPs have been constructed (e.g., in [IKO07]) by starting with a linear PCP with soundness against provers that (1) only apply linear strategies and (2) apply the same strategy to each query. These requirements are relaxed by (1) adding a linearity test, thereby removing the linearity assumption, and (2) adding a consistency check between the queries, thereby removing the single strategy requirement.

This construction suffices to get strong linear MIPs with constant soundness, which can then be boosted by repeating the protocol. Unfortunately, the concrete parameters achieved by this process leave much to be desired. We give an alternate transformation that is specific to 2-query linera PCPs which combines the linearity and consistency checks into one combined check inspired by the linear-consistent test of [AHRS01], thus improving the constants derived by this transformation. See Section 4.5.1 for further technical details.

Dv-SNARGs from packed ElGamal. We combine the packed ElGamal encryption scheme with strong linear MIPs to construct dv-SNARGs. Recall that we had compressed ciphertexts of the form $\mathsf{cct} = (c_0, v_1, \ldots, v_q)$, such that c_0 is a group element, and $v_i \in \mathbb{Z}_p$, where $\mathbb{Z}_p$ is the message space. Thus, the dv-SNARG has proof length equal to one compressed ciphertext of a message of length equal to the size of the query answers in the MIP.

Thus, when instantiated with a linear MIP (small) field $\mathbb{F}_p$, with $O(\tau)$ queries and soundness error $2^{-\tau}$ (which can be constructed from ones with constant soundness via τ-wise repetition), we get a dv-SNARG whose proof length is a single group element along with $O(\tau \cdot \log p)$ bits (Theorem 4.1.1). Figure 4.1 provides a summary of our transformations from linear PCP to dv-SNARG.

4.2.3 Improved Proof Length by Reducing Malleability

The main issues described in the previous section are caused due to the malicious prover having the ability to make DDL fail without being detected. We show that this behavior of the malicious verifier can be limited at the cost of appending a hash of the points that DDL synchronizes to (i.e., the locations where ϕ is zero which the DDL algorithm outputs). Let H be a hash function, modeled by a random oracle. We change the compression and decryption schemes in the following way:

- $\mathsf{Compress}(\mathsf{ct} = (c_0, c_1, \ldots, c_q))$:
  1. Let $v_i \leftarrow \mathsf{DDL}(c_i)$ for $i \in [q]$.

[7] In [IKO07] this notion is called linear MIP. In more recent work [BISW18], linear MIP refers to a notion in which the malicious prover also has to behave linearly.

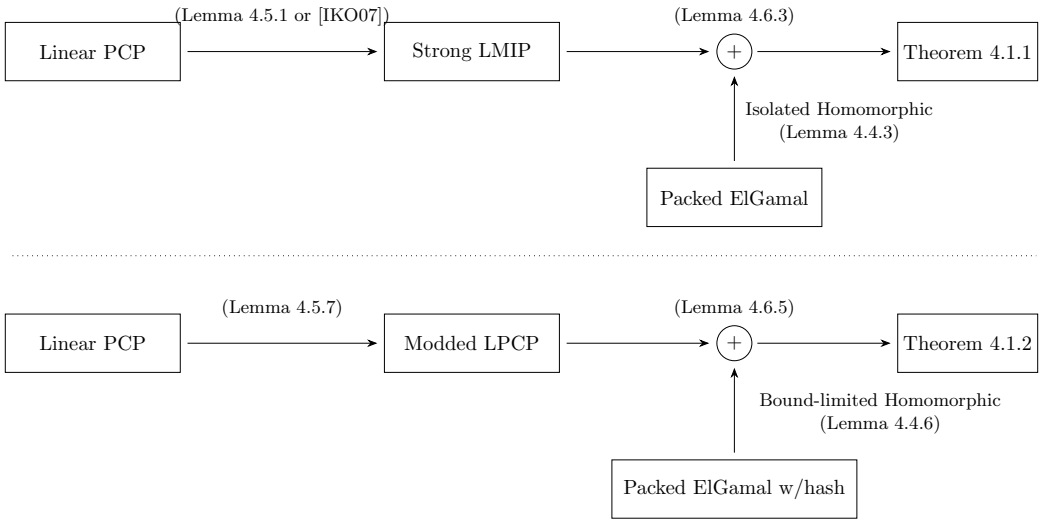

Figure 4.1: Summary of our transformations.

2. Let $k \leftarrow \mathsf{H}(c_1 g^{v_1}, \ldots, c_n g^{v_q})$.
3. Output $(c_0, v_1 \mod p, \ldots, v_q \mod p, k)$.

- $\mathsf{Dec}(\mathsf{sk} = (x_1, \ldots, x_q), \mathsf{cct} = (c, e_1, \ldots, e_q, k))$:
  1. Let $v'_i \leftarrow \mathsf{DDL}(c^{x_i})$ for $i \in [q]$.
  2. If $\mathsf{H}(c^{x_1} g^{v'_1}, \ldots, c^{x_n} g^{v'_q}) \neq k$ output $\perp$.
  3. Otherwise, output $((v'_1 - e_1) \mod p, \ldots, (v'_n - e_q) \mod p)$

Observe that compressed ciphertexts have size 1 group element, one hash output, and $q \log p$ bits, as opposed to 1 group element and $q \log p$ bits, which seems worse than our previous scheme.

However, we show that the protocol above is bound-limited homomorphic, a notion which enables the prover only slight non-linear power (see Section 4.4.1 for a formal definition). Importantly, a bound-limited homomorphism is significantly more restrictive than isolated homomorphism. Thus, we do not have to divert to strong linear PCPs, and can model these attacks as a slightly modified version of linear PCPs (which we call modded linear PCPs). We show that standard linear PCPs can be adapted to this modified model with small loss in the query complexity. Known linear PCPs can have significantly better parameters than strong linear MIPs, translating to a smaller number of queries q.

This allows us to construct a dv-SNARG with proof size 1 group element, one hash output, and bits approaching 2τ (Theorem 4.1.2). This is a significant improvement over our previous dv-SNARG when in the high-soundness regime (e.g., when we think of $\tau = 80$ and the size of a group element and a hash being 256 bits each). See Figure 4.1 for an overview of these transformations.

"Fishing in the dark." We believe that our approach can be pushed towards an even smaller SNARG. As previously demonstrated, the packed ElGamal scheme can be used to homomorphically evaluate non-linear functions. However, the class of non-linear functions described in our attack is quite limited. Recall that in the evaluation, the adversary forces a synchronization error of DDL to get non-linear behavior, meaning that $c_i g^{\mathsf{DDL}(c_i)}$ (which

is run on the adversary side) is different from $c_0^x g^{\mathsf{DDL}(c_0^x)}$ (which the decryptor computes). In the hash-verified approach, we added a hash to stop such behavior.

We conjecture that it is possible to "integrate" the hash check into the ciphertext, thus making it hard for an adversary to maul the ciphertexts even without the additional cost of the hash output:

- $\mathsf{Compress}(\mathsf{ct} = (c_0, c_1, \ldots, c_q))$:
  1. Let $v_i \leftarrow \mathsf{DDL}(c_i)$ for $i \in [q]$.
  2. Let $k_1, \ldots, k_q \leftarrow \mathsf{H}(c_1 g^{v_1}, \ldots, c_q g^{v_q})$.
  3. Output $(c_0, v_1 + k_1 \mod p, \ldots, v_q + k_q \mod p)$.
- $\mathsf{Dec}(\mathsf{sk} = (x_1, \ldots, x_q), \mathsf{cct} = (c, e_1, \ldots, e_q))$:
  1. Let $v_i' \leftarrow \mathsf{DDL}(c^{x_i})$ for $i \in [q]$.
  2. Let $k_1, \ldots, k_n \leftarrow \mathsf{H}(c^{x_1} g^{v_1'}, \ldots, c^{x_n} g^{v_q'})$.
  3. Otherwise, output $((v_1' - e_1 - k_1) \mod p, \ldots, (v_q' - e_n - k_q) \mod p)$

While we are currently unable to prove this, intuitively, this scheme should still be secure when combined with a linear PCP over $\mathbb{Z}_p$. To see why, suppose that the decryptor's outputs of $c^{x_i} g^{\mathsf{DDL}(c^{x_i})}$ have high entropy from the perspective of the adversary. In this case, it cannot query $\mathsf{H}(c^{x_1} g^{v_1'}, \ldots, c^{x_n} g^{v_n'})$, and the decryption output will seem truly random. Thus, this attack reduces to a *random* attack function, i.e., a random attack on the underlying PCP (to which all natural linear PCPs are secure).

If, however, the values $c^{x_i} g^{\mathsf{DDL}(c^{x_i})}$ have relatively low entropy from the perspective of the adversary, then each possible input to the hash function H defines an affine function over $\mathbb{Z}_p$. Our linear PCP will then provide soundness against each of the affine functions individually. However, we cannot afford a union bound over all such affine functions. We believe that our scheme is secure since the prover does not have full control of these functions, as they are partially defined using the hash function.

Thus, in this approach, in either case, the prover must "fishing in the dark" for a random function with which to attack the scheme. If we are correct, then the resultant dv-SNARG could have length that approaches 1 group element and 2τ bits. This improvement would be the first step towards proving Conjecture 4.1.3. We leave further analysis of this encryption scheme when used to compile a SNARG for exciting future work.

4.3 Compressible Encryptions Schemes

Our constructions of designated-verifier SNARGs will utilize linearly homomorphic encryption schemes that have compressible ciphertexts. In this section, we define compressible encryption schemes. In subsequent sections we give constructions of such schemes: in Section 4.3.1 we describe the packed ElGamal encryption scheme, and in Section 4.3.2 we show a variant on this scheme that additionally uses a hash function.

Definition 4.3.1 (Compressible linearly homomorphic encryption)**.** A compressible bounded linearly homomorphic encryption scheme in the generic group model with bounded message space and compressed ciphertext size σ_{cct} is a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Compress})$ that must satisfy the following properties:

- *Syntax.* We describe an encryption scheme with homomorphism modulus p', $n \in \mathbb{N}$ slots, plaintext moduli $p_1, \ldots, p_n$, decryption bound B, ciphertext size σ_{ct}, compressed ciphertext size σ_{cct}, encryption, decryption, and eval running times $t_{\mathsf{enc}}, t_{\mathsf{dec}}, t_{\mathsf{eval}}$. All algorithms have access to a GGM oracle of size λ.

- KeyGen: Outputs a public key pk and a secret key sk.
- $\mathsf{Enc}(\mathsf{pk}, m)$: On input a public key pk and a message $m \in \mathbb{Z}_{p'}^n$, outputs a ciphertext ct. This is done in time $t_{\mathsf{enc}}(\lambda)$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{cct})$: On input a secret key sk and a compressed ciphertext cct, output a message $m \in \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_n} \cup \{\perp\}$. This is done in time $t_{\mathsf{dec}}(\lambda)$.
- $\mathsf{Eval}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \pi)$: On input a public key pk, ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$ and a linear function $\pi \in \mathbb{Z}_{p'}^n$, outputs a new ciphertext ct'. This is done in time $t_{\mathsf{eval}}(\lambda, \ell)$.
- $\mathsf{Compress}(\mathsf{pk}, \mathsf{ct})$: On input a public key pk and a ciphertext ct, outputs a compressed ciphertext cct with $|\mathsf{cct}| = \sigma_{\mathsf{cct}}$.

- *Correctness.* The encryption has correctness error $\varepsilon_{\mathsf{cor}}$ if for every $\lambda, \in \mathbb{N}$, $\mathbf{m}_1, ..., \mathbf{m}_t \in \mathbb{Z}_{p'}^n$, and $\pi \in \mathbb{Z}_{p'}^\ell$

$$\Pr\left[\begin{array}{c|r} \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ \left(\mathbb{Z}_{p_i}\left(\langle \pi, \begin{pmatrix} \mathbf{m}_1[i] \\ \vdots \\ \mathbf{m}_t[i] \end{pmatrix} \rangle\right)\right)_{i \in [n]} & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ & \forall i \in [t],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{m}_i) \\ & \mathsf{ct}' \leftarrow \mathsf{Eval}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \pi) \\ & \mathsf{cct} \leftarrow \mathsf{Compress}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}') \\ & \forall i \in [n], \langle \pi, \begin{pmatrix} \mathbf{m}_1[i] \\ \vdots \\ \mathbf{m}_t[i] \end{pmatrix} \rangle \in [-B, B] \end{array}\right]$$
$$\geq 1 - \varepsilon_{\mathsf{cor}}(\lambda) \ .$$

We say that it is correct if $\varepsilon_{\mathsf{cor}}(\lambda) = \mathsf{negl}(\lambda)$.

- *Semantic security.* The encryption scheme has semantic security advantage $\varepsilon_{\mathsf{sem}}$ for t-query adversaries if for any $\lambda \in \mathbb{N}$ and adversary $\mathcal{A}$ that makes at most t queries to the GGM oracle:

$$\Pr\left[\begin{array}{c|r} \forall i,\ m_{0,i}, m_{1,i} \in \mathbb{Z}_q^n & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ \wedge\ b = b' & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ & ((m_{j,1}, \ldots, m_{j,\ell})_{j \in \{0,1\}}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{G}}(\mathsf{pk}) \\ & b \leftarrow \{0,1\} \\ & \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, m_{b,i}) \\ & b' \leftarrow \mathcal{A}^{\mathcal{G}}(\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{st}_{\mathcal{A}}) \end{array}\right]$$
$$\leq \frac{1}{2} + \varepsilon_{\mathsf{sem}}(\lambda, t, \ell) \ .$$

We say that it is semantically secure if for $t, \ell = \mathsf{poly}(\lambda)$ we have $\varepsilon_{\mathsf{sem}}(\lambda, t, \ell) = \mathsf{negl}(\lambda)$.

Remark 4.3.2. Construction 4.3.9 has an additional random oracle with output length λ. It is straightforward to add sampling of the random oracle into the notation above.

4.3.1 Packed ElGamal

We describe the packed ElGamal compressible encryption scheme:

Theorem 4.3.3. *The packed ElGamal encryption scheme described in Construction 4.3.4 is compressible bounded linearly homomorphic with the following properties:*

- *Homomorphism modulus:* p'*, the size of the generic group,*
- *Number of Slots:* n*,*
- *Correctness error:* 0*,*
- *Semantic security advantage:* $\varepsilon_{\mathsf{sem}}(\lambda, t, \ell) = 4t^2/2^\lambda$*.*
- *Plaintext moduli:* $p = p_1 = ... = p_n$*,*
- *Ciphertext size:* $n + 1$ $\mathbb{G}$*-elements,*
- *Compressed ciphertext size:* 1 $\mathbb{G}$*-element and* $\lceil n \log p \rceil$ *bits,*
- *Encryption time:* $(4n + 2) \log \lambda$ *group operations,*
- *Evaluation time for* ℓ *linear combination:* $\ell(2 \log \lambda + 1)(n + 1)$ *group operations,*
- *Decryption time:* $n(8Bn + \log \lambda)$ *group operations,*
- *Expected compression time:* $32Bn^2 + 2(n + 1) \log \lambda$ *group operations.*

Construction 4.3.4 (Packed ElGamal)**.** We specify the encryption scheme, parameterized by a message-space $p, B, n \in \mathbb{N}$ and number of supported additions ℓ. Let $\delta = 1/2n$ and number of GGM queries per DDL $T = 8Bn$.

$\mathsf{KeyGen}^{\mathcal{G}}$:

1. Let p' be the order of the generic group $\mathcal{G}$ and g its generator.
2. For every $i \in [n]$, sample $x_i \xleftarrow{\$} \mathbb{Z}_{p'}$.
3. Output $\mathsf{pk} = (g^{x_1}, \ldots, g^{x_n})$ and $\mathsf{sk} = (x_1, \ldots, x_n)$.

$\mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, m)$:

1. Parse $\mathsf{pk} = (h_1, \ldots, h_n)$ and $m \in \mathbb{Z}_{p'}$.
2. Sample $r \xleftarrow{\$} \mathbb{Z}_{p'}$.
3. Output $\mathsf{ct} = (g^r, h_1^r \cdot g^{m_1}, \ldots, h_n^r \cdot g^{m_n})$.

$\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$:

1. Parse $\mathsf{sk} = (x_1, \ldots, x_n)$ and $\mathsf{cct} = (c, e_1, \ldots, e_n)$.
2. For every $i \in [n]$, let $m_i = (\mathsf{DDL}_{B,\delta}(c^{x_i}) - e_i) \mod p$.
3. Output $m = (m_1, \ldots, m_n)$.

$\mathsf{Eval}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_t, \pi)$:

1. Parse $\mathsf{pk} = (h_1, \ldots, h_n)$ and $\mathsf{ct}_i = (a_i, \mathbf{b}_i)$.
2. Let $a' = a_1^{\pi_1} \cdot \ldots \cdot a_\ell^{\pi_\ell}$ and $b'_j = \mathbf{b}_1[j]^{\pi_1} \cdot \ldots \cdot \mathbf{b}_\ell[j]^{\pi_\ell}$ for $j \in [n]$.
3. Output $\mathsf{ct}' = (a', b'_1, \ldots, b'_n)$.

$\mathsf{Compress}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct})$:

1. Parse $\mathsf{pk} = (h_1, \ldots, h_n)$ and $\mathsf{ct} = (a, b_1, \ldots, b_n)$.
2. Do in a loop:
   - Sample $r \xleftarrow{\$} \mathbb{Z}_{p'}$ uniformly at random.
   - Compute $a \leftarrow a \cdot g^r$, and $b_i \leftarrow b_i \cdot h_i^r$ for every $i \in [n]$.
3. Until for every $i \in [n]$ and $j \in [-B, B]$ it holds that
$$\mathsf{DDL}^{\mathcal{G}}_{B,\delta}(b_i \cdot g^j) + j = \mathsf{DDL}^{\mathcal{G}}_{B,\delta}(b_i).$$
4. Output $\mathsf{cct} = (a, \mathsf{DDL}^{\mathcal{G}}_{B,\delta}(b_1) \mod p, \ldots, \mathsf{DDL}^{\mathcal{G}}_{B,\delta}(b_n) \mod p)$.

Lemma 4.3.5. *Construction 4.3.4 satisfies correctness and the running times are as described in Theorem 4.3.3.*

Proof. Fix parameters $\lambda, p, B, n, t \in \mathbb{N}$, messages $m_1, \ldots, m_t \in \mathbb{Z}_p^n$, and $\pi \in \mathbb{Z}_p^t$. We follow the correctness experiment, keeping track of what each value is:

- *Setup.* Fix any $(\mathsf{pk}, \mathsf{sk})$ sampled by KeyGen. Then letting $\mathsf{sk} = (x_1, \ldots, x_n)$, we have $\mathsf{pk} = (g^{x_1}, \ldots, g^{x_n})$.

- *Encryption.* For every $i \in [\ell]$, and any randomness r_i sampled during the encryption of ct_i, and $\mathbf{m}_i$, it holds that

$$\begin{aligned}\mathsf{ct}_i &= (g^{r_i}, h_1^{r_i} \cdot g^{\mathbf{m}_i[1]}, \ldots, h_n^{r_i} \cdot g^{\mathbf{m}_i[n]}) \\ &= (g^{r_i}, g^{x_1 \cdot r_i + \mathbf{m}_i[1]}, \ldots, g^{x_n \cdot r_i + \mathbf{m}_i[n]}) = (a_i, \mathbf{b}_i) \ .\end{aligned}$$

- *Linear evaluation.* Following the evaluation step, we have a ciphertext $\mathsf{ct}' = (a', b'_1, \ldots, b'_n)$ where

$$a' = \prod_{i \in [\ell]} a_i^{\pi_i} = g^{\sum_{i \in [\ell]} \pi_i \cdot r_i} \ ,$$

  and for every $k \in [n]$,

$$b'_k = \prod_{i \in [\ell]} \mathbf{b}_i[k]^{\pi_i} = g^{\sum_{i \in [\ell]} \pi_i \cdot r_i \cdot x_k + \pi_i \cdot \mathbf{m}_i[k]} \ .$$

- *Compression.* The compressed ciphertext is $\mathsf{cct} = (c, v_1, \ldots, v_n)$, where for some $r \in \mathbb{Z}_{p'}$ we have

$$c = a' \cdot g^r = g^{r + \sum_{i \in [\ell]} \pi_i \cdot r_i} \ ,$$

  and for every $k \in [n]$,

$$v_k = \mathsf{DDL}(b'_k \cdot h_k^r) = \mathsf{DDL}\left(g^{r \cdot x_k + \sum_{i \in \ell} \pi_i \cdot r_i \cdot x_k + \pi_i \cdot \mathbf{m}_i[k]}\right)$$

  and we know that for all $j \in [-B, B]$ we have

$$\begin{aligned}&\mathsf{DDL}\left(g^{r \cdot x_k + \sum_{i \in \ell} \pi_i \cdot r_i \cdot x_k + \pi_i \cdot \mathbf{m}_i[k]}\right) + j \\ =&\mathsf{DDL}\left(g^{r \cdot x_k + \sum_{i \in [\ell]} \pi_i \cdot r_i \cdot x_k + \pi_i \cdot \mathbf{m}_i[k] - j}\right) \\ =&\mathsf{DDL}\left(g^{r \cdot x_k + \sum_{i \in [\ell]} \pi_i \cdot r_i \cdot x_k + \sum_{i \in [\ell]} \pi_i \cdot \mathbf{m}_i[k] - j}\right)\end{aligned}$$

- *Decryption.* For every $k \in [n]$ the decryptor computes

$$\begin{aligned}&(\mathsf{DDL}(c^{x_k}) - v_k) \mod p \\ =&\Big(\mathsf{DDL}\left(g^{(r + \sum_{i \in [\ell]} \pi_i \cdot r_i) \cdot x_k}\right) \\ &- \mathsf{DDL}\left(g^{r \cdot x_k + \sum_{i \in \ell} \pi_i \cdot r_i \cdot x_k + \sum_{i \in [\ell]} \pi_i \cdot \mathbf{m}_i[k]}\right)\Big) \mod p\end{aligned}$$

Because $\sum_{i\in[\ell]} \pi_i \cdot \mathbf{m}_i[k] \in [-B, B]$ we get

$$\begin{aligned}
&\Big(\mathsf{DDL}\left(g^{(r+\sum_{i\in[\ell]}\pi_i\cdot r_i)\cdot x_k}\right)\\
&-\mathsf{DDL}\left(g^{r\cdot x_k+\sum_{i\in\ell}\pi_i\cdot r_i\cdot x_k+\sum_{i\in[\ell]}\pi_i\cdot\mathbf{m}_i[k]}\right)\Big) \mod p\\
=&\Big(\mathsf{DDL}\left(g^{r\cdot x_k+\sum_{i\in\ell}\pi_i\cdot r_i\cdot x_k+\sum_{i\in[\ell]}\pi_i\cdot\mathbf{m}_i[k]}\right)+\sum_{i\in[\ell]}\pi_i\cdot\mathbf{m}_i[k]\\
&-\mathsf{DDL}\left(g^{r\cdot x_k+\sum_{i\in\ell}\pi_i\cdot r_i\cdot x_k+\sum_{i\in[\ell]}\pi_i\cdot\mathbf{m}_i[k]}\right)\Big) \mod p\\
=&\left(\sum_{i\in[\ell]}\pi_i\cdot\mathbf{m}_i[k]\right) \mod p\ ,
\end{aligned}$$

which is the correct value.

What is left is to analyze the running times of the algorithms. For everything except Compress this is trivial. By Lemma 2.6.6 we get that for every $i \in [n]$

$$\Pr\left[\forall j \in [-B, B] : \mathsf{DDL}_{B,\delta}(b_i \cdot g^{-j}) = \mathsf{DDL}_{B,\delta}(b_i) - j\right] \geq 1 - \delta$$

Therefore, by union bound it follows that

$$\Pr\left[\forall j \in [-B, B], i \in [n] : \mathsf{DDL}_{B,\delta}(b_i \cdot g^{-j}) = \mathsf{DDL}_{B,\delta}(b_i) - j\right] \geq 1 - n\delta = 1/2$$

Therefore, Compress runs the loop a constant number of times in expectation.

□

Remark 4.3.6. As described above the Compress runs in polynomial time with overwhelming probability. To turn it into strict poly time one can limit the number of times the loop is run. Further, to drastically speed up compression one can leave out the loop entirely. Both of these changes result in imperfect correctness.

Lemma 4.3.7. *The encryption scheme described in Construction 4.3.4 satisfies statistical semantic security for multiple ciphertexts in the generic group model against adversaries with t queries with a statistical distance of $4t^2/p' \leq 4t^2/2^\lambda$.*

Proof. Follows from arguments almost identical to Claim 4.4.4. □

4.3.2 Packed ElGamal with Hash Check

We describe the packed ElGamal compressible encryption scheme extended with a hash:

Theorem 4.3.8. *The packed ElGamal with hash check encryption scheme described in Construction 4.3.9 is compressible bounded linearly homomorphic with the following properties:*

- *Homomorphism modulus: p', the size of the generic group,*
- *Number of Slots: n,*
- *Correctness error: 0,*
- *Semantic security advantage: $\varepsilon_{\mathsf{sem}}(\lambda, t, \chi, \ell) = 4t^2/2^\lambda$ for big enough t.*
- *Plaintext moduli: $p_1, ..., p_n$,*
- *Ciphertext size: $n + 1$ $\mathbb{G}$-elements,*
- *Compressed ciphertext size: 1 $\mathbb{G}$-element, 1 $\mathbb{H}$ hash and $\lceil \sum_{i\in[n]} \log p_i \rceil$ bits,*

- *Encryption time:* $(4n+2)\log\lambda$ *group operations,*
- *Evaluation time for* ℓ *linear combination:* $\ell(2\log\lambda+1)(n+1)$ *group operations,*
- *Decryption time:* $n(8Bn+\log\lambda)$ *group operations and* 1 $\mathbb{H}$ *hash operation,*
- *Expected compression time:* $32Bn^2+2(n+1)\log\lambda$ *group operations and* 1 $\mathbb{H}$ *hash operation.*

Construction 4.3.9 (Packed ElGamal with hash check)**.** We specify the encryption scheme, parameterized by $n, p_1, \ldots, p_n, B \in \mathbb{N}$ and number of supported additions ℓ. Let $\delta = 1/2n$, number of GGM queries per DDL $T = 8Bn$, and H be a random oracle with output size χ.

$\mathsf{KeyGen}^{\mathcal{G}}$:

1. Let p' be the order of the generic group $\mathcal{G}$ and g its generator.
2. For every $i \in [n]$, sample $x_i \xleftarrow{\$} \mathbb{Z}_{p'}$.
3. Output $\mathsf{pk} = (g^{x_1}, \ldots, g^{x_n})$ and $\mathsf{sk} = (x_1, \ldots, x_n)$.

$\mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, m)$:

1. Parse $\mathsf{pk} = (h_1, \ldots, h_n)$ and $\mathbf{m} \in \mathbb{Z}_{p'}^n$.
2. Sample $r \xleftarrow{\$} \mathbb{Z}_{p'}$.
3. Output $\mathsf{ct} = (g^r, h_1^r \cdot g^{\mathbf{m}[1]}, \ldots, h_n^r \cdot g^{\mathbf{m}[n]})$.

$\mathsf{Eval}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \pi)$:

1. Parse $\mathsf{pk} = (h_1, \ldots, h_n)$ and $\mathsf{ct}_i = (a_i, \mathbf{b}_i)$.
2. Let $a' = a_1^{\pi_1} \cdot \ldots \cdot a_\ell^{\pi_\ell}$ and $b'_j = \mathbf{b}_1^{\pi_1}[j] \cdot \ldots \cdot \mathbf{b}_\ell^{\pi_\ell}[j]$ for $j \in [n]$.
3. Output $\mathsf{ct}' = (a', b'_1, \ldots, b'_n)$.

$\mathsf{Compress}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct})$:

1. Parse $\mathsf{pk} = (h_1, \ldots, h_n)$ and $\mathsf{ct} = (a, b_1, \ldots, b_n)$.
2. Do in a loop:
   - Sample $r \xleftarrow{\$} \mathbb{Z}_{p'}$ uniformly at random.
   - Compute $a \leftarrow a \cdot g^r$, and $b_i \leftarrow b_i \cdot h_i^r$ for every $i \in [n]$.
3. Until for every $i \in [n]$ and $j \in [-B, B]$ it holds that
$$\mathsf{DDL}^{\mathcal{G}}_{B,\delta}(b_i \cdot g^j) + j = \mathsf{DDL}^{\mathcal{G}}_{B,\delta}(b_i).$$
4. For $i \in [n]$ let $e_i \leftarrow \mathsf{DDL}^{\mathcal{G}}_{B,\delta}(b_i)$
5. Let $k \leftarrow \mathsf{H}(a, b_1 \cdot g^{e_1}, \ldots, b_n \cdot g^{e_n})$.
6. Output $\mathsf{cct} = (a, e_1 \mod p_1, \ldots, e_n \mod p_n, k)$.

$\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$:

1. Parse $\mathsf{sk} = (x_1, \ldots, x_n)$ and $\mathsf{cct} = (c, e_1, \ldots, e_n, k)$.
2. For every $i \in [n]$, let $d_i = \mathsf{DDL}^{\mathcal{G}}_{B,\delta}(c^{x_i})$.
3. If $k \neq \mathsf{H}(c, c^{x_1} \cdot g^{d_1}, \ldots, c^{x_n} \cdot g^{d_n})$ output $\perp$.
4. For every $i \in [n]$, let $m_i = (d_i - e_i) \mod p_i$.
5. Output $m = (m_1, \ldots, m_n)$.

Lemma 4.3.10. *The encryption scheme described in Construction 4.3.9 is correct and statistically semantically secure for multiple ciphertexts in the generic group model with a loss of* $4t^2/p' \leq 4t^2/2^\lambda$*. Moreover, the running times are as described in Theorem 4.3.8.*

Proof. This follows from the exact same arguments as in Lemmas 4.3.5 and 4.3.7. □

4.4 Targeted Malleability

In this section we define malleability notions, and prove that our encryption schemes satisfy these notions in the generic group model. In Section 4.4.1, we define the two notions that we consider: isolated homomorphism, and bound-limited homomorphism. Then, in Section 4.4.2, we show that the packed ElGamal encryption scheme satisfies isolated homomorphism, and in Section 4.4.3 we show that packed ElGamal with hash satisfies bound-limited homomorphism.

4.4.1 Malleability Notions

We define two malleability notions for compressible encryption schemes. The first says that no adversary can mix information between different slots of the messages:

Definition 4.4.1 (Isolated homomorphism)**.** An n-slot compressible encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Compress})$ is *isolated linearly homomorphic* with distinguishing error $\varepsilon_{\mathsf{ih}} = \varepsilon(\lambda, t, m)$ in the generic group model if there exists a poly time simulator $\mathcal{S}$ such that for every plaintext generator $\mathcal{T}$ and oracle machine $\mathcal{A}$ that makes t queries to the GGM oracle, the following distributions have statistical distance at most $\varepsilon_{\mathsf{ih}}$:

- **Real world:**
  1. Sample $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$.
  2. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$.
  3. $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$.
  4. $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{a}_i)$ for all $i \in [m]$.
  5. $(\mathsf{cct}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m)$.
  6. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \perp$ output $\perp$
  7. $(a'_1, \ldots, a'_n) \leftarrow \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$
  8. Output $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \ldots, a'_n)$.
- **Ideal world:**
  1. Sample $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$.
  2. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$.
  3. $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$.
  4. $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ for all $i \in [m]$.
  5. $(\mathsf{cct}, \mathsf{st}_{\mathcal{A}}) \leftarrow_{\mathsf{tr}} \mathcal{A}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m)$, where tr is the trace of queries $\mathcal{A}$ made to the generic group oracle and the oracle's responses.
  6. $(f_1, \ldots, f_n) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m, \mathsf{cct})$.
  7. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \perp$ output $\perp$.
  8. $a'_i = f_i(\mathbf{a}_1[i], \ldots, \mathbf{a}_m[i])$ for all $i \in [n]$.
  9. Output $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \ldots, a'_n)$.

We say that the encryption scheme is isolated homomorphic if $t = \mathsf{poly}(\lambda)$ we have $\varepsilon_{\mathsf{ih}}(\lambda, m, t) = \mathsf{negl}(\lambda)$.

The second notion is a notion of bound-limited linear-only encryption, which says that any adversary can only apply linear functions to an encrypted message. These linear functions may be over a large field $\mathbb{F}_{p'}$, but their results need to be within a bounded range. If they are within that range the decryptor learns the values but modded by some smaller modulus p. This will later match our definition of modded linear PCPs (see Section 4.5.2).

Definition 4.4.2 (Bound-limited homomorphism)**.** A compressible encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Compress})$ is *B'-bounded limited homomorphic* with n slots and moduli p', $(p_i)_{i\in[n]}$ with distinguishing error $\varepsilon_{\mathsf{mb}} = \varepsilon_{\mathsf{mb}}(\lambda, t, m)$ in the generic group model if there exists a poly time simulator $\mathcal{S}$ such that that for every plaintext generator $\mathcal{T}$ and oracle machine $\mathcal{A}$ makes t to the GGM oracle the following distributions have statistical distance at most $\varepsilon_{\mathsf{mb}}$:

- **Real world:**
  1. Sample $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$.
  2. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$.
  3. $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$.
  4. $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{a}_i)$ for all $i \in [m]$.
  5. $(\mathsf{cct}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m)$.
  6. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \perp$ output $\perp$
  7. $(a'_1, \ldots, a'_n) \leftarrow \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$
  8. Output $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \ldots, a'_n)$.
- **Ideal world:**
  1. Sample $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$.
  2. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$.
  3. $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$.
  4. $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{a}_i)$ for all $i \in [m]$.
  5. $(\mathsf{cct}, \mathsf{st}_{\mathcal{A}}) \leftarrow_{\mathsf{tr}} \mathcal{A}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m)$, where tr is the trace of queries $\mathcal{A}$ made to the generic group oracle and the oracle's responses.
  6. $(\Pi \in \mathbb{Z}_{p'}^{m}, b_1 \in \mathbb{Z}_{p_1}, \ldots, b_1 \in \mathbb{Z}_{p_n}) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m, \mathsf{cct})$.
  7. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \perp$ output $\perp$.
  8. $a'_i = \mathbb{Z}_{p_i}\left(\sum_{j\in[m]} \Pi_j \cdot \mathbf{a}_j[i]\right) + b_i$ for all $i \in [n]$.
  9. If there exists an i such that $a'_i \notin [-B', B']$ output $\perp$.
  10. Output $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \ldots, a'_n)$.

As with the definition of compressible encryption schemes, here we also allow a random oracle H which is sampled together with $\mathcal{G}$ in Item 1. We say that the encryption scheme is bound-limited homomorphic if $t = \mathsf{poly}(\lambda)$ we have $\varepsilon_{\mathsf{ih}}(\lambda, m, t) = \mathsf{negl}(\lambda)$.

4.4.2 Isolated Homomorphism of Packed ElGamal

In this section, we prove that the packed ElGamal encryption scheme is isolated homomorphic.

Lemma 4.4.3. *Packed ElGamal is isolated homomorphic in the GGM with a distinguishing error $\varepsilon_{\mathsf{ih}}(\lambda, t, m) = 4t^2/p' \leq 4t^2/2^{\lambda}$ for $t > 4t_{\mathsf{Dec}} + 1$, where t_{Dec} is the number of queries the decryption algorithm does.*

Proof. Let p' be the size of the GGM group. We design a simulator for the isolated homomorphism experiment:

1. In the beginning of the security game, the simulator receives the trace tr, public key $\mathsf{pk} := (h_j)_{j\in[n]}$, and $\mathsf{ct}_i := (\mathsf{ct}_{i,0}, (\mathsf{ct}_{i,j})_{j\in[n]})$ for $i \in [m]$. The simulator initializes an empty table T and for $i \in [m]$ and $j \in [n]$ adds the expressions

$$g \mapsto 1, \qquad h_j \mapsto \hat{x}_j, \qquad \mathsf{ct}_{i,0} \mapsto \hat{r}_i, \qquad \text{and} \qquad \mathsf{ct}_{i,j} \mapsto \hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j]$$

to the table where $\hat{x}_j$, $\hat{r}_i$, and $\hat{\mathbf{a}}_i[j]$ are formal variables representing secret key, randomness and messages respectively.

2. The simulator goes through the trace tr first to last entry and does the following for each entry:

   Each entry has two input labels handles ξ_1 and ξ_2, the simulator checks whether there are mappings T from ξ_1 and ξ_2 to polynomials over formal variables Φ_1 and Φ_2 in the table, respectively. If ξ_i does not map to a polynomial in the table T and the existing formal variables are $(\hat{u}_j)_{j\in[\ell]}$ then the simulator generates a new formal variable $\hat{u}_{\ell+1}$ and adds the mapping $\xi_i \mapsto \hat{u}_{\ell+1}$ into the table T. Now, ξ_1 and ξ_2 both have mappings in T to polynomials over formal variables Φ_1, Φ_2.

   The simulator computes the polynomial $\Phi_3 := \Phi_1 + \Phi_2$. The simulator looks at the output label of the trace entry ξ_3. If the table T contains an entry $\xi_3 \mapsto \Phi_3'$ for some polynomial Φ_3' then the simulator outputs $\perp$ if $\Phi_3' \not\equiv \Phi_3$.

3. The simulator also has the adversary's output $\mathsf{cct} := (\mathsf{cct}_0, (e_j')_{j\in[n]})$. The simulator checks whether T contains mappings $\mathsf{cct}_0 \mapsto \Phi_0'$, where Φ_0' is a polynomial over formal variables equivalent to $\sum_{i\in[m]} \alpha_i \hat{r}_i + \beta$ for $\alpha_1, \ldots, \alpha_m, \beta \in \mathbb{Z}_{p'}$. If this is not the case, then for each $j \in [n]$ the simulator samples a uniform label v_j not used in T yet, and outputs $f_1, \ldots, f_n$ functions that entirely ignore the input:

   - $f_j(\mathbf{a}_1[j], \ldots, \mathbf{a}_m[j])$: Output $(\mathsf{DDL}_g(v_j) - e_j) \mod q$.

4. If the simulator has reached this point it outputs the functions $f_1, \ldots, f_n$ defined below. The functions have the labels for $(\mathsf{ct}_j^{(i)})_{i\in[m]}$ hardcoded, which correspond to the polynomials $\hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j]$

   - $f_j(\mathbf{a}_1[j], \ldots, \mathbf{a}_m[j])$:
     (a) Let $u_j \leftarrow \sum_i \alpha_i \mathbf{a}_i[j]$.
     (b) Compute a label v_j for the polynomial over formal variables $\sum_i \alpha_i(\hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j]) + \beta\hat{x}_j - u_j$.
     (c) Output $(\mathsf{DDL}_g(v_j) - e_j) \mod q$.

We show the extracted function outputs the correct distribution if the output ciphertext decrypts successfully with overwhelming probability. We prove this via hybrid argument.

Hyb_0**:** It is the same as the real distribution in Definition 4.4.1. In more detail:

1. In the beginning of the security game, for each $j \in [n]$ the experiment samples $x_j \xleftarrow{\$} \mathbb{Z}_{p'}$. The experiment initializes an empty table T and adds the mapping $g \mapsto 1$. For $j \in [n]$ the experiment checks whether there already exists a mapping from to x_j. If not the experiment samples a new distinct label h_j and adds $h_j \mapsto x_j$ to the table T. It sets the public key $\mathsf{pk} := (h_j)_{j\in[n]}$ and the secret key $\mathsf{sk} := (x_j)_{j\in[n]}$.
2. Then the experiment samples the plaintexts and a state with the plaintext generator $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$ and corresponding ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ using the generic group for each $i \in [m]$. More specifically, $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ samples $r_i \xleftarrow{\$} \mathbb{Z}_{p'}$ uniformly at random and for the values r_i and $r_i x_j + \mathbf{a}_i[j]$

for $j \in [n]$ checks whether there already is a mapping in T if not it adds a random label that is uniformly random from the label space $\mathcal{L}$ and different from all existing labels. Now, there are labels $\mathsf{ct}_{i,0}$ and $\mathsf{ct}_{i,j}$ such that the mappings

$$\mathsf{ct}_{i,0} \mapsto r_i, \qquad \text{and} \qquad \mathsf{ct}_{i,j} \mapsto r_i x_j + \mathbf{a}_i[j]$$

are in the table T.

3. The experiment sends the public key pk and the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$ to the adversary $\mathcal{A}$. When the adversary $\mathcal{A}$ uses its oracle access to the generic group the experiment does the following:

   When the adversary queries the generic group with the two handles ξ_1, ξ_2 the experiment checks whether there are mappings in the table T from ξ_1 and ξ_2 to $\mathbb{Z}_{p'}$ elements Φ_1, Φ_2 respectively. If for $i \in \{1, 2\}$ we have ξ_i does not map to a $\mathbb{Z}_{p'}$ element in T then the experiment samples a new distinct $\mathbb{Z}_{p'}$ element u_i and adds the mapping $\xi_i \mapsto u_i$ into the table T.

   Now, ξ_1 and ξ_2 have mappings in T to $\mathbb{Z}_{p'}$ elements Φ_1, Φ_2. Compute $\Phi_3 := \Phi_1 + \Phi_2$. If the table T contains an entry $\xi_3 \mapsto \Phi_3$ for some ξ_3 the experiment forwards ξ_3 to the adversary $\mathcal{A}$. Otherwise, there is no entry for Φ_3 in the table T. The experiment then samples a new distinct label ξ_3 from $\mathcal{L}$ and adds an entry $\xi_3 \mapsto \Phi_3$ to the table T and forwards ξ_3 to the adversary $\mathcal{A}$.

   The adversary finally outputs a ciphertext cct, a state $\mathsf{st}_{\mathcal{A}}$ and the trace of its GGM queries tr.

4. The experiment sends the trace tr, the public key pk, the input ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$, and the output ciphertext cct to the simulator $\mathcal{S}$. The simulator outputs n functions $(f_j)_{j \in [n]}$.

5. Let $(a'_j)_{j \in [n]} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{cct})$. The experiment outputs the states and messages $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \ldots, a'_n)$. That is because for Packed ElGamal decryption algorithm Dec never outputs $\perp$.

Hyb_1: Same as Hyb_0 but keeps track of generic group queries with formal variables as in the simulator $\mathcal{S}$. More specifically, the experiment behaves in the following way:

1. In the beginning of the security game, the experiment samples distinct labels $g, (h_j)_{j \in [n]} \xleftarrow{\$} \mathcal{L}$. For $j \in [n]$ the experiment samples $x_j \xleftarrow{\$} \mathbb{Z}_{p'}$. The experiment initializes an empty table T and adds the mappings $g \mapsto 1$ and $h_j \mapsto \hat{x}_j$ where $\hat{x}_j$ are formal variables. It sets the public key $\mathsf{pk} := (h_j)_{j \in [n]}$ and the secret key $\mathsf{sk} := (x_j)_{j \in [n]}$.

2. Then the experiment samples the plaintexts and a state with the plaintext generator $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$ and corresponding ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ using the generic group for each $i \in [m]$. More specifically, $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ samples $r_i \xleftarrow{\$} \mathbb{Z}_{p'}$ uniformly at random and distinct labels $\mathsf{ct}_{i,0}$ and $(\mathsf{ct}_{i,j})_{j \in [n]}$ that are uniformly random under the condition that they are not yet in T. The experiment adds the mappings

$$\mathsf{ct}_{i,0} \mapsto \hat{r}_i, \qquad \text{and} \qquad \mathsf{ct}_{i,j} \mapsto \hat{r}_i \hat{x}_j + \hat{\mathbf{a}}_i[j]$$

   where $j \in [n]$ to the table T.

3. The experiment sends the public key pk and the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$. When the adversary $\mathcal{A}$ also uses it oracle access to the generic group.

   If the adversary queries the generic group with the two handles ξ_1, ξ_2 the experiment checks whether there are mappings in the table T from ξ_1 and ξ_2 to polynomials over formal variables Φ_1, Φ_2 respectively. If for $i \in \{1, 2\}$ we have ξ_i does not map to a polynomial over formal variables in T and the existing formal variables are $(\hat{u}_j)_{j\in[\ell]}$ then the experiment generates a new formal variable $\hat{u}_{\ell+1}$ and adds the mapping $\xi_i \mapsto \hat{u}_{\ell+1}$ into the table T. Further, the experiment samples a new distinct $\mathbb{Z}_{p'}$ element $u_{\ell+1}$.

   Now, ξ_1 and ξ_2 have mappings in T to polynomials over formal variables Φ_1, Φ_2. Compute $\Phi_3 := \Phi_1 + \Phi_2$. If the table T contains an entry $\xi_3 \mapsto \Phi_3$ for some ξ_3 the experiment forwards ξ_3 to the adversary $\mathcal{A}$. Otherwise, there is no entry for Φ_3 in the table T. The experiment then samples a new distinct label ξ_3 from $\mathcal{L}$ and adds an entry $\xi_3 \mapsto \Phi_3$ to the table T and forwards ξ_3 to the adversary $\mathcal{A}$.

   The adversary finally outputs a ciphertext cct, a state $\mathsf{st}_{\mathcal{A}}$ and the trace of its GGM queries tr.

4. Then the experiment instantiates all the formal variables $(\hat{x}_j)_{j\in[n]}$, $(\hat{r}_i)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])$, and $(\hat{u}_i)_{i\in[\ell]}$ by their respective $\mathbb{Z}_{p'}$ values $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])$, and $(u_i)_{i\in[\ell]}$. Now T is a table that maps from labels to $\mathbb{Z}_{p'}$ elements. Continue as in Hyb_0.

Hyb_2: Same as Hyb_1 except that checking the well-formedness of the adversary's output $\mathsf{cct} := (\mathsf{cct}_0, (e'_j)_{j\in[n]})$ using the formal variables instead of the $\mathbb{Z}_{p'}$ values. Instead of the final step of Hyb_1 it does the following:

1. At the end of the simulation of the adversary $\mathcal{A}$, it, among other things, outputs a ciphertext $\mathsf{cct} := (\mathsf{cct}_0, (e'_j)_{j\in[n]})$. The experiment checks whether T contains mappings $\mathsf{cct}_0 \mapsto \Phi'_0$, where Φ'_0 is a polynomial over formal variables equivalent to $\sum_{i\in[m]} \alpha_i \hat{r}_i + \beta$ for $\alpha_1, \ldots, \alpha_m, \beta \in \mathbb{Z}_{p'}$. If this is not the case, for $j \in [n]$ compute the experiment samples unused labels v_j outputs the functions:
   - $f_j(\mathbf{a}_1[j], \ldots, \mathbf{a}_m[j])$: Output $(\mathsf{DDL}_g(v_j) - e_j) \mod q$.
2. Then the experiment instantiates all the formal variables $(\hat{x}_j)_{j\in[n]}$, $(\hat{r}_i)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])$, and $(\hat{u}_i)_{i\in[\ell]}$ by their respective $\mathbb{Z}_{p'}$ values $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])$, and $(u_i)_{i\in[\ell]}$. Now T is a table that maps from labels to $\mathbb{Z}_{p'}$ elements.
3. The experiment outputs

$$(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, f_j(\mathbf{a}_1[1], \ldots, \mathbf{a}_m[1]), \ldots, f_j(\mathbf{a}_1[n], \ldots, \mathbf{a}_m[n])) \ . \tag{4.1}$$

We argue Hyb_2 is identically distributed to the ideal distribution. This follows from the decryption algorithm $\mathsf{Dec}(\mathsf{sk}, \mathsf{cct} := (\mathsf{cct}_0, (e_j)_{j\in[n]}))$ outputting $\mathsf{DDL}(\mathsf{cct}_j) - e_j$ with $j \in [n]$ and cct_j being the label for the value $\Phi_0 \cdot x_j$, which exactly matches the simulator's output functions after instantiating $(\hat{x}_j)_{j\in[n]}$, $(\hat{r}_i)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$, and $(\hat{u}_i)_{i\in[\ell]}$ by their corresponding values $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$, and $(u_i)_{i\in[\ell]}$. That is because after instantiation

$$\sum_i \alpha_i(\hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j]) + \beta\hat{x}_j - \sum_i \alpha_i \mathbf{a}_i[j]$$

becomes

$$\sum_i \alpha_i(r_i x_j + \mathbf{a}_i[j]) + \beta x_j - \sum_i \alpha_i \mathbf{a}_i[j] = \sum_i \alpha_i r_i x_j + \beta x_j - \sum_i = \Phi_0 \cdot x_j.$$

Therefore, $f_j(\mathbf{a}_1[j], \ldots, \mathbf{a}_m[j]) = a'_j$ and the distributions are the same.

Finally, we argue statistical distance of the hybrids for polynomial generic group queries. Statistical distance between Hyb_0 and Hyb_1 is established in Claim 4.4.4 and the statistical distance between Hyb_1 and Hyb_2 in Claim 4.4.5. The sum of these distances is $< 4t^2/p'$ for $t > 4t_{\mathsf{Dec}} + 1$, where t_{Dec} is the number of queries the decryption algorithm does. □

Claim 4.4.4. For any adversary $\mathcal{A}$ making t many queries to the generic group oracle $\mathsf{Hyb}_0(\mathcal{A})$ has a statistical distance of $3t(t+1)/p'$ from $\mathsf{Hyb}_1(\mathcal{A})$.

Proof. We show that the view of the adversary $\mathcal{A}$ is statistically close in the two hybrid if it only makes polynomially many queries to the generic group oracle. We argue that the public key pk, the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$ and adversary's queries to the generic group are statistically close in Hyb_0 and Hyb_1. We argue via a hybrid argument over each query to the generic group. We start with $\mathsf{Hyb}_{0,0}$ which is identically distributed to Hyb_0 and with $\mathsf{Hyb}_{0,s}$ the first s computed elements are as in Hyb_1 and all the rest are still as in Hyb_0. We then show that $\mathsf{Hyb}_{0,s-1}$ and $\mathsf{Hyb}_{0,s}$ are statistically close. It suffices to consider the i-th computed element:

- The hybrids only differ in behavior if the label ξ given to the adversary maps to a polynomial over formal variables Φ that has a non-zero linear term in $(\hat{r}_i)_{i\in[m]}$, $(\hat{r}_i\hat{x}_j)_{i\in[m],j\in[n]}$, or $(\hat{u}_i)_{i\in[\ell]}$ monomials while $\Phi((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (\mathbf{a}_i[j])_{i\in[m],j\in[n]}, (u_i)_{i\in[\ell]})$ is already in the table T. This condition is equivalent to the condition that there exists a constant $c \in \mathbb{Z}_{p'}$ in the table T such that $\Phi((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}, (\hat{u}_i)_{i\in[\ell]}) - c \not\equiv 0$, but $\Phi((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (\mathbf{a}_i[j])_{i\in[m],j\in[n]}, (u_i)_{i\in[\ell]}) - c = 0$. We show this happens with negligible probability. In both, $\mathsf{Hyb}_{0,s-1}$ and $\mathsf{Hyb}_{0,s}$ the first $s-1$ elements are handled as in Hyb_1. Therefore, these elements are independent of $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$, $(u_i)_{i\in[\ell]}$. Fix an arbitrary constant $c \in \mathbb{Z}_{p'}$ from the table T. We define a new polynomial

$$\begin{aligned}&\Psi_c((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{u}_i)_{i\in[\ell]})\\ =&\Phi((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\mathbf{a}_i[j])_{i\in[m],j\in[n]}, (\hat{u}_i)_{i\in[\ell]}) - c.\end{aligned}$$

  Notice, that we instantiated $(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$ by $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$. In the polynomial $\Phi((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}, (\hat{u}_i)_{i\in[\ell]})$ we have monomial $\hat{\mathbf{a}}_i[j]$ always with the same arity as the corresponding monomial $\hat{r}_i\hat{x}_j$ for $i \in [m], j \in \{0,1\}$ because the challenger initially only gives out labels for polynomials of the form $\hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j]$ and the oracles only allow the adversary to learn linear combinations. Therefore, if the polynomial Ψ_c is the zero polynomial then so is $\Phi((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}, (\hat{u}_i)_{i\in[\ell]}) - c$.

  Because Ψ_c is a polynomial of total degree 2 we derive by polynomial identity lemma that

$$\Pr[\Psi_c((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{u}_i)_{i\in[\ell]}) = 0] \leq 2/p'.$$

  By union bound we get that

$$\Pr[\exists c \in \mathbb{Z}_{p'} \cap \mathsf{T} : \Psi_c((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{u}_i)_{i\in[\ell]}) = 0] \leq 2|\mathsf{T}|/p'.$$

  Thus, with probability $2|\mathsf{T}|/p'$ for Φ degree ≥ 1 we have the value $\Phi((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (\mathbf{a}_i[j])_{i\in[m],j\in[n]})$ is not in the table T. Therefore, $\mathsf{Hyb}_{0,s-1}$ and $\mathsf{Hyb}_{0,s}$ have the same behavior with probability $1 - 2|\mathsf{T}|/p'$.

If the number of queries to the generic group oracle is ℓ then $\mathsf{Hyb}_{0,0}$ is identically distributed to Hyb_0 and $\mathsf{Hyb}_{0,\ell}$ is identically distributed to Hyb_1. Because every query introduces at most 3 entries into T we get the statistical distance between Hyb_0 and Hyb_1 is at most $\sum_{i\in[\ell]} 2(3i)/p' = 3\ell(\ell+1)/p'$.

□

Claim 4.4.5. For any adversary $\mathcal{A}$ making $t_{\mathcal{A}}$ many queries and Dec making t_{Dec} queries to the generic group oracle $\mathsf{Hyb}_1(\mathcal{A})$ has a statistical distance of $3t_{\mathsf{Dec}}(t_{\mathcal{A}} + (t_{\mathsf{Dec}}+1)/2)/p'$ from $\mathsf{Hyb}_2(\mathcal{A})$.

Proof. The view of the adversary in Hyb_1 and Hyb_2 are identical, therefore, all that is left to prove is that decryption of the output ciphertext is statistically close. Notice that the adversary's view is independent of $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$, and $(u_i)_{i\in[\ell]}$, therefore we can treat them as being sampled after the adversary outputs the ciphertext. We analyze the following three scenarios:

- If the label in the output ciphertext cct_0 is not in the table T: Without loss of generality this does not happen because one can always introduce a new formal variable $\hat{u}_{\ell+1}$, where $(\hat{u}_i)_{i\in[\ell]}$ are the variables that represent unknown elements, and add $\mathsf{cct}_0 \mapsto \hat{u}_{\ell+1}$ to the table T and this case reduces to the next one.

- The table T contains a mapping from the label of the output ciphertext $\mathsf{cct} := (\mathsf{cct}_0, (e'_j)_{j\in[n]})$ to a polynomial Φ'_0 such that $\Phi'_0 \not\equiv \sum_{i\in[m]} \alpha_i \hat{r}_i + \beta$ for $\alpha_1, \ldots \alpha_m, \beta \in \mathbb{Z}_{p'}$:

  Now, we have that Φ'_0 contains one of the following monomials: $\hat{r}_i\hat{x}_j$, $\hat{x}_j$, or $(\hat{u}_i)_{i\in[\ell]}$ for $i \in [m]$ and $j \in [n]$. Therefore, the adversary interacted with the group oracle involving the label cct_ℓ (either used is as input or received it as output) for $\Phi'_0 \cdot \hat{x}_\ell$ with $t \in [n]$ as it can only linearly combine 1, $(\hat{x}_j)_{j\in[n]}$, $(r_i\hat{x}_j + \hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$, and $(\hat{u}_i)_{i\in[\ell]}$. Therefore, Hyb_1 and Hyb_2 only differ if in Hyb_1 the $\mathsf{DDL}(\mathsf{cct}_t)$ algorithm queries an entry that has already been assigned to a value. We argue via a sequence of hybrids that the probability of this happening is negligible. To prove this we use the property that $\mathsf{DDL}_g(\mathsf{cct}_t)$ only has access to g (the label for 1) and its input label cct_t. We start with $\mathsf{Hyb}_{1,0}$ which is identically distributed to Hyb_1 and then in $\mathsf{Hyb}_{1,s}$ the first i queries are handled with formal variables. We now show that $\mathsf{Hyb}_{1,s-1}$ and $\mathsf{Hyb}_{1,s}$ are statistically close. It suffices to consider the s-th computed element:

  - The hybrids only differ in behavior if for some $j \in [n]$ a label ξ that $\mathsf{DDL}(\mathsf{cct}_0^{x_j})$ received from the generic group orale maps to a degree ≥ 1 polynomial Φ over formal variables while $\Phi((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (\mathbf{a}_i[j])_{i\in[m],j\in[n]}, (u_i)_{i\in[\ell]})$ is already in the table T. This condition is equivalent to the condition that there exists a constant $c \in \mathbb{Z}_{p'}$ in the table T such that $\Phi((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}, (\hat{u}_i)_{i\in[\ell]}) - c \not\equiv 0$, but $\Phi((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (\mathbf{a}_i[j])_{i\in[m],j\in[n]}, (u_i)_{i\in[\ell]}) - c = 0$. We show this happens with negligible probability. In both, $\mathsf{Hyb}_{1,s-1}$ and $\mathsf{Hyb}_{1,s}$ the first $s-1$ elements are handled with formal variables.

    Therefore, $\mathsf{DDL}(\mathsf{cct}_\ell)$ has only the adversary interacted with the group oracle involving labels for linear combinations of 1, $\Phi'_0 \cdot \hat{x}_t$, and formal variables $(\hat{u}_{i+\ell})_{i\in[s-1]}$. So, the s-th query will be a linear combination of 1, $\Phi'_0 \cdot \hat{x}_t$, and formal variables $(\hat{u}_{i+\ell})_{i\in[s]}$ (it might have introduced a new formal variable). More precisely, this means that the query will output a label that maps to $\alpha + \beta \cdot \Phi'_0 \cdot \hat{x}_t + \sum_{i\in[s]} \gamma_i \hat{u}_{i+\ell}$ for some $\alpha, \beta, \gamma_i \in \mathbb{Z}_{p'}$. If $\beta = \gamma_1 = ... = \gamma_s = 0$

then the formal polynomial is not of degree ≥ 1 meaning $\mathsf{Hyb}_{1,s-1}$ and $\mathsf{Hyb}_{1,s}$ are identically distributed.

Further, as they only depend on formal variables, these elements are independent of $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$, $(u_i)_{i\in[\ell+s]}$. Fix an arbitrary constant $c \in \mathbb{Z}_{p'}$ from the table T. We define a new polynomial

$$\begin{aligned}&\Psi_{c,t}((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{u}_i)_{i\in[\ell+s]})\\ =&\alpha + \beta \cdot \Phi_0'((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{u}_i)_{i\in[\ell+s]}, (\mathbf{a}_i[j])_{i\in[m],j\in[n]}) \cdot \hat{x}_t - c.\end{aligned}$$

Notice, that we instantiated $(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$ by $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$. The polynomial $\Phi((\hat{x}_j)_{j\in[n]}, (\hat{r}_1)_{i\in[m]}, (\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}, (\hat{u}_i)_{i\in[\ell]})$ must at least contain one of the following monomials:

$$(\hat{r}_i\hat{x}_j\hat{x}_t)_{i\in[m],j,t\in[n]}, \qquad (\hat{x}_j\hat{x}_t)_{j,t\in[n]}, \qquad \text{or} \qquad (\hat{u}_i\hat{x}_t)_{i\in[\ell+s],t\in[n]}.$$

Therefore, if the polynomial $\Psi_{c,t}$ is the zero polynomial then so is $\Phi((\hat{x}_j)_{j\in[n]}$, $(\hat{r}_1)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$, $(\hat{u}_i)_{i\in[\ell+s]}) - c$.

Because Φ_0' is an comes from an output of the adversary and the adversary only knows polynomials of degree ≤ 2 and can only compute linear combinations we have Φ_0' is also of degree ≤ 2. It follows that $\Psi_{c,t}$ is ≤ 3 degree polynomial. Then, we derive by polynomial identity lemma that

$$\Pr[\Psi_{c,t}((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (u_i)_{i\in[\ell]}) = 0] \leq 3/p'.$$

By union bound we get that

$$\Pr[\exists c \in \mathbb{Z}_{p'} \cap \mathsf{T} : \Psi_{c,t}((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (u_i)_{i\in[\ell]}) = 0] \leq 3|\mathsf{T}|/p'.$$

Thus, with probability $3|\mathsf{T}|/p'$ for Φ with degree ≥ 1 we have the value $\Phi((x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(a_i[j])_{i\in[m],j\in[n]})$ is not in the table T. Therefore, $\mathsf{Hyb}_{0,s-1}$ and $\mathsf{Hyb}_{0,s}$ have the same behavior with probability $1 - 3|\mathsf{T}|/p'$.

We have $\mathsf{Hyb}_{1,0}$ is identically distributed to Hyb_1. Let $\ell_{\mathcal{A}}$ be the number of generic group oracle calls the adversary made and ℓ_{Dec} be the number of generic group oracle call Dec makes.

Further, the statistical distance between $\mathsf{Hyb}_{1,0}$ and $\mathsf{Hyb}_{1,\ell_{\mathsf{Dec}}}$ is

$$\sum_{i\in[\ell_{\mathsf{Dec}}]} 3(\ell_{\mathcal{A}} + i)/p' = 3\ell_{\mathsf{Dec}}(\ell_{\mathcal{A}} + (\ell_{\mathsf{Dec}} + 1)/2)/p'$$

because each query can introduce at most 3 terms.

Then Hyb_2 is identically distributed to $\mathsf{Hyb}_{1,\ell_{\mathsf{Dec}}}$ because as argued above the adversary does not have access to the formal polynomial $\Phi_0' \cdot \hat{x}_t$, therefore, from its perspective the corresponding label could also be chosen uniformly at random from the labels the adversary has not seen yet.

- The table T contains a mapping from each label of the output ciphertext $\mathsf{cct} := (\mathsf{cct}_0, (e_j')_{j\in[n]})$ to a polynomial Φ_0' such that $\Phi_0' \equiv \sum_{i\in[m]} \alpha_i\hat{r}_i + \beta$ for $\alpha_1, \ldots \alpha_m, \beta \in \mathbb{Z}_{p'}$:

  Hyb_1 and Hyb_2 behave identically in that case.

□

4.4.3 Bound-Limited Homomorphism of Packed ElGamal with Hash Check

In this section, we prove that Packed ElGamal with hash check has bound limited homomorphism.

Lemma 4.4.6. *Packed ElGamal with hash check for has $8Bn$-bound-limited homomorphism with distinguishability error $\varepsilon_{\mathsf{mb}}(\lambda, t, \chi, m) = 4t^2/p' + 2t^2/2^\chi \leq 6t^2/2^\lambda$ for $t > 4t_{\mathsf{Dec}} + 1$, where t_{Dec} is the number of queries the decryption algorithm does.*

Proof. For any adversary $\mathcal{A}$ and plaintext generator $\mathcal{T}$ in the bound-limited homomorphism experiment we define an extractor Ext:

1. In the beginning of the security game, the extractor receives the trace tr, the public key $\mathsf{pk} := (h_1, h_2)$, and $\mathsf{ct}_i := (\mathsf{ct}_{i,0}, (\mathsf{ct}_{i,j})_{j\in[n]})$ for $i \in [m]$. The extractor initializes an empty table T and for $i \in [m]$ and $j \in [n]$ adds the expressions

$$g \mapsto 1, \qquad h_j \mapsto \hat{x}_j, \qquad \mathsf{ct}_{i,0} \mapsto \hat{r}_i, \qquad \text{and} \qquad \mathsf{ct}_{i,j} \mapsto \hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j]$$

   to the table where $\hat{x}_j$, $\hat{r}_i$, and $\hat{\hat{a}}_i[j]$ are formal variables representing secret key, randomness and messages respectively for $i \in [m]$.

2. The simulator goes through the trace tr first to last entry and does the following for each entry:

   Each entry has two input labels handles ξ_1 and ξ_2, the simulator checks whether there are mappings T from ξ_1 and ξ_2 to polynomials over formal variables Φ_1 and Φ_2 in the table, respectively. If ξ_i does not map to a polynomial in the table T and the existing formal variables are $(\hat{u}_j)_{j\in[\ell]}$ then the simulator generates a new formal variable $\hat{u}_{\ell+1}$ and adds the mapping $\xi_i \mapsto \hat{u}_{\ell+1}$ into the table T. Now, ξ_1 and ξ_2 both have mappings in T to polynomials over formal variables Φ_1, Φ_2.

   The simulator computes the polynomial $\Phi_3 := \Phi_1 + \Phi_2$. The simulator looks at the output label of the trace entry ξ_3. If the table T contains an entry $\xi_3 \mapsto \Phi_3'$ for some polynomial Φ_3' then the simulator outputs $\perp$ if $\Phi_3' \not\equiv \Phi_3$.

3. The simulator also has the adversary's output $\mathsf{cct} := (\mathsf{cct}_0, (e_j)_{j\in[n]}, k)$. The simulator checks whether T contains mappings $\mathsf{cct}_0 \mapsto \Phi_0'$, where Φ_0' is a polynomial over formal variables equivalent to $\sum_{i\in[m]} \alpha_i\hat{r}_i + \beta$ for $\alpha_1, \ldots, \alpha_m, \beta \in \mathbb{Z}_{p'}$. If this is not the case, the simulator outputs $\perp$.

4. For each $j \in [n]$ the simulator computes labels cct_j corresponding to $\beta\hat{x}_j + \sum_{i\in[m]} \alpha_i\hat{r}_i\hat{x}_j$.

5. For each $j \in [n]$ the simulator computes $v_j \leftarrow \mathsf{DDL}(\mathsf{cct}_j)$.

6. If the extractor has reached this point it outputs $(\Pi = (\alpha_1, \ldots, \alpha_m), \mathbf{b} = (v_j - e_j \mod p)_{j\in[n]})$.

We show the simulated linear function outputs the correct value if the output ciphertext decrypts successfully with all but negligible probability. We prove this via hybrid argument.

Hyb_0**:** It is the same as the real distribution in Definition 4.4.2. In more detail:

1. In the beginning of the security game, for each $j \in [n]$ the experiment samples $x_j \xleftarrow{\$} \mathbb{Z}_{p'}$. The experiment initializes an empty table T and adds the mapping $g \mapsto 1$. For $j \in [n]$ the experiment checks whether there already exists a mapping from to x_j. If not the experiment samples a new distinct label h_j and adds $h_j \mapsto x_j$ to the table T. It sets the public key $\mathsf{pk} := (h_j)_{j\in[n]}$ and the secret key $\mathsf{sk} := (x_j)_{j\in[n]}$.
2. Then the experiment samples the plaintexts and a state from the plaintext generator $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$ and corresponding ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ using the generic group for each $i \in [m]$. More specifically, $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ samples $r_i \xleftarrow{\$} \mathbb{Z}_{p'}$ uniformly at random and for the values r_i and $r_i x_j + \mathbf{a}_i[j]$ for $j \in [n]$ checks whether there already is a mapping in T if not it adds a random label that is uniformly random from the label space $\mathcal{L}$ and different from all existing labels. Now, there are labels $\mathsf{ct}_{i,0}$ and $\mathsf{ct}_{i,j}$ such that the mappings

$$\mathsf{ct}_{i,0} \mapsto r_i, \qquad \text{and} \qquad \mathsf{ct}_{i,j} \mapsto r_i x_j + \mathbf{a}_i[j]$$

   are in the table T.
3. The experiment sends the public key pk and the input ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$ to the adversary $\mathcal{A}$. When the adversary $\mathcal{A}$ uses its oracle access to the generic group the experiment does the following:

   When the adversary queries the generic group with the two handles ξ_1, ξ_2 the experiment checks whether there are mappings in the table T from ξ_1 and ξ_2 to $\mathbb{Z}_{p'}$ elements Φ_1, Φ_2 respectively. If for $i \in \{1, 2\}$ we have ξ_i does not map to a $\mathbb{Z}_{p'}$ element in T then the experiment samples a new distinct $\mathbb{Z}_{p'}$ element u_i and adds the mapping $\xi_i \mapsto u_i$ into the table T.

   Now, ξ_1 and ξ_2 have mappings in T to $\mathbb{Z}_{p'}$ elements Φ_1, Φ_2. Compute $\Phi_3 := \Phi_1 + \Phi_2$. If the table T contains an entry $\xi_3 \mapsto \Phi_3$ for some ξ_3 the experiment forwards ξ_3 to the adversary $\mathcal{A}$. Otherwise, there is no entry for Φ_3 in the table T. The experiment then samples a new distinct label ξ_3 from $\mathcal{L}$ and adds an entry $\xi_3 \mapsto \Phi_3$ to the table T and forwards ξ_3 to the adversary $\mathcal{A}$.

   The adversary finally outputs a ciphertext cct, a state $\mathsf{st}_{\mathcal{A}}$ and the trace of its GGM queries tr.
4. The experiment sends the trace tr, the public key pk, the input ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$, and the output ciphertext cct to the simulator $\mathcal{S}$. The simulator outputs Π and $\mathbf{b}$.
5. If $\mathsf{Dec}(\mathsf{sk}, \mathsf{cct}) = \perp$ the experiment outputs $\perp$. More specifically $\mathsf{Dec}(\mathsf{sk}, \mathsf{cct} := (\mathsf{cct}_0, (e'_j)_{j\in[n]}, k'))$ checks if

$$k = \mathsf{H}(c_0, (c_0^{x_j} \cdot g^{\mathsf{DDL}(c_0^{x_j})})_{j\in[n]}).$$

   If this is not the case the experiment outputs $\perp$.
6. Let $(a'_j)_{j\in[n]} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{cct})$. The experiment outputs $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \ldots, a'_n)$.

Hyb_1: Same as Hyb_0 but keeps track of generic group queries with formal variables as in the simulator $\mathcal{S}$. More specifically, the experiment behaves in the following way:

1. In the beginning of the security game, the experiment samples distinct labels $g, (h_j)_{j\in[n]} \xleftarrow{\$} \mathcal{L}$. For $j \in [n]$ the experiment samples $x_j \xleftarrow{\$} \mathbb{Z}_{p'}$. The experiment

initializes an empty table T and adds the mappings $g \mapsto 1$ and $h_j \mapsto \hat{x}_j$ where $\hat{x}_j$ are formal variables. It sets the public key $\mathsf{pk} := (h_j)_{j\in[n]}$ and the secret key $\mathsf{sk} := (x_j)_{j\in[n]}$.

2. Then the experiment samples the plaintexts and a state from the plaintext generator $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$ and corresponding ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ using the generic group for each $i \in [m]$. More specifically, $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ samples $r_i \xleftarrow{\$} \mathbb{Z}_{p'}$ uniformly at random and distinct labels $\mathsf{ct}_{i,0}$ and $(\mathsf{ct}_{i,j})_{j\in[n]}$ that are uniformly random under the condition that they are not yet in T. The experiment adds the mappings

$$\mathsf{ct}_{i,0} \mapsto \hat{r}_i, \qquad \text{and} \qquad \mathsf{ct}_{i,j} \mapsto \hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j]$$

where $j \in [n]$ to the table T.

3. The experiment sends the public key pk and the input ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$. When the adversary $\mathcal{A}$ also uses it oracle access to the generic group.

   If the adversary queries the generic group with the two handles ξ_1, ξ_2 the experiment checks whether there are mappings in the table T from ξ_1 and ξ_2 to polynomials over formal variables Φ_1, Φ_2 respectively. If for $i \in \{1, 2\}$ we have ξ_i does not map to a polynomial over formal variables in T and the existing formal variables are $(\hat{u}_j)_{j\in[\ell]}$ then the experiment generates a new formal variable $\hat{u}_{\ell+1}$ and adds the mapping $\xi_i \mapsto \hat{u}_{\ell+1}$ into the table T. Further, the experiment samples a new distinct $\mathbb{Z}_{p'}$ element $u_{\ell+1}$.

   Now, ξ_1 and ξ_2 have mappings in T to polynomials over formal variables Φ_1, Φ_2. Compute $\Phi_3 := \Phi_1 + \Phi_2$. If the table T contains an entry $\xi_3 \mapsto \Phi_3$ for some ξ_3 the experiment forwards ξ_3 to the adversary $\mathcal{A}$. Otherwise, there is no entry for Φ_3 in the table T. The experiment then samples a new distinct label ξ_3 from $\mathcal{L}$ and adds an entry $\xi_3 \mapsto \Phi_3$ to the table T and forwards ξ_3 to the adversary $\mathcal{A}$.

   The adversary finally outputs a ciphertext cct, a state $\mathsf{st}_{\mathcal{A}}$ and the trace of its GGM queries tr.

4. Then the experiment instantiates all the formal variables $(\hat{x}_j)_{j\in[n]}$, $(\hat{r}_i)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])$, and $(\hat{u}_i)_{i\in[\ell]}$ by their respective $\mathbb{Z}_{p'}$ values $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])$, and $(u_i)_{i\in[\ell]}$. Now T is a table that maps from labels to $\mathbb{Z}_{p'}$ elements. Continue as in Hyb_0.

Hyb_2**:** Same as Hyb_1 but instead of the last step the experiment does the following:

1. For the ciphertext $\mathsf{cct} := (\mathsf{cct}_0, (e_j)_{j\in[n]}, k)$ if there is no mapping $\mathsf{cct}_0 \mapsto \Phi_0'$ in the table T such that $\Phi_0' \equiv \sum_{i\in[m]} \alpha_i\hat{r}_i + \beta$ for $\alpha_1, \ldots \alpha_m, \beta \in \mathbb{Z}_{p'}$ the experiment samples n uniform unused labels $(\mathsf{cct}_j)_{j\in[n]}$.
2. If $\mathsf{H}(\mathsf{cct}_0, (\mathsf{cct}_j \cdot g^{\mathsf{DDL}(\mathsf{cct}_j)})_{j\in[n]}) \neq k$ the experiment outputs $\perp$.
3. Otherwise, the experiment instantiates all formal variables $(\hat{x}_j)_{j\in[n]}$, $(\hat{r}_i)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$, and $(\hat{u}_i)_{i\in[\ell]}$ in T by their respective values $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$, and $(u_i)_{i\in[\ell]}$. Now T is a table that maps from labels to $\mathbb{Z}_{p'}$ elements.
4. The experiment computes $(a_j')_{j\in[n]} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{cct})$. The experiment outputs $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a_1', \ldots, a_n')$.

Hyb_3**:** Same as Hyb_2 but instead of creating dummy variables if there is no mapping $\mathsf{cct}_0 \mapsto \Phi_0'$, we simply output $\perp$.

1. For the ciphertext $\mathsf{cct} := (\mathsf{cct}_0, (e_j)_{j\in[n]}, k)$ if there is no mapping $\mathsf{cct}_0 \mapsto \Phi'_0$ in the table T such that $\Phi'_0 \equiv \sum_{i\in[m]} \alpha_i \hat{r}_i + \beta$ for $\alpha_1, \ldots \alpha_m, \beta \in \mathbb{Z}_{p'}$ the experiment outputs $\perp$.
2. If $\mathsf{H}(\mathsf{cct}_0, (\mathsf{cct}_j \cdot g^{\mathsf{DDL}(\mathsf{cct}_j)})_{j\in[n]}) \neq k$ the experiment outputs $\perp$.
3. Otherwise, the experiment instantiates all formal variables $(\hat{x}_j)_{j\in[n]}$, $(\hat{r}_i)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$, and $(\hat{u}_i)_{i\in[\ell]}$ in T by their respective values $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$, and $(u_i)_{i\in[\ell]}$. Now T is a table that maps from labels to $\mathbb{Z}_{p'}$ elements.
4. The experiment computes $(a'_j)_{j\in[n]} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{cct})$. The experiment outputs $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \ldots, a'_n)$.

Hyb_4**:** Same as Hyb_3 but instead of the final step the experiment now does the following:

1. For a $j \in [n]$, with cct_j being the label in T for $\beta\hat{x}_j + \sum_{i\in[m]} \alpha_i(\hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j])$, and cct^*_j being the label for $\beta\hat{x}_j + \sum_{i\in[m]} \alpha_i(\hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j] - \mathbf{a}_i[j])$ if
$$\mathsf{cct}^*_j \cdot g^{\mathsf{DDL}(\mathsf{cct}^*_j)} \neq \mathsf{cct}_j \cdot g^{\mathsf{DDL}(\mathsf{cct}_j)}.$$
the experiment outputs $\perp$.
2. Otherwise, the experiment instantiates all formal variables $(\hat{x}_j)_{j\in[n]}$, $(\hat{r}_i)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$, and $(\hat{u}_i)_{i\in[\ell]}$ in T by their respective values $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$, and $(u_i)_{i\in[\ell]}$. Now T is a table that maps from labels to $\mathbb{Z}_{p'}$ elements.
3. The experiment computes $(a'_j)_{j\in[n]} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{cct})$. The experiment outputs $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \ldots, a'_n)$.

We argue that in Hyb_4 the adversaries winning probability is 0. First, note that if for all $j \in [n]$ we have $\mathsf{cct}^*_j \cdot g^{\mathsf{DDL}(\mathsf{cct}^*_j)} = \mathsf{cct}_j \cdot g^{\mathsf{DDL}(\mathsf{cct}_j)}$ then $\sum_{i\in[m]} \alpha_i \mathbf{a}_i[j] \in [-8Bn, 8Bn]$ by contrapositon of Lemma 2.6.6 because DDL does $8Bn$ queries.

Then, we argue that decryption outputs exactly what the extractor outputs. cct^*_j is a label for $\beta\hat{x}_j + \sum_{i\in[m]} \alpha_i(\hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j] - \mathbf{a}_i[j])$. After instantiating $(\hat{r}_i)_{i\in[m]}$, $\hat{x}_j$, and $(\hat{\mathbf{a}}_i[j])_{i\in[m]}$ by their values $(r_i)_{i\in[m]}$, x_j, and $(\mathbf{a}_i[j])_{i\in[m]}$ this exactly corresponds to $\sum_{i\in[m]} \alpha_i r_i x_j + \beta x_j$. Therefore, Dec computes $\mathsf{DDL}(\mathsf{cct}^*_j)$ and outputs

$$(\mathsf{DDL}(\mathsf{cct}^*_j) - e_j) \mod p = (\mathsf{DDL}(\mathsf{cct}_j) + \sum_{i\in[m]} \alpha_i \mathbf{a}_i[j] - e_j) \mod p.$$

This is exactly what the experiment outputs in the ideal world because $\mathsf{DDL}(\mathsf{cct}^*_j)$ and e_j are much smaller than the cryptographic group.

Finally, we argue statistical distance of the hybrids for polynomial generic group queries. Statistical distance between Hyb_0 and Hyb_1 is established using the exact same arguments as Claim 4.4.4 and the statistical distance between Hyb_1 and Hyb_2 is established using the exact same arguments as Claim 4.4.5. The statistical distance between Hyb_2 and Hyb_4 is established in Claims 4.4.7 and 4.4.8. The sum of these distances is $< (t+1)^2/p'$ for $t > 4t_{\mathsf{Dec}} + 1$, where t_{Dec} is the number of queries the decryption algorithm does. □

Claim 4.4.7. For any adversary $\mathcal{A}$ we have $\mathsf{Hyb}_2(\mathcal{A})$ and $\mathsf{Hyb}_3(\mathcal{A})$ are at statistical distance $2^{-\chi}$.

Proof. Follows from H being a universal hash function with a codomain of $\{0,1\}^{\chi}$. □

Claim 4.4.8. For any adversary $\mathcal{A}$ with t queries to the ROM we have $\mathsf{Hyb}_3(\mathcal{A})$ are at statistical distance of $t^2/2^\chi$ to $\mathsf{Hyb}_4(\mathcal{A})$.

Proof. We prove that checking equality between

$$(\mathsf{cct}_j \cdot g^{\mathsf{DDL}(\mathsf{cct}_j)})_{j\in[n]} \qquad \text{and} \qquad (\mathsf{cct}^*_j \cdot g^{\mathsf{DDL}(\mathsf{cct}^*_j)})_{j\in[n]}$$

is statistically close close to checking the equality of their hashes. This follows from collision resistance of H. More precisely, if $(\mathsf{cct}_j \cdot g^{\mathsf{DDL}(\mathsf{cct}_j)})_{j\in[n]} \neq (\mathsf{cct}^*_j \cdot g^{\mathsf{DDL}(\mathsf{cct}^*_j)})_{j\in[n]}$ but $\mathsf{H}((\mathsf{cct}_j \cdot g^{\mathsf{DDL}(\mathsf{cct}_j)})_{j\in[n]}) = \mathsf{H}((\mathsf{cct}^*_j \cdot g^{\mathsf{DDL}(\mathsf{cct}^*_j)})_{j\in[n]})$ the adversary found a collision in the hash function. If H is modeled with the random oracle then the collision resistance is with probability $t(t+1)/2^{\chi+1}$. □

4.5 Constructing Linear PCPs and MIPs

In this section, we show how to adapt known linear PCPs into the information-theoretic objects underlying our dv-SNARG constructions. In Section 4.5.1 we show how to transform 2-query linear PCPs into 3-prover strong linear MIPs, and in Section 4.5.2, we show that linear PCPs can be transformed into modded bounded LPCPs.

4.5.1 Linear PCPs to Strong Linear MIPs

[IKO07] show that linear PCPs can be transformed into strong linear MIPs. However, their transformation garners a large loss in soundness error, and so requires many repetitions to achieve specific soundness errors. We give an alternate transformation for LPCPs with 2 queries which is more efficient in practice:

Lemma 4.5.1. *Let $\mathcal{R}$ be a relation with a smooth LPCP over $\mathbb{F}_p$ with length ℓ, query complexity 2, and knowledge soundness κ against affine strategies. There exists a strong LMIP for $\mathcal{R}$ over $\mathbb{F}_p$ with length ℓ, query complexity 3, and strong knowledge soundness $\frac{7}{9} + \frac{36}{505-729\cdot\kappa}$. If the LPCP is input-oblivious, then so is the strong LMIP. If the LPCP is B-bounded with error α, then the strong LMIP is $(B + 4p\sqrt{\ell\lambda})$-bounded with error $\alpha + 2^{-\lambda}$.*

Proof. We begin by showing that an LPCP can be made smooth with the addition of a single query:

Claim 4.5.2. Let $\mathcal{R}$ be a relation with an LPCP over $\mathbb{F}_p$ with length ℓ, query complexity q, and knowledge error κ against affine strategies. Then $\mathcal{R}$ has a smooth LPCP over $\mathbb{F}_p$ with length ℓ, query complexity $q+1$, and knowledge error κ against affine strategies. If the LPCP is input-oblivious, then so is the smooth LPCP. If the LPCP is B-bounded with error α, then for every $\lambda \in \mathbb{N}$, the smooth LPCP is $(B + p^2\sqrt{\ell\lambda})$-bounded with error $\alpha + 2^{-\lambda}$.

Proof. Let $(\mathbf{P}, (V_Q, V_D))$ be the linear PCP for $\mathcal{R}$. We give a smooth linear PCP $(\mathbf{P}', (V'_Q, V'_D))$ for $\mathcal{R}$:

- $\mathbf{P}'(x, w)$: Output $\pi \leftarrow \mathbf{P}(x, w)$.
- $V'_Q(x)$:
  1. Sample $(\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x)$ and $\mathbf{a}'_0 \xleftarrow{\$} \mathbb{F}_p^\ell$.
  2. Output $(\mathsf{st}, \mathbf{a}'_0, \mathbf{a}'_1, \ldots, \mathbf{a}'_q)$ where $\mathbf{a}'_i = \mathbf{a}_i + \mathbf{a}'_0$.

- $V'_D(\mathsf{st}, x, b'_0, \ldots, b'_q)$: Output $b \leftarrow V_D(\mathsf{st}, x, b_1, \ldots, b_q)$ where $b_i = b'_i - b'_0$.

The linear PCP $(\mathbf{P}', (V'_Q, V'_D))$ is smooth: $\mathbf{a}'_0$ is uniform over $\mathbb{F}^\ell$, and so $\mathbf{a}'_i = \mathbf{a}_i + \mathbf{a}'_0$ is 1-wise uniform over $\mathbb{F}_p^\ell$. Observe that for every affine prover strategy $\pi \in \mathbb{F}_p^\ell$ and $c_0, \ldots, c_q \in \mathbb{F}_p$:

$$b_i = b'_i - b'_0 = \langle \pi, \mathbf{a}_i + \mathbf{a}'_0 \rangle + c_i - \langle \pi, \mathbf{a}'_0 \rangle - c_0 = \langle \pi, \mathbf{a}_i \rangle + c_i - c_0 \ .$$

Hence we can translate any prover strategy $\pi, c_0, \ldots, c_q$ in the new protocol to a prover strategy $\pi, (c_1 - c_0), \ldots, (c_q - c_0)$ in the original proof. Moreover, any set of queries $(\mathbf{a}'_0, \ldots, \mathbf{a}'_q)$ in the new scheme can be translated into $(\mathbf{a}_1, \ldots, \mathbf{a}_q)$, where the distribution of a random set of these queries is the identical to the original protocol. Completeness and knowledge soundness follow immediately from this fact.

To see that the smooth LMIP is $2B$-bounded, notice that

$$|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_i) \rangle| = |\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i + \mathbf{a}'_0) \rangle| \leq |\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i) \rangle| + |\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_0) \rangle| \ ,$$

Since the PCP is bounded, $|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i) \rangle| > B$ with probability at most α. By Hoeffding's inequality, since $\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_0) \rangle \in [-\ell \cdot p^2, \ell \cdot p^2]$ for every $t > 0$ we have

$$\Pr[|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_0)) \rangle| > t] < 2 \cdot \exp\left(-\frac{2 \cdot t^2}{\ell^2 p^4} \right)$$

Setting $t = p^2\sqrt{\ell\lambda}$, we get that we have that $|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_0)) \rangle| > p^2\sqrt{\ell\lambda}$ with probability at most $2^{-2\lambda+1} < 2^{-\lambda}$. Thus, $|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_i) \rangle| > B + p^2\sqrt{\ell\lambda}$ with probability at most $\alpha + 2^{-\lambda}$. □

Given Claim 4.5.2, we can safely assume that our linear PCP is smooth with 3 queries. We use this to construct a 3-query strong LMIP. Let $(\mathbf{P}, (V_Q, V_D))$ be the smooth LPCP. We design a 3-query strong LMIP $(\mathbf{P}', (V'_Q, V'_D))$:

- $\mathbf{P}'(x, w)$: Output $\pi \leftarrow \mathbf{P}(x, w)$.
- $V'_Q(x)$: Sample $b \leftarrow \{0, 1\}$ from the Bernoulli distribution where 0 is sampled with probability $\beta = \frac{162}{505 - 729 \cdot \kappa}$. Then, do the following:
  1. If $b = 0$: Sample $(\mathsf{st}, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \leftarrow V_D(x)$. Output $(\mathsf{st}', \mathbf{a}'_1, \mathbf{a}'_2, \mathbf{a}'_3)$ where $\mathsf{st}' = (b, \mathsf{st})$ and $\mathbf{a}'_i = \mathbf{a}_i$.
  2. If $b = 1$: Sample $\mathbf{z}_1, \mathbf{z}_2 \xleftarrow{\$} \mathbb{F}^\ell$ and set $\mathbf{z}_3 = \mathbf{z}_1 + \mathbf{z}_2$. Output $(\mathsf{st}', \mathbf{a}'_1, \mathbf{a}'_2, \mathbf{a}'_3)$ where $\mathsf{st}' = (b, \perp)$ and $\mathbf{a}'_i = \mathbf{z}_i$.
- $V'_D(\mathsf{st}', x, b'_1, b'_2, b'_3)$: Parse $\mathsf{st}' = (b, \mathsf{st})$, and
  1. If $b = 0$: Output 1 if and only if $V_D(\mathsf{st}, x, b'_1, b'_2, b'_3) = 1$.
  2. If $b = 1$: Output 1 if and only if $b'_1 + b'_2 = b'_3$.

Completeness holds by perfect completeness of the smooth linear PCP in the case of $b = 0$, and by linearity of the honest prover strategy in the case of $b = 1$. We now prove knowledge soundness. Let Ext be the extractor for the LPCP. We construct an extractor Ext' as follows:

1. On input x and given oracle access to functions $f_1, f_2, f_3 \colon \mathbb{F}_p^\ell \to \mathbb{F}_p$.
2. For every $i \in [3]$ extract from f_i an affine function π_i, c_i by doing standard local correction of affine functions for distance at most $2/9$ (note this can be done in time $\mathsf{poly}(\ell)$ with constant probability).
3. If $\pi_i \neq \pi_j$ for some i, j, then repeat the above.

4. Otherwise, output $w \leftarrow \mathsf{Ext}(x, \pi_1, c_1, c_2, c_3)$.

Fix x and functions $f_1, \ldots, f_3$ so that $\mathbf{V}'$ accepts x when given access to f_1, f_2, f_3 with probability $\kappa' > \frac{7}{9} + \frac{36}{505-729\cdot\kappa}$. It is clear that if f_1, f_2, f_3 are 2/9-close to affine shifts (c_1, c_2, c_3) of the same linear function π, where $(\pi, (c_1, c_2, c_3))$ causes $\mathbf{V}$ to accept on x with probability greater than κ, then Ext' runs in expected polynomial time and outputs w so that $(x, w) \in \mathcal{R}$. We therefore show that this holds.

Observe that:

$$\begin{aligned}
&\Pr\left[V'_{D}\left(\mathsf{st}', x, b'_1, b'_2, b'_3\right) = 1 \;\middle|\; \begin{array}{r}(\mathsf{st}', \mathbf{a}'_1, \mathbf{a}'_2, \mathbf{a}'_3) \leftarrow V'_{Q}(x) \\ b'_i = f_i(\mathbf{a}'_i)\end{array}\right] \\
&= \beta \cdot \Pr\left[V_{D}\left(\mathsf{st}, x, b'_1, b'_2, b'_3\right) = 1 \;\middle|\; \begin{array}{r}(\mathsf{st}, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \leftarrow V_{Q}(x) \\ b'_i = f_i(\mathbf{a}_i)\end{array}\right] \\
&+ (1-\beta) \cdot \Pr\left[b'_1 + b'_2 = b'_3 \;\middle|\; \begin{array}{r}\mathbf{z}_1, \mathbf{z}_2 \xleftarrow{\$} \mathbb{F}_p^{\ell} \\ \mathbf{z}_3 = \mathbf{z}_1 + \mathbf{z}_3 \\ b'_i = f_i(\mathbf{z}_i)\end{array}\right] .
\end{aligned}$$

We first show that due to the linearity check, there must be linear-consistent functions that are 2/9-close to f_1, f_2, f_3. Note that linear-consistent functions is simply a specific case of a linear function with affine shifts.

Claim 4.5.3. There exist linear-consistent functions $g_1, g_2, g_3 \colon \mathbb{F}^{\ell} \to \mathbb{F}_p$ so that for every $i \in [3]$, $\delta_i = \Delta(f_i, g_i) < 2/9$.

Proof. Suppose that there are no linear-consistent functions $g_1, g_2, g_3 \colon \mathbb{F}_p^{\ell} \to \mathbb{F}_p$ so that for every $i \in [3]$, $\delta_i = \Delta(f_i, g_i) < 2/9$. In this case, by Theorem 2.3.9,

$$\Pr\left[b'_1 + b'_2 = b'_3 \;\middle|\; \begin{array}{r}\mathbf{z}_1, \mathbf{z}_2 \xleftarrow{\$} \mathbb{F}^{\ell} \\ \mathbf{z}_3 = \mathbf{z}_1 + \mathbf{z}_3 \\ b'_i = f_i(\mathbf{z}_i)\end{array}\right] \leq 7/9 .$$

Therefore,

$$\Pr\left[V'_{D}\left(\mathsf{st}', x, b'_1, b'_2, b'_3\right) = 1 \;\middle|\; \begin{array}{r}(\mathsf{st}', \mathbf{a}'_1, \mathbf{a}'_2, \mathbf{a}'_3) \leftarrow V'_{Q}(x) \\ b'_i = f_i(\mathbf{a}'_i)\end{array}\right] \leq \beta + \frac{7}{9} \cdot (1-\beta) < \kappa' ,$$

which contradicts our assumption that $\mathbf{V}'$ accepts x when given access to f_1, f_2, f_3 with probability κ'. □

Let $\pi \in \mathbb{F}_p^{\ell}$, and $c_1, c_2, c_3 \in \mathbb{F}$ be so that $g_i(\mathbf{a}) = \langle \pi, \mathbf{a} \rangle + c_i$ and $c_1 + c_2 = c_3$. We now show that these linear-consistent functions describe shifts of a linear function that cause $\mathbf{V}$ to accept with probability greater than κ:

Claim 4.5.4. $\Pr\left[V_{D}\left(\mathsf{st}, x, b_1, \ldots, b_q\right) = 1 \;\middle|\; \begin{array}{c}(\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_{Q}(x) \\ \forall i \in [q],\ b_i = \langle \pi, \mathbf{a}_i \rangle + c_i\end{array}\right] > \kappa$.

Proof. Suppose towards contradiction that the claim did not hold. Since each query $\mathbf{a}_i$ is smooth, $b_i = f_i(\mathbf{a}_i) = \langle \pi, \mathbf{a}_i \rangle + c_i$ with probability at least $1 - \delta_i \geq 7/9$. Thus,

$$\begin{aligned}
&\Pr\left[V_D\left(\mathsf{st}, x, b'_1, b'_2, b'_3\right) = 1 \;\middle|\; \begin{array}{r} (\mathsf{st}, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \leftarrow V_Q(x) \\ b_i = f_i(\mathbf{a}_i) \end{array} \right] \\
\leq & 1 - \prod_{i=1}^{3}(1-\delta_i) \\
& + \Pr\left[V_D\left(\mathsf{st}, x, b'_1, b'_2, b'_3\right) = 1 \;\middle|\; \begin{array}{r} (\mathsf{st}, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \leftarrow V_Q(x) \\ b_i = \langle \pi, \mathbf{a}_i \rangle + c_i \end{array} \right] \cdot \prod_{i=1}^{3}(1-\delta_i) \\
\leq & 1 - \prod_{i=1}^{3}(1-\delta_i) + \kappa \cdot \prod_{i=1}^{3}(1-\delta_i) \\
\leq & 1 - \left(\frac{7}{9}\right)^3 + \kappa \ .
\end{aligned}$$

Thus, the probability that the verifier accepts is at most

$$\begin{aligned}
&\Pr\left[V'_D\left(\mathsf{st}', x, b'_1, b'_2, b'_3\right) = 1 \;\middle|\; \begin{array}{r} (\mathsf{st}', \mathbf{a}'_1, \mathbf{a}'_2, \mathbf{a}'_3) \leftarrow V'_Q(x) \\ b'_i = \mathbf{P}'_i(\mathbf{a}'_i) \end{array} \right] \\
& \leq \beta \cdot \left(1 - \left(\frac{7}{9}\right)^3 + \kappa\right) + 1 - \beta < \kappa' \ .
\end{aligned}$$

□

Claim 4.5.5. The strong LMIP is $(B + 4p^2\sqrt{\ell\lambda})$-bounded with error $\alpha + 2^{-\lambda}$.

Proof. We consider each choice of $b \in \{0, 1\}$.

- If $b = 0$, then the smooth linear LPCP is $(B + p^2\sqrt{\ell\lambda})$-bounded with error $\alpha + 2^{-\lambda}$.
- If $b = 1$, then two of the queries are uniformly random over $\mathbb{F}$ and the last is the sum of the two. That is, for the first and second queries, $\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i) \rangle$ is uniform over $[-\ell \cdot p^2, \ell \cdot p^2]$. By the Hoeffding inequality, for every $t > 0$ and $i \in \{1, 2\}$,

  $$\Pr\left[|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i) \rangle| \geq t \right] \leq 2\exp\left(-\frac{t^2}{2\ell \cdot p^2}\right)$$

  For the final query, we have

  $$|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_3) \rangle| = |\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_1 + \mathbf{a}_2) \rangle| \leq |\langle \mathbb{Z}(\pi), \mathbf{a}_1 \rangle + \langle \mathbb{Z}(\pi), \mathbf{a}_2 \rangle| \ ,$$

  and so when each of the queries above is within t, the final query is within bound $2t$. Taking a union bound, with probability $1 - 4\exp\left(-\frac{t^2}{2\ell \cdot p^2}\right)$, all of the queries are within bound $2t$.

  By choosing $t = 4p \cdot \sqrt{\ell\lambda}$ we get that all of the queries are within a range of $[-4p\sqrt{\ell\lambda}, 4p\sqrt{\ell\lambda}]$ with probability at least $1 - 2^{-\lambda}$.

Putting both of these together, we have that all of the queries are within bound $B + 4p^2\sqrt{\ell\lambda}$ with probability at least $1 - \alpha - 2^{-\lambda}$. □

□

4.5.2 Modded LPCPs

We define modded linear PCPs, and show that any standard linear PCP can be transformed into a modded one. Modded LPCPs are linear PCPs where each query needs to be within a certain bound (over a large field) and, if it is within this bound, then the verifier receives the query answer after being modded.

Definition 4.5.6. A (bounded) modded linear PCP (mod-LPCP) $(\mathbf{P}, (V_Q, V_D))$ for a relation $\mathcal{R} = \{(x, w)\}$ S is defined as follows:

- *Syntax.* We describe a mod-LPCP with input length n, proof length ℓ, a big field size p', query complexity q, small field sizes $p_1, \ldots, p_q \in \mathbb{N}$, and a bound $B', B \in \mathbb{N}$ with $B' > B$:
  - The verifier query algorithm V_Q receives as input $x \in \mathbb{F}^n$. It outputs a state $\mathsf{st} \in \{0,1\}^*$, and q queries $\mathbf{a}_1, \ldots, \mathbf{a}_q \in \mathbb{F}_{p'}^{\ell}$.
  - The (honest) prover algorithm $\mathbf{P}$ receives an input $x \in \mathbb{F}^n$ and witness $w \in \mathbb{F}^h$. It outputs a proof $\pi \in \mathbb{F}_{p'}^{\ell}$.
  - The verifier decision algorithm V_D receives as input a state $\mathsf{st} \in \{0,1\}^*$, an input $x \in \mathbb{F}^n$, and query answers $b_1 \in \mathbb{F}_{p_1}, \ldots, b_q \in \mathbb{F}_{p_q}$. It outputs a bit $b \in \{0,1\}$.
- *Perfect completeness.* A mod-LPCP has perfect completeness if for all $(x, w) \in \mathcal{R}$:

$$\Pr\left[\begin{array}{c|r} \forall i \in [q], d_i \in [-B, B] & \pi \leftarrow \mathbf{P}(x, w) \\ V_D(\mathsf{st}, x, b_1, \ldots, b_q) = 1 & (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ & \forall i \in [q],\ d_i = \langle \pi, \mathbf{a}_i \rangle \\ & \forall i \in [q],\ b_i = \mathbb{Z}_{p_i}(d_i) \end{array}\right] = 1 \ .$$

- *Soundness.* A mod-LPCP has soundness error δ if for every $x \notin \mathcal{L}$, $\pi \in \mathbb{F}_{p'}^{\ell}$, and $c_1 \in \mathbb{F}_{p_1}, \ldots, c_q \in \mathbb{F}_{p_q}$:

$$\Pr\left[\begin{array}{c|r} \forall i \in [q], d_i \in [-B', B'] & (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ V_D(\mathsf{st}, x, b_1, \ldots, b_q) = 1 & \forall i \in [q],\ d_i = \langle \pi, \mathbf{a}_i \rangle \\ & \forall i \in [q],\ b_i = \mathbb{Z}_{p_i}(d_i) + c_i \end{array}\right] \leq \delta \ .$$

- *Knowledge.* A mod-LPCP satisfies knowledge soundness κ if there exists a PPT extractor Ext such that for every instance x, proof $\pi \in \mathbb{F}_{p'}^{\ell}$, and shifts $c_1 \in \mathbb{F}_{p_1}, \ldots, c_q \in \mathbb{F}_{p_q}$ if,

$$\Pr\left[\begin{array}{c|r} \forall i \in [q], d_i \in [-B', B'] & (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ V_D(\mathsf{st}, x, b_1, \ldots, b_q) = 1 & \forall i \in [q],\ d_i = \langle \pi, \mathbf{a}_i \rangle \\ & \forall i \in [q],\ b_i = \mathbb{Z}_{p_i}(d_i) + c_i \end{array}\right] > \kappa \ ,$$

  then $(x, \mathsf{Ext}(x, \pi, c_1, \ldots, c_q)) \in \mathcal{R}$.

We say that a mod-LPCP is **instance-independent** if $V_Q(x)$ is a function only of $|x|$, in which case we specify its input by $1^{|x|}$ (i.e., the verifier query algorithm is $V_Q(1^{|x|})$).

We show that linear PCPs can be transformed into mod-LPCP:

Lemma 4.5.7. *Let $\mathcal{R}$ be a relation with a LPCP over $\mathbb{F}_p$ with soundness error δ query complexity q and proof length ℓ, and let $p' > p$ be a parameter so that $\frac{p'}{2 \cdot p \cdot \ell} > 2B'$. Then $\mathcal{R}$ has a mod-LPCP over moduli p' and $\mathbf{p}$ with soundness error δ, $q + \lceil -\log \delta \rceil$ queries, and proof length ℓ. The mod-LPCP moduli are $p_i = p$ for $i \in [q]$ and $p_i = 1$ for $i \in [q+1, q+\lceil -\log \delta \rceil]$.*

If the LPCP is instance-oblivious, then so is the mod-LPCP. If the LPCP is B-bounded with error α, then the mod-LPCP is $\max\{B, 2\sqrt{p\ell\lambda}\}$-bounded with error $\alpha + 2^{-\lambda}\lceil \log 1/\delta \rceil$.

Remark 4.5.8. Notice that in the resultant mod-LPCP of Lemma 4.5.7, the last $\lceil -\log \delta \rceil$ queries have a modulus of 1. Therefore, V_D always receives 0 at those positions. They help only with soundness because of the bounds check but do not communicate any information to the verifier.

Proof. Let $q' = q + \lceil -\log \delta \rceil$ and let $(\mathbf{P}, (V_Q, V_D))$ be the original LPCP. We define a mod-LPCP $(\mathbf{P}', (V'_Q, V'_D))$ im the following construction:

- $\mathbf{P}'(x, w)$:
  1. Let $\pi \leftarrow \mathbf{P}(x, w) \in \mathbb{F}_p^\ell$.
  2. Output $\pi' = \mathbb{Z}_{p'}(\pi) \in \mathbb{F}_{p'}^\ell$.
- $V'_Q(x)$:
  1. Let $(\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x)$.
  2. We define a distribution $\mathcal{D}$ whose image is $\{-1, 0, 1\}$ which outputs:
     - -1 with probability $1/4$;
     - 1 with probability $1/4$;
     - 0 with probability $1/2$.
  3. For $i \in [\lceil -\log \delta \rceil]$ sample $\mathbf{a}_{q+i} \leftarrow \mathcal{D}^\ell$.
  4. Output queries $(\mathbf{a}'_i)_{i \in [q']} = (\mathbb{Z}_{p'}(\mathbf{a}_i))_{i \in [q']}$.
- $V'_D(\mathsf{st}, x, b_1, \ldots, b_{q'})$:
  1. Output $b \leftarrow V_D(\mathsf{st}, x, b_1, \ldots, b_q)$. (Note that the verifier ignores b_i for $i > q$.)

Completeness follows by the construction, we prove soundness and later explain knowledge soundness. To prove soundness we distinguish two cases:

- If for every $j \in [\ell]$, we have $|\pi[j]| < \frac{p'}{2 \cdot p \cdot \ell}$. In this case, for all $i \in [q]$, the magnitude of $\langle \pi, \mathbf{a}_i \rangle$ when computations are done over the integers (rather than p') is small:

$$\begin{aligned} |\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_i) \rangle| &= \left| \sum_{j \in [\ell]} \mathbb{Z}(\pi[j]) \cdot \mathbb{Z}(\mathbf{a}_i[j]) \right| \leq \sum_{j \in [\ell]} |\mathbb{Z}(\pi[j])| \cdot |\mathbb{Z}(\mathbf{a}_i[j])| \\ &< \sum_{j \in [\ell]} \frac{p'}{2 \cdot p \cdot \ell} \cdot p = p'/2 \ . \end{aligned}$$

  The above holds by triangle inequality. Since over the integers, the inner product is smaller than p', there is no wrap-around when computing over the integers versus over $\mathbb{Z}_{p'}$. Thus,

$$\begin{aligned} \mathbb{Z}_p(\langle \pi, \mathbf{a}'_i \rangle) &= \mathbb{Z}_p(\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_i) \rangle) = \mathbb{Z}_p \left(\sum_{j \in [\ell]} \mathbb{Z}(\pi[j]) \cdot \mathbb{Z}(\mathbf{a}'_i[j]) \right) \\ &= \sum_{j \in [\ell]} \mathbb{Z}_p(\pi[j]) \cdot \mathbb{Z}_p(\mathbf{a}'_i[j]) = \langle \mathbb{Z}_p(\pi), \mathbb{Z}_p(\mathbf{a}'_i) \rangle = \langle \mathbb{Z}_p(\pi), \mathbf{a}_i \rangle \end{aligned}$$

  Therefore,

$$\begin{aligned} &V'_D(\mathsf{st}, x, \mathbb{Z}_p(\langle \pi, \mathbf{a}'_1 \rangle) + c_1, \ldots, \mathbb{Z}_p(\langle \pi, \mathbf{a}'_{q'} \rangle) + c_{q'}) \\ &= V_D(\mathsf{st}, x, \langle \mathbb{Z}_p(\pi), \mathbf{a}_1 \rangle + c_1, \ldots, \langle \mathbb{Z}_p(\pi), \mathbf{a}_q \rangle + c_q) \ . \end{aligned}$$

Thus, this case reduces to LPCP soundness of $(\mathbf{P}, (V_Q, V_D))$ against the linear proof $(\mathbb{Z}_p(\pi), c_1, \ldots, c_q)$.

- If there exists $j \in [\ell]$ such that $|\pi[j]| \geq \frac{p'}{2 \cdot p \cdot \ell}$. Fix this j for the remainder of the proof. For the rest of the analysis we relax the malicious prover's winning condition: the prover wins if for every $i \in [q+1, q']$ it holds that $\langle \pi, \mathbf{a}_i \rangle \in [-B', B']$. Since we have removed restrictions from the probability that we need to bound, this can only increase the probability. Thus by bounding this probability, we bound the probability that the results are within range and the verifier accepts.

  Fix an arbitrary $i \in [q+1, q']$. We analyze the probability that $\langle \pi, \mathbf{a}_i \rangle \in [-B', B']$. By opening up the definition,

  $$\langle \pi, \mathbf{a}_i \rangle = \pi[j] \cdot \mathbf{a}_i[j] + \sum_{j' \in [\ell] \setminus \{j\}} \pi[j'] \cdot \mathbf{a}_i[j'].$$

  We show that for every $s \in \mathbb{F}_{p'}$ we show that

  $$\Pr_{\mathbf{a}_i[j] \leftarrow \mathcal{D}}[(\pi[j] \cdot \mathbf{a}_i[j] + s_i) \in [-B', B']] \leq 1/2 \ .$$

  Fix $s \in \mathbb{F}_{p'}$ and recall that $|\pi[j]| > \frac{p'}{2 \cdot p \cdot \ell} > 2B'$ and that $\mathbf{a}_i[j]$ is chosen from $\mathcal{D}$. Then,

  - If $|s| \leq B'$ then $|\pi[j] \cdot 1 + s| > B'$ and $|\pi[j] \cdot -1 + s| > B'$. Since $\mathbf{a}_i[j] \in \{-1, 1\}$ with probability $1/2$, we have that $|\pi[j] \cdot \mathbf{a}_i[j] + s| > B'$ with probability $1/2$.
  - If $|s| > B'$, then $|\pi[j] \cdot 0 + s| > B'$. Since $\mathbf{a}_i[j] = 0$ with probability $1/2$, we have that $|\pi[j] \cdot \mathbf{a}_i[j] + s| > B'$ with probability $1/2$.

  Thus, we have that

  $$\begin{aligned} &\Pr_{\mathbf{a}_i \leftarrow \mathcal{D}^\ell}[\langle \pi, \mathbf{a}_i \rangle \in [-B', B']] \\ &= \Pr\left[(\pi[j] \cdot \mathbf{a}_i[j] + s) \in [-B', B'] \;\middle|\; \begin{array}{r} \forall j' \in [\ell] \setminus \{j\} \mathbf{a}_i[j'] \leftarrow \mathcal{D} \\ s = \sum_{j' \in [\ell] \setminus \{j\}} \pi[j'] \cdot \mathbf{a}_i[j'] \\ \mathbf{a}_i[j] \leftarrow \mathcal{D} \end{array} \right] \leq 1/2 \end{aligned}$$

  Finally, since the queries $\mathbf{a}_{q+1}, \ldots, \mathbf{a}_{q'}$ are all chosen independently,

  $$\begin{aligned} \Pr[\forall i \in [q'], \langle \pi, \mathbf{a}_i \rangle \in [-B', B']] &\leq \Pr[\forall i \in [q+1, q'], \langle \pi, \mathbf{a}_i \rangle \in [-B', B']] \\ &\leq 2^{-\lceil -\log \delta \rceil} \\ &\leq \delta. \end{aligned}$$

Remark 4.5.9. If $(\mathbf{P}, (V_Q, V_D))$ has knowledge soundness β then $(\mathbf{P}', (V_Q', V_D'))$ also has knowledge soundness β. This follows by the exact same argument as above using the same extractor as the original LPCP.

Claim 4.5.10. The mod-PCP is $\max\{B, 2\sqrt{p\ell\lambda}\}$-bounded with an error of $\alpha + 2^{-\lambda}\lceil \log 1/\delta \rceil$.

Proof. The first q queries are B-bounded with error α by assumption. Each for $\mathbf{a}_i$, each entry $\mathbf{a}_i[j]$ is bounded within $[-1, 1]$ and has expectation 0. Thus, $|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i) \rangle| \leq p \cdot \ell$ and has an expectation of 0. By Hoeffding's inequality,

$$\Pr[|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i) \rangle| > t] < 2 \cdot \exp\left(-\frac{t^2}{2p^2\ell^2} \right) \ .$$

Since there are $\lceil \log 1/\delta \rceil$ such queries, the probability that there exists a value that is out-of-bound t is at most $\lceil \log 1/\delta \rceil \cdot 2 \cdot \exp\left(-\frac{t^2}{2p^2\ell^2}\right)$. Setting $t = 2\sqrt{p\ell\lambda}$, we get that all of these are within bounds with probability at least $1 - 2^{-\lambda}\lceil \log 1/\delta \rceil$.

Thus, all together, we get the bound $\max\{B, 2\sqrt{p\ell\lambda}\}$ with an error of $\alpha + 2^{-\lambda}\lceil \log 1/\delta \rceil$. □

□

4.6 Dv-SNARGs from Compressible Encryption

In this section, we show how to construct SNARGs (in fact, SNARKs) by combining compressible encryption schemes with suitable information-theoretic protocols. In Section 4.6.1, we show this from isolated homomorphism (Definition 4.4.1) and strong MIPs, and in Section 4.6.2, we show this from bound-limited homomorphism (Definition 4.4.2) and modded LPCPs.

We state here the dv-SNARKs resultant as corollaries from our proofs and transformations.

- Derived by combining Lemma 4.6.3 with the packed ElGamal encryption scheme and a strong linear MIP and making enough repetitions:

  Corollary 4.6.1 (Dv-SNARKs from packed ElGamal)**.** *Let $C\colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \mathbb{F}_p$ be an arithmetic circuit of size s. For every $\lambda, \tau \in \mathbb{N}$ there are dv-SNARKs for $\mathcal{R}_C = \left\{(x, w) \in \mathbb{F}^n \times \mathbb{F}^h \mid C(x, w) = 1\right\}$ in the GGM with of group size 2^λ with completeness error $\mathsf{negl}(\lambda)$, and the following parameters:*

  (Linear CRS.) *Using the LMIP of Corollary 2.3.6:*

  - *Message length: $1\mathbb{G}$ element and $O(\tau)$ bits;*
  - *Knowledge soundness: $2^{-\tau} + 8t^2/2^\lambda$ against t-query adversaries;*
  - *CRS length: $O(\tau \cdot (s + \mathsf{poly}(p)))$ $\mathbb{G}$ elements;*
  - *Setup time: $O(\tau \cdot (s + \mathsf{poly}(p)))$;*
  - *Prover expected runtime: $\tilde{O}(\lambda s \tau^2 \mathsf{poly}(p))$;*
  - *Verifier runtime: $\tilde{O}(\lambda s \tau^2 p^2)$.*

  (Concrete efficiency.) *Using the LMIP derived by combining the LPCP from Theorem 2.3.4 and the transform from Lemma 4.5.1:*

  - *Message length: $1\mathbb{G}$ element and $\lceil 3\tau \log_2 p \rceil$ bits;*
  - *Knowledge soundness: $(\frac{7}{9} + \frac{36p}{505p-1458})^\tau + 8t^2/2^\lambda$ against t-query adversaries;*
  - *CRS length: $(3\tau + 1) \cdot (s^2 + s)$ $\mathbb{G}$ elements;*
  - *Setup time: $O(\tau s^2)$;*
  - *Prover expected runtime: $\tilde{O}(\lambda s^2 \tau^2 p^2)$;*
  - *Verifier runtime: $\tilde{O}(\lambda s^2 \tau^2 p^2)$ ($\tilde{O}(\lambda s \tau^2 p^2)$ if we only require to Boolean circuits).*

  Above, the knowledge soundness errors hold for $t > 4 \cdot t_{\mathbf{V}} + 1$ where $t_{\mathbf{V}}$ is the verification time.

- Derived by combining Lemma 4.6.5 with the packed ElGamal with hash scheme and an LPCP transformed into a modded LPCP using Lemma 4.5.7 we get dv-SNARKs where the number of added bits approaches 2τ:

Corollary 4.6.2 (Dv-SNARKs from packed ElGamal with hash)**.** *Let $C\colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \mathbb{F}_p$ be an arithmetic circuit of size s with $p > 2$. For every $\lambda, \tau \in \mathbb{N}$ there are dv-SNARKs for the relation*

$$\mathcal{R}_C = \left\{(x, w) \in \mathbb{F}^n \times \mathbb{F}^h \mid C(x, w) = 1\right\}$$

in the GGM with group size 2^λ and with random oracle output length λ with completeness error $\mathsf{negl}(\lambda)$ and the following parameters:

(Linear CRS.) *Using the LPCP of Theorem 2.3.5:*

- *Message length: $1\mathbb{G}$ group element, 1 ROM output (λ bits), and $\lceil \tau \log p \rceil$ bits;*
- *Knowledge soundness: $O(p^{-\tau/2}) + 10t^2/2^\lambda$ against t-query adversaries (to both the ROM and GGM);*
- *CRS length: $O(\tau \cdot (s + \mathsf{poly}(p)))$ $\mathbb{G}$ elements;*
- *Setup time: $O(\tau \cdot (s + \mathsf{poly}(p)))$;*
- *Prover expected runtime: $\tilde{O}(\lambda s \tau^2 \mathsf{poly}(p))$;*
- *Verifier runtime: $\tilde{O}(\lambda s \tau^2 p^2)$.*

(Minimal length.) *Using the LPCP of Theorem 2.3.4:*

- *Message length: $1\mathbb{G}$ group element, 1 ROM output (λ bits), and $\lceil 2\tau \log p \rceil$ bits;*
- *Knowledge soundness: $(2/p)^{-\tau} + 10t^2/2^\lambda$ against t-query adversaries (to both the ROM and GGM);*
- *CRS length: $O(\tau s^2)$ $\mathbb{G}$ elements;*
- *Setup time: $O(\tau s^2)$;*
- *Prover expected runtime: $\tilde{O}(\lambda s^2 \tau^2 p^2)$;*
- *Verifier runtime: $\tilde{O}(\lambda s^2 \tau^2 p^2)$ (If we restrict ourselves to Boolean circuits it becomes$\tilde{O}(\lambda s \tau^2 p^2)$).*

Above, the knowledge soundness errors hold for $t > 4 \cdot t_{\mathbf{V}} + 1$ where $t_{\mathbf{V}}$ is the verification time.

4.6.1 Construction from Isolated Homomorphism

We show how to combine a strong LMIP with knowledge soundness, and an isolated homomorphic encryption scheme into a dv-SNARK:

Lemma 4.6.3. *Suppose the existence of the following ingredients:*

- *An input-oblivious strong linear MIP over finite field $\mathbb{F}_p$ for a relation $\mathcal{R}$ that is B-bounded with error α, soundness δ, knowledge soundness κ, q queries, query length ℓ, prover running time t_P, and verifier running time (t_Q, t_D).*
- *A compressible linearly-homomorphic encryption with q slots, plaintext moduli p, ciphertext size σ_{ct}, compressed ciphertext size σ_{cct}, decryption bound B, and encryption, compression, evaluation, and decryption times $(t_{\mathsf{enc}}, t_{\mathsf{cmp}}, t_{\mathsf{eval}}, t_{\mathsf{dec}})$. Furthermore, let $\varepsilon_{\mathsf{cor}}$ be its correctness error, $\varepsilon_{\mathsf{ih}}$ be its isolated homomorphism distinguishability error and $\varepsilon_{\mathsf{sem}}$ be its semantic security advantage.*

Then there is a designated-verifier SNARK for $\mathcal{R}$ with:

- *Completeness error: $\alpha + \ell \cdot \varepsilon_{\mathsf{cor}}(\lambda)$;*

- *Soundness:* $\delta + \varepsilon_{\mathsf{ih}}(\lambda, t, \ell) + \varepsilon_{\mathsf{sem}}(\lambda, t, \ell)$ *against* t*-query adversaries;*
- *Knowledge soundness:* $\kappa + \varepsilon_{\mathsf{ih}}(\lambda, t, \ell) + \ell \cdot \varepsilon_{\mathsf{sem}}(\lambda, t)$ *against* t*-query adversaries;*
- *Message length:* $\sigma_{\mathsf{cct}}(\lambda)$*;*
- *CRS length:* $\ell \cdot \sigma_{\mathsf{ct}}(\lambda)$*;*
- *Setup time:* $O(t_Q + \ell \cdot t_{\mathsf{enc}}(\lambda))$*;*
- *Prover runtime:* $O(t_P + t_{\mathsf{eval}}(\lambda, \ell) + t_{\mathsf{cmp}}(\lambda))$*;*
- *Verifier runtime:* $O(t_D + t_{\mathsf{dec}}(\lambda))$*.*

Proof. Let $(\mathbf{P}, (V_Q, V_D))$ be the linear MIP, and $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be the homomorphic encryption scheme. We construct a SNARK

Construction 4.6.4. The SNARK $(\mathsf{Setup}, \mathbf{P}', \mathbf{V}')$ is defined as follows:

- $\mathsf{Setup}^{\mathcal{G}}(1^n)$:
  1. Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$ and $(\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(1^n)$.
  2. For every $i \in [\ell]$ let $\mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]) \in \mathbb{F}^q$ and compute $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{z}_i)$.
  3. Output $\mathsf{crs} := (\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ and $\mathsf{st} := (\mathsf{sk}, \mathsf{st}')$.
- $\mathbf{P}'^{\mathcal{G}}(\mathsf{crs}, x, w)$:
  1. Parse $\mathsf{crs} := (\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ and compute $\pi \leftarrow \mathbf{P}(x, w)$.
  2. Compute $\mathsf{ct}' \leftarrow \mathsf{Eval}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \pi)$.
  3. Let $\mathsf{cct} := \mathsf{Compress}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}')$.
  4. Output $\mathsf{pf} := \mathsf{cct}$.
- $\mathbf{V}'^{\mathcal{G}}(\mathsf{st}, x, \mathsf{pf})$:
  1. Parse $\mathsf{st} = (\mathsf{sk}, \mathsf{st}')$ and $\mathsf{pf} = \mathsf{cct}$.
  2. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \bot$, then output 0. Otherwise, let $(b_1, \ldots, b_q) = \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$.
  3. Output 1 if $V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1$ and otherwise output 0.

Completeness, and the complexity parameters follow immediately from the construction. We prove soundness and knowledge by first proving soundness, and then explaining the (small) differences when proving knowledge.

Soundness. Fix λ and a prover $\mathbf{P}'$ for the dv-SNARK soundness experiment that makes t queries. We show that the probability that $\mathbf{V}'$ outputs 1 is at most $\delta + \varepsilon_{\mathsf{sem}} + \varepsilon_{\mathsf{ih}}$. Suppose towards contradiction to the soundness error of the strong LMIP that $\mathbf{P}'$ causes the verifier to accept with probability $\delta' > \delta + \varepsilon_{\mathsf{sem}} + \varepsilon_{\mathsf{ih}}$.

Let $\mathcal{S}$ be the simulator of the isolated homomorphism experiment of the encryption scheme, and define a plaintext generator $\mathcal{T}$ and adversary $\mathcal{A}$:

- $\mathcal{T}^{\mathcal{G}}(\mathsf{pk})$: Sample $(\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_1) \leftarrow V_Q(x)$. Output $(\mathsf{st}', (\mathbf{z}_1, \ldots, \mathbf{z}_\ell))$ where $\mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i])$.
- $\mathcal{A}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$: Let $\mathsf{crs} = (\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ and compute and output $(x, \mathsf{cct}) \leftarrow \mathbf{P}'^{\mathcal{G}}(\mathsf{crs})$.

By plugging in to the isolated homomorphism experiment this plaintext generator and adversary, the outputs of the following two experiments have statistical distance at most $\varepsilon_{\mathsf{ih}}$:

- **Real**:
  1. Sample $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$.

2. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$.
3. $(\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_\ell) \leftarrow V_Q(1^n)$.
4. $\mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i])$ for all $i \in [\ell]$.
5. $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i)$ for all $i \in [\ell]$.
6. $(x, \mathsf{cct}) \leftarrow \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.
7. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \bot$, output $\bot$.
8. $(b_1, \ldots, b_q) \leftarrow \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$
9. Output $(\mathsf{st}', x, b_1, \ldots, b_q)$.

- **Ideal**:
  1. Sample $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$.
  2. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$.
  3. $(\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_\ell) \leftarrow V_Q(1^n)$.
  4. $\mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i])$ for all $i \in [\ell]$.
  5. $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i)$ for all $i \in [m]$.
  6. $(\mathsf{cct}, x) \leftarrow_{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.
  7. $(f_1, \ldots, f_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct})$.
  8. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \bot$, output $\bot$.
  9. $b_i = f_i(\mathbf{a}_i)$ for all $i \in [q]$.
  10. Output $(\mathsf{st}', x, b_1, \ldots, b_q)$.

Consider the following predicate $p(X)$: if $X = \bot$ or X cannot be parsed as $X = (\mathsf{st}', x, b_1, \ldots, b_q)$, then output 0. Otherwise, output 1 if $|x| = n$, $x \in \mathcal{L}(\mathcal{R})$, and $V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1$, and otherwise output 0. Observe that after applying the predicate to the output of the real experiment, we get the predicate of whether the SNARK verifier accepted in the adaptive soundness experiment, which happens with probability δ'. Thus, by $\varepsilon_{\mathsf{ih}}$ statistical indistinguishability of the real and ideal games, the probability of the predicate being satisfied in the ideal game is at least $\delta' - \varepsilon_{\mathsf{ih}}$:

$$\Pr\left[\begin{array}{c|r} x \notin \mathcal{L}(\mathcal{R}) \\ \wedge\ \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) \neq \bot \\ \wedge\ V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1 & \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ \forall i \in [\ell],\ \mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]) \\ \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i) \\ (\mathsf{cct}, x) \leftarrow_{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ (f_1, \ldots, f_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct}) \\ \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \end{array} \end{array}\right]$$
$$\geq \delta' - \varepsilon_{\mathsf{ih}}(\lambda, t, \ell)\ .$$

We can now remove the check that decryption is done correctly, thus making the predicate independent of the encryption secret key. That is, the expression above is bounded from above by

$$\Pr\left[\begin{array}{c|r} x \notin \mathcal{L}(\mathcal{R}) \\ \wedge\ V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1 & \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ \forall i \in [\ell],\ \mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]) \\ \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i) \\ (\mathsf{cct}, x) \leftarrow_{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ (f_1, \ldots, f_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct}) \\ \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \end{array} \end{array}\right]$$

We now utilize semantic security of the encryption scheme to change the encryptions to 0^q.

$$\Pr\left[\begin{array}{c} x \notin \mathcal{L}(\mathcal{R}) \\ \wedge\, V_{\mathsf{D}}(\mathsf{st}', x, b_1, \ldots, b_q) = 1 \end{array} \middle| \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_{\mathsf{Q}}(1^n) \\ \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, 0^q) \\ (\mathsf{cct}, x) \leftarrow_{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ (f_1, \ldots, f_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct}) \\ \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \end{array}\right]$$

$$\geq \Pr\left[\begin{array}{c} x \notin \mathcal{L}(\mathcal{R}) \\ \wedge\, V_{\mathsf{D}}(\mathsf{st}', x, b_1, \ldots, b_q) = 1 \end{array} \middle| \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_{\mathsf{Q}}(1^n) \\ \forall i \in [\ell],\ \mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]) \\ \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i) \\ (\mathsf{cct}, x) \leftarrow_{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ (f_1, \ldots, f_q) \leftarrow \mathcal{S}(\mathcal{A}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \end{array}\right]$$

$$- \varepsilon_{\mathsf{sem}}(\lambda, t, \ell)\ .$$

Indeed, otherwise we could run the above experiments to distinguish ciphertexts of $(\mathbf{z}_1, \ldots, \mathbf{z}_\ell)$ from ciphertexts of the all-zeroes string. Thus, we have that

$$\Pr\left[\begin{array}{c} x \notin \mathcal{L}(\mathcal{R}) \\ \wedge\, V_{\mathsf{D}}(\mathsf{st}', x, b_1, \ldots, b_q) = 1 \end{array} \middle| \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_{\mathsf{Q}}(1^n) \\ \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, 0^q) \\ (\mathsf{cct}, x) \leftarrow_{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ (f_1, \ldots, f_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct}) \\ \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \end{array}\right]$$

$$\geq \delta' - \varepsilon_{\mathsf{ih}}(\lambda, \ell, t) - \varepsilon_{\mathsf{sem}}(\lambda, t, \ell) > \delta\ .$$

Observe that, now the choice of x, $f_1, \ldots, f_q$ does not depend on $\mathbf{a}_1, \ldots, \mathbf{a}_q$. By an averaging argument, there exist x, $f_1, \ldots, f_q$ so that:

$$\Pr\left[\ V_{\mathsf{D}}(\mathsf{st}', x, b_1, \ldots, b_q) = 1 \ \middle|\ \begin{array}{r} (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_{\mathsf{Q}}(1^n) \\ \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \end{array}\ \right] > \delta\ ,$$

which contradicts soundness of the strong LMIP.

Knowledge soundness. Let Ext be the extractor of the LMIP. We specify our SNARK extractor Ext':

1. On input $\mathsf{crs} = (\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$, x, cct, and a trace tr.
2. Run $(f_1, \ldots, f_q) \leftarrow \mathcal{S}(\mathsf{pk}, \mathsf{cct}, \mathsf{tr})$.
3. Output $w \leftarrow \mathsf{Ext}^{f_1, \ldots, f_q}(x)$.

It is immediate that Ext runs in expected time that is polynomial in λ, n, and t. The proof that our scheme has knowledge soundness $\kappa + \varepsilon_{\mathsf{sem}} + \varepsilon_{\mathsf{ih}}$ closely follows that of standard

soundness. Indeed, the knowledge soundness game is:

$$\Pr\left[\begin{array}{c|r} (x,w)\notin\mathcal{R} & \mathcal{G}\leftarrow\mathsf{GGM}(\lambda) \\ \wedge\ \mathbf{V}^{\mathcal{G}}(\mathsf{st},x,\mathsf{pf})=1 & (\mathsf{crs},\mathsf{st})\leftarrow\mathsf{Setup}^{\mathcal{G}}(1^n) \\ & (x,\mathsf{pf})\leftarrow_{\mathsf{tr}}\mathbf{P}'^{\mathcal{G}}(\mathsf{crs}) \\ & w\leftarrow\mathsf{Ext}(x,\mathsf{pf},\mathsf{tr}) \end{array}\right]$$

$$=\Pr\left[\begin{array}{c|r} (x,w)\notin\mathcal{R} & \mathcal{G}\leftarrow\mathsf{GGM}(\lambda) \\ \wedge\ \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk},\mathsf{cct})\neq\perp & (\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{KeyGen}^{\mathcal{G}} \\ \wedge\ V_D(\mathsf{st}',x,b_1,\ldots,b_q)=1 & (\mathsf{st}',\mathbf{a}_1,\ldots,\mathbf{a}_q)\leftarrow V_Q(1^n) \\ & \forall i\in[\ell],\ \mathbf{z}_i=(\mathbf{a}_1[i],\ldots,\mathbf{a}_q[i]) \\ & \forall i\in[\ell],\ \mathsf{ct}_i\leftarrow\mathsf{Enc}^{\mathcal{G}}(\mathsf{pk},\mathbf{z}_i) \\ & (\mathsf{cct},x)\leftarrow_{\mathsf{tr}}\mathbf{P}'^{\mathcal{G}}(\mathsf{pk},\mathsf{ct}_1,\ldots,\mathsf{ct}_\ell) \\ & (f_1,\ldots,f_q)\leftarrow\mathcal{S}(\mathsf{tr},\mathsf{pk},\mathsf{ct}_1,\ldots,\mathsf{ct}_\ell,\mathsf{cct}) \\ & (b_1,\ldots,b_q)\leftarrow\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk},\mathsf{cct}) \\ & w\leftarrow\mathsf{Ext}^{f_1,\ldots,f_q}(x) \end{array}\right].$$

By then following precisely the same arguments as in soundness, with this experiment we end up at the fact that there exist x and $f_1,\ldots,f_q$ so that,

$$\Pr\left[\begin{array}{c|r} (x,w)\notin\mathcal{R} & (\mathsf{st}',\mathbf{a}_1,\ldots,\mathbf{a}_q)\leftarrow V_Q(1^n) \\ \wedge\ V_D(\mathsf{st}',x,b_1,\ldots,b_q)=1 & \forall i\in[q],\ b_i=f_i(\mathbf{a}_i) \\ & w\leftarrow\mathsf{Ext}^{f_1,\ldots,f_q}(x) \end{array}\right]>\kappa\ ,$$

which contradicts knowledge soundness of the strong LMIP. □

4.6.2 Construction from Bound-Limited Homomorphism

In this section we show how to combine a modded LPCP with knowledge soundness, and a bound-limited homomorphic encryption scheme into a designated-verifier SNARK:

Lemma 4.6.5. *Suppose the existence of the following ingredients:*

- *An input-oblivious mod-LPCP over big modulus p' and moduli (p_i) for a relation $\mathcal{R}$ that is B-bounded with error α, soundness δ, knowledge soundness κ, q queries, query length ℓ, prover running time t_P, and verifier running time (t_Q,t_D). Let B' be its big moduli bound.*
- *A compressible linearly homomorphic encryption scheme over p' with q slots, plaintext moduli (p_i), ciphertext size σ_{ct}, compressed ciphertext size σ_{cct}, decryption bound B, and encryption, compression, evaluation, and decryption times $(t_{\mathsf{enc}},t_{\mathsf{cmp}},t_{\mathsf{eval}},t_{\mathsf{dec}})$. Furthermore, let $\varepsilon_{\mathsf{cor}}$ be its correctness error, $\varepsilon_{\mathsf{mb}}$ be its bound-limited homomorphism error with bound B' error and $\varepsilon_{\mathsf{sem}}$ be its semantic security advantage.*

Then there is a designated-verifier SNARK for $\mathcal{R}$ with:

- *Completeness error: $\ell\cdot\varepsilon_{\mathsf{cor}}(\lambda)$;*
- *Soundness: $\delta+\varepsilon_{\mathsf{mb}}(\lambda,t,\ell)+\varepsilon_{\mathsf{sem}}(\lambda,t,\ell)$ against t-query adversaries;*
- *Knowledge soundness: $\kappa+\varepsilon_{\mathsf{mb}}(\lambda,t,\ell)+\ell\cdot\varepsilon_{\mathsf{sem}}(\lambda,t)$ against t-query adversaries;*
- *Message length: $\sigma_{\mathsf{cct}}(\lambda)$;*
- *CRS length: $\ell\cdot\sigma_{\mathsf{ct}}(\lambda)$;*
- *Setup time: $O(t_Q+\ell\cdot t_{\mathsf{enc}}(\lambda))$;*

- *Prover runtime:* $O(t_P + t_{\mathsf{eval}}(\lambda, \ell))$*;*
- *Verifier runtime:* $O(t_D + t_{\mathsf{dec}}(\lambda))$*.*

Proof. Let $(\mathbf{P}, (V_Q, V_D))$ be the linear MIP, and $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be the homomorphic encryption scheme. We construct a SNARK

Construction 4.6.6. The SNARK $(\mathsf{Setup}, \mathbf{P}', \mathbf{V}')$ is defined as follows:

- $\mathsf{Setup}^{\mathcal{G}}(1^n)$:
  1. Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$ and $(\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(1^n)$.
  2. For every $i \in [\ell]$ let $\mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]) \in \mathbb{F}^q$ and compute $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i)$.
  3. Output $\mathsf{crs} := (\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ and $\mathsf{st} := (\mathsf{sk}, \mathsf{st}')$.
- $\mathbf{P}'^{\mathcal{G}}(\mathsf{crs}, x, w)$:
  1. Parse $\mathsf{crs} := (\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ and compute $\pi \leftarrow \mathbf{P}(x, w)$.
  2. Compute $\mathsf{ct}' \leftarrow \mathsf{Eval}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \pi)$.
  3. Let $\mathsf{cct} := \mathsf{Compress}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}')$.
  4. Output $\mathsf{pf} := \mathsf{cct}$.
- $\mathbf{V}'^{\mathcal{G}}(\mathsf{st}, x, \mathsf{pf})$:
  1. Parse $\mathsf{st} = (\mathsf{sk}, \mathsf{st}')$ and $\mathsf{pf} = \mathsf{cct}$.
  2. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \bot$, then output 0. Otherwise, let $(b_1, \ldots, b_q) = \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$.
  3. Output 1 if $V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1$ and otherwise output 0.

Completeness, and the complexity parameters follow immediately from the construction. We prove soundness and knowledge by first proving soundness, and then explaining the (small) differences when proving knowledge.

Soundness. Fix λ and a prover $\mathbf{P}'$ for the dv-SNARK soundness experiment that makes t queries. We show that the probability that $\mathbf{V}'$ outputs 1 is at most $\delta + \varepsilon_{\mathsf{sem}} + \varepsilon_{\mathsf{mb}}$. Suppose towards contradiction of the soundness error of the mod-LPCP that $\mathbf{P}'$ causes the verifier to accept with probability $\delta' > \delta + \varepsilon_{\mathsf{sem}} + \varepsilon_{\mathsf{mb}}$.

Let $\mathcal{S}$ be the simulator of the bounded-restricted homomorphism experiment of the encryption scheme, and define a plaintext generator $\mathcal{T}$ and adversary $\mathcal{A}$:

- $\mathcal{T}^{\mathcal{G}}(\mathsf{pk})$: Sample $(\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_1) \leftarrow V_Q(x)$. Output $(\mathsf{st}', (\mathbf{z}_1, \ldots, \mathbf{z}_\ell))$ where $\mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i])$.
- $\mathcal{A}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$: Let $\mathsf{crs} = (\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ and compute and output $(x, \mathsf{cct}) \leftarrow \mathbf{P}'^{\mathcal{G}}(\mathsf{crs})$.

By plugging in to the bound-limited homomorphism experiment this plaintext generator and adversary, the outputs of the following two experiments have statistical distance at most $\varepsilon_{\mathsf{mb}}$:

- **Real**:
  1. Sample $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$.
  2. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$.
  3. $(\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_\ell) \leftarrow V_Q(1^n)$.
  4. $\mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i])$ for all $i \in [\ell]$.
  5. $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i)$ for all $i \in [\ell]$.

6. $(x, \mathsf{cct}) \leftarrow \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.
7. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \bot$, output $\bot$.
8. $(b_1, \ldots, b_q) \leftarrow \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$
9. Output $(\mathsf{st}', x, b_1, \ldots, b_q)$.

- **Ideal**:

1. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$.
2. $(\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_\ell) \leftarrow V_Q(1^n)$.
3. $\mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i])$ for all $i \in [\ell]$.
4. $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i)$ for all $i \in [m]$.
5. $(\mathsf{cct}, x) \leftarrow_{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.
6. $(\pi, c_1, \ldots, c_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct})$, where $\pi \in \mathbb{Z}_{p'}^q$ and $c_i \in \mathbb{Z}_{p_i}$.
7. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \bot$, output $\bot$.
8. $b_i \leftarrow \mathbb{Z}_{p_i}\left(\sum_{j\in[q]} \pi_j \mathbf{z}_j[i]\right) + c_i = \mathbb{Z}_{p_i}\left(\langle \pi, \mathbf{a}_i \rangle\right) + c_i$.
9. If there exists an i such that $b_i \notin [-B', B']$ output $\bot$.
10. Output $(\mathsf{st}', x, b_1, \ldots, b_q)$.

Consider the following predicate $p(X)$: if $X = \bot$ or X cannot be parsed as $X = (\mathsf{st}', x, b_1, \ldots, b_q)$, then output 0. Otherwise, output 1 if $|x| = n$, $x \in \mathcal{L}(\mathcal{R})$, and $V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1$, and otherwise output 0. Observe that after applying the predicate to the output of the real experiment, we get the predicate of whether the SNARK verifier accepted in the adaptive soundness experiment, which happens with probability δ'. Thus, by $\varepsilon_{\mathsf{mb}}$ statistical indistinguishability of the real and ideal games, the probability of the predicate being satisfied in the ideal game is at least $\delta' - \varepsilon_{\mathsf{mb}}$:

$$\Pr\left[\begin{array}{c|r} x \notin \mathcal{L}(\mathcal{R}) & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ \wedge\ \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) \neq \bot & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ \wedge\ \forall i \in [q],\ b_i \in [-B', B'] & (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ \wedge\ V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1 & \forall i \in [\ell],\ \mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]) \\ & \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i) \\ & (\mathsf{cct}, x) \leftarrow_{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ & (\pi, c_1, \ldots, c_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct}) \\ & \forall i \in [q],\ b_i \leftarrow \mathbb{Z}_{p_i}\left(\langle \pi, \mathbf{a}_i \rangle\right) + c_i \end{array}\right]$$
$$\geq \delta' - \varepsilon_{\mathsf{mb}}(\lambda, t, \ell)\ .$$

We can now remove the check that the decryption succeeds, thus making the predicate independent of the encryption secret key. That is, the above expression is upper bounded by:

$$\Pr\left[\begin{array}{c|r} x \notin \mathcal{L}(\mathcal{R}) & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ \wedge\ \forall i \in [q],\ b_i \in [-B', B'] & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ \wedge\ V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1 & (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ & \forall i \in [\ell],\ \mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]) \\ & \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i) \\ & (\mathsf{cct}, x) \leftarrow_{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ & (\pi, c_1, \ldots, c_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct}) \\ & \forall i \in [q],\ b_i \leftarrow \mathbb{Z}_{p_i}\left(\langle \pi, \mathbf{a}_i \rangle\right) + c_i \end{array}\right]$$

We now utilize semantic security of the encryption scheme to change the encryptions to 0^q.

$$\Pr\left[\begin{array}{c} x \notin \mathcal{L}(\mathcal{R}) \\ \wedge\, \forall i \in [q],\ b_i \in [-B', B'] \\ \wedge\, V_{\scriptscriptstyle D}(\mathsf{st}', x, b_1, \ldots, b_q) = 1 \end{array} \middle| \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_{\scriptscriptstyle Q}(1^n) \\ \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, 0^q) \\ (\mathsf{cct}, x) \leftarrow_{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ (\pi, c_1, \ldots, c_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct}) \\ \forall i \in [q],\ b_i \leftarrow \mathbb{Z}_{p_i}\left(\langle \pi, \mathbf{a}_i \rangle\right) + c_i \end{array}\right]$$

$$\geq \Pr\left[\begin{array}{c} x \notin \mathcal{L}(\mathcal{R}) \\ \wedge\, \forall i \in [q],\ b_i \in [-B', B'] \\ \wedge\, V_{\scriptscriptstyle D}(\mathsf{st}', x, b_1, \ldots, b_q) = 1 \end{array} \middle| \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_{\scriptscriptstyle Q}(1^n) \\ \forall i \in [\ell],\ \mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]) \\ \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i) \\ (\mathsf{cct}, x) \leftarrow_{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ (\pi, c_1, \ldots, c_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct}) \\ \forall i \in [q],\ b_i \leftarrow \mathbb{Z}_{p_i}\left(\langle \pi, \mathbf{a}_i \rangle\right) + c_i \end{array}\right]$$

$$- \varepsilon_{\mathsf{sem}}(\lambda, t, \ell)\ .$$

Indeed, otherwise we could run the above experiments to distinguish ciphertexts of $(\mathbf{z}_1, \ldots, \mathbf{z}_\ell)$ from ciphertexts of the all-zeroes string. Thus, we have that

$$\Pr\left[\begin{array}{c} x \notin \mathcal{L}(\mathcal{R}) \\ \wedge\, \forall i \in [q],\ b_i \in [-B', B'] \\ \wedge\, V_{\scriptscriptstyle D}(\mathsf{st}', x, b_1, \ldots, b_q) = 1 \end{array} \middle| \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_{\scriptscriptstyle Q}(1^n) \\ \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, 0^q) \\ (\mathsf{cct}, x) \leftarrow_{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ (\pi, c_1, \ldots, c_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct}) \\ \forall i \in [q],\ b_i \leftarrow \mathbb{Z}_{p_i}\left(\langle \pi, \mathbf{a}_i \rangle\right) + c_i \end{array}\right]$$

$$\geq \delta' - \varepsilon_{\mathsf{mb}}(\lambda, \ell, t) - \varepsilon_{\mathsf{sem}}(\lambda, t, \ell) > \delta\ .$$

Observe that, now the choice of x, π, and $c_1, \ldots, c_q$ does not depend on $\mathbf{a}_1, \ldots, \mathbf{a}_q$. By an averaging argument, there exist x, π, and $c_1, \ldots, c_q$ so that:

$$\Pr\left[\begin{array}{c} \forall i \in [q],\ b_i \in [-B', B'] \\ \wedge\, V_{\scriptscriptstyle D}(\mathsf{st}', x, b_1, \ldots, b_q) = 1 \end{array} \middle| \begin{array}{c} (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_{\scriptscriptstyle Q}(1^n) \\ \forall i \in [q],\ b_i \leftarrow \mathbb{Z}_{p_i}\left(\langle \pi, \mathbf{a}_i \rangle\right) + c_i \end{array}\right] > \delta\ ,$$

which contradicts soundness of the LPCP.

Knowledge soundness. Let Ext be the extractor of the LPCP. We specify our SNARK extractor Ext':

1. On input $\mathsf{crs} = (\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$, x, cct, and a trace tr.
2. Run $(\pi, c_1, \ldots, c_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct})$.
3. Output $w \leftarrow \mathsf{Ext}(x, \pi, c_1, \ldots, c_q)$.

It is immediate that Ext runs in expected time that is polynomial in λ, n, and t. The proof that our scheme has knowledge soundness $\kappa + \varepsilon_{\mathsf{sem}} + \varepsilon_{\mathsf{mb}}$ closely follows that of standard

soundness. Indeed, the knowledge soundness game is:

$$\Pr\left[\begin{array}{c|r} (x,w)\notin\mathcal{R} & \mathcal{G}\leftarrow\mathsf{GGM}(\lambda) \\ \wedge\ \mathbf{V}^{\mathcal{G}}(\mathsf{st},x,\mathsf{pf})=1 & (\mathsf{crs},\mathsf{st})\leftarrow\mathsf{Setup}^{\mathcal{G}}(1^n) \\ & (x,\mathsf{pf})\leftarrow_{\mathsf{tr}}\mathbf{P}'^{\mathcal{G}}(\mathsf{crs}) \\ & w\leftarrow\mathsf{Ext}(x,\mathsf{pf},\mathsf{tr}) \end{array}\right]$$

$$=\Pr\left[\begin{array}{c|r} & \mathcal{G}\leftarrow\mathsf{GGM}(\lambda) \\ & (\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{KeyGen}^{\mathcal{G}} \\ & (\mathsf{st}',\mathbf{a}_1,\ldots,\mathbf{a}_q)\leftarrow V_Q(1^n) \\ (x,w)\notin\mathcal{R} & \forall i\in[\ell],\ \mathbf{z}_i=(\mathbf{a}_1[i],\ldots,\mathbf{a}_q[i]) \\ \wedge\ \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk},\mathsf{cct})\neq\perp & \forall i\in[\ell],\ \mathsf{ct}_i\leftarrow\mathsf{Enc}^{\mathcal{G}}(\mathsf{pk},\mathbf{z}_i) \\ \wedge\ V_D(\mathsf{st}',x,b_1,\ldots,b_q)=1 & (\mathsf{cct},x)\leftarrow_{\mathsf{tr}}\mathbf{P}'^{\mathcal{G}}(\mathsf{pk},\mathsf{ct}_1,\ldots,\mathsf{ct}_\ell) \\ & (\pi,(c_i)_{i\in[q]})\leftarrow\mathcal{S}(\mathsf{tr},\mathsf{pk},\mathsf{ct}_1,\ldots,\mathsf{ct}_\ell,\mathsf{cct}) \\ & (b_1,\ldots,b_q)\leftarrow\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk},\mathsf{cct}) \\ & w\leftarrow\mathsf{Ext}(x,\pi,c_1,\ldots,c_q) \end{array}\right].$$

By then following precisely the same arguments as in soundness, with this experiment we end up at the fact that there exist x, π, and $c_1,\ldots,c_q$ so that,

$$\Pr\left[\begin{array}{c|r} (x,w)\notin\mathcal{R} & (\mathsf{st}',\mathbf{a}_1,\ldots,\mathbf{a}_q)\leftarrow V_Q(1^n) \\ \wedge\ \forall i\in[q],\ b_i\in[-B',B'] & \forall i\in[q],\ b_i\leftarrow\mathbb{Z}_{p_i}\left(\langle\pi,\mathbf{a}_i\rangle\right)+c_i \\ \wedge\ V_D(\mathsf{st}',x,b_1,\ldots,b_q)=1 & w\leftarrow\mathsf{Ext}(x,\pi,c_1,\ldots,c_q) \end{array}\right]>\kappa\ ,$$

which contradicts knowledge soundness of the strong LMIP. □

Chapter 5

Incompressible Encryption

5.1 Introduction

Incompressible cryptography [Dzi06b, DGO19, GLW20, MW20, GZ21, GWZ22] is a flourishing paradigm trying to leverage memory limitations of adversaries to achieve strong security goals. While traditionally, the goal of cryptography in the bounded storage model [Mau93] is to minimize the need for computational assumptions or even obtain information-theoretically secure constructions, incompressible cryptography is geared more toward mitigating the consequences of key exfiltration and key exposure attacks. In this work, we focus on the notion of *incompressible encryption* [Dzi06b, GWZ22] [1] recently coined by Guan et al. [GWZ22]. An incompressible encryption scheme produces large, incompressible ciphertexts and guarantees that any adversary who *forgets* even a small fraction of the ciphertext data will learn nothing about the encrypted data, even if he is later given the corresponding secret key!

One motivation for incompressible encryption is to hamper adversaries conducting a mass-surveillance operation by forcing them to store massive amounts of ciphertext data even if they are just interested in a tiny fraction of the encrypted data. In a similar scenario, an adversary trying to exfiltrate information encrypted under an incompressible encryption scheme from a data-center will have to exfiltrate massive amounts of data, even if his exfiltration target is just a small piece of information.

Encryption in the *bounded-retrieval model* [Dzi06a, DLW06] where the goal is to make the secret key large and incompressible (to make it hard to exfiltrate) while keeping all other system parameters small are orthogonal to incompressible encryption.

Encryption with High Rate. An important efficiency measure of encryption schemes is their ciphertext expansion or rate. The rate of an encryption scheme is the ratio between plaintext size and ciphertext size. The closer the rate is to 1, the more efficient a scheme manages to pack information into a ciphertext. Conversely, the closer the rate is to 0, the less information is encoded in potentially large ciphertexts. For incompressible encryption, achieving a high rate (ideally converging to 1), especially if we think of the data center application above, where a small rate would also put a massive burden on the data center.

Guan et al. [GWZ22] provided two constructions of incompressible encryption.

- One from public-key encryption that has ciphertext-rate approaching 0.

[1] Dziembowski [Dzi06b] introduced this concept under the name *forward-secure storage* in the symmetric key setting.

- The other from indistinguishability obfuscation (iO) [BGI+01, GGH+13, JLS21] that achieves ciphertext-rate 1.

We remark that their rate-1 construction relies on non-black-box techniques and iO, which gives this result a strong feasibility flavor.

Given this state of affairs, this work is motivated by the following question:

Can we build a rate-1 incompressible encryption scheme based on standard assumptions while only making black-box use of cryptographic primitives?

5.1.1 Our Results

In this work, we build a rate-1 incompressible encryption scheme from standard assumptions while only using black-box techniques. Our result uses what we call programmable hash proof systems (HPS) (which are a variant of standard HPS [CS98, CS02] with some additional properties), plain-model incompressible encodings [MW20] and a pseudorandom generator (PRG). In particular, we prove the following theorem.

Theorem 5.1.1 (Informal)**.** *Let S be the storage capacity of the adversary and let n be the size of the encrypted messages. Assuming programmable HPS, incompressible encodings and PRGs exist, there is an incompressible encryption scheme fulfilling the following properties:*

1. *Ciphertexts are of size $n + n^{\varepsilon} \cdot \mathsf{poly}[\lambda]$ for some $\varepsilon > 0$.*
2. *The public key is of size $n^{\varepsilon'} \cdot \mathsf{poly}[\lambda]$ for some $\varepsilon' > 1/2$.*
3. *Moreover, the size of ciphertexts is only slightly larger than the adversary's storage space, that is, $S + \mathsf{poly}[\lambda]$.*

The ciphertext rate $n/(n + n^{\varepsilon} \cdot \mathsf{poly}[\lambda])$ approaches 1 for large enough messages. Additionally, the public key is sublinear in the size of the encrypted message.

In terms of assumptions, incompressible encodings can be based on either decisional composite residuosity (DCR) or learning with errors (LWE). The PRG can be based on any one-way function. We also show that programmable HPS can be instantiated from the decisional Diffie-Hellman (DDH) assumption by tweaking the famous HPS by Cramer and Shoup [CS02]. Consequently, our final incompressible encryption scheme can be based solely on standard assumptions.

Post-quantum construction. In our main construction we crucially use the DDH assumption. It is well-known that DDH can be broken by quantum adversaries, so our construction is prone to quantum attacks [Sho94].

To overcome this issue, we show that the HPS construction of [ADMP20], which is based on isogeny-based assumptions, is also a programmable HPS. The drawback of this construction is that it has large public keys. That is, the public key size grows linearly with the size of the message.

Recall that isogeny-based assumptions are presumably post-quantum secure. By plugging this HPS into our main construction, we obtain a classically secure incompressible encryption scheme only based on post-quantum assumptions.

This does not necessarily mean that our construction is secure against quantum adversaries as we only allow the adversary to compress into classical (non-quantum) memory. While we think limiting the adversary to classical long-term storage is reasonable as long-term quantum storage seems to be even harder to achieve than quantum computation the

scheme could still be insecure against quantum adversaries as demonstrated the cocurrent work of [LMQW22].

Streaming encryption. Streaming encryption/decryption is a property of incompressible encryption schemes which allows the honest encryptor/decryptor to perform operations with very low storage capacity. It is easy to see that streaming decryption is an inherently conflicting property with high rate ciphertexts [GWZ22]. This is because the honest decryptor needs storage at least as large as the size of the message. Otherwise, an adversary can essentially mimic the decryptor and learn something about the encrypted message (e.g., the most significant bit).

However, we note that our scheme has *stream encryption*, i.e., the honest encryptor does not need much space to perform encryption. This follows from the fact that the incompressible encodings construction of [MW20] has stream encoding.

Extension to CCA security. In the security experiment for incompressible encryption presented in [GWZ22] the adversary is never allowed to query a decryption oracle. In other words, their work only considered IND-CPA incompressible encryption. In this work, we also give the adversary access to an decryption oracle extending incompressible encryption to IND-CCA2 incompressible encryption. We stress that IND-CCA2 security is usually considered the *right* security definition to use in practice. We show that our construction is, in fact, is IND-CCA2 incompressible secure.

Focus on the Plain Model. We demonstrate a concretely and asymptotically more efficient construction in the ideal cipher model (ICM). However, we also show that incompressible encryption, which is secure in the ICM, might be prone to attacks as soon as we instantiate the ideal cipher by a specific keyed permutation. Similarly to [CGH04, Den02, GK03, BBP04, MRH04, Bla06, BFM15, GKW17], we construct scheme that can be proven secure in the ICM. However, when we instantiate the ideal cipher, the scheme turns out to be insecure. These observations support our focus on the plain model.

5.1.2 Comparison with Previous Work

As mentioned previously, incompressible encryption was introduced in [GWZ22] where two schemes are presented. The first one is based only on the minimal assumption of PKE. However, the ciphertext rate is very far from 1. The second one achieves rate-1 but is based on i$\mathcal{O}$. We compare these schemes in Table 5.1.

Other related work. Some recent works made significant progress in the area of incompressible cryptography. The works of [DGO19, GLW20, MW20] proposed constructions for incompressible encodings either in the random oracle model or in the CRS model. The work of [GZ21] used the BSM together with computational assumptions to propose constructions of primitives that are not known just from computational assumptions, such as virtual grey-box obfuscation.

Incompressible cryptography is closely related to the bounded storage model (BSM) [Mau92]. However, most works in the BSM (e.g. [CM97, AR99, Raz17, GZ19, DQW21]) focus on achieving unconditional security for primitives that are already known from computational assumptions such as public-key encryption and oblivious transfer.

	Ciphertext Rate	Public key Size	Hardness Assumption	Security
[GWZ22]	$1/\mathsf{poly}[\lambda]$	$\mathsf{poly}[\lambda]$	PKE	CPA
[GWZ22]	1	$\mathsf{poly}[\lambda]$	i$\mathcal{O}$	CPA
Our result	1	$n^{\varepsilon'} \cdot \mathsf{poly}[\lambda]$	DDH + {DCR,LWE}	CCA
Our result	1	$n \cdot \mathsf{poly}[\lambda]$	isogeny-based + {DCR,LWE}	CCA
Our result	1	$n^{\epsilon'} \cdot \omega(\text{polylog}(\lambda))$	LWE	CPA

Table 5.1: Comparison with previous work. Here, n denotes the size of the encrypted messages and ε' is any constant between 1/2 and 1.

Open Problems. We leave the open problem of developing an incompressible encryption scheme that combines concretely short public keys with small ciphertexts. A possible approach for this would be to find a programmable hash proof system where the size of the public key is essentially independent of the size of the encapsulated key.

It might also be interesting if our construction only using post-quantum assumption is post-quantum secure. If this is not the case, a construction that is also post-quantum secure is of interest.

Subsequent Work. In subsequent work, [GWZ23] built upon incompressible encryption schemes, including the ones presented in this work, to construct schemes that can be consider a multi-user incompressible encryption scheme. It forces an adversary that wants to learn information about ciphertexts by receiving the key to keep many ciphertexts around. I.e. it can not hope to compress the ciphertexts from multiple users and still hope to learn information about the contents of all of the ciphertexts.

5.2 Technical Overview

In this technical overview, we sketch the main techniques to build an IND-CPA incompressible scheme. We later argue how these techniques can be tweaked to obtain a scheme that is IND-CCA2 incompressible secure.

Security Notion The syntax and correctness notions for incompressible encryption are identical to standard public-key encryption (PKE). The main difference is in the security definition. Since the security notion of incompressible encryption is relatively new, we will briefly detail its security experiment here. Consider the following security game between a challenger $\mathcal{C}$ and a 3-stage PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$.

1. $\mathcal{C}$ creates a pair of public and secret keys pk, sk.
2. Given pk, the first stage $\mathcal{A}_1$ chooses two messages $\mathsf{m}_0, \mathsf{m}_1$.
3. $\mathcal{C}$ chooses $b \xleftarrow{\$} \{0,1\}$ uniformly at random and encrypts $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_b)$.
4. Given the ciphertext ct and the state of $\mathcal{A}_1$, the second stage $\mathcal{A}_2$ produces a state st of size $S < |\mathsf{ct}|$. That is, the state st should be somewhat smaller than ct.

5. Now, the third stage $\mathcal{A}_3$ receives as input the state st (produced by $\mathcal{A}_2$) and the secret key sk. The goal of $\mathcal{A}_3$ is to guess the bit b.

We say that an incompressible encryption is secure if, for any adversary, $\mathcal{A}$ the advantage of winning the following game is negligible in the security parameter λ.

5.2.1 The Scheme of GWZ

Before we provide an outline of our construction, we will briefly discuss the underlying ideas of the low-rate incompressible encryption scheme constructed in [GWZ22]. At the very core is the following idea: The ciphertext essentially consists of a very long truly random random string R and a short payload part $c = (c_1, c_2)$, where c_1 is an encryption of a seed k for a randomness extractor Ext, and $c_2 = \mathsf{Ext}(k, R) \oplus \mathsf{m}$ is essentially a one-time-pad encryption of the message m under the key $\mathsf{Ext}(k, R)$. Clearly, if c_1 was *not* part of the ciphertext, then security of this scheme follows routinely by the following observations:

- In the view of the third stage $\mathcal{A}_3$ of the adversary R has high min-entropy, as R is uniformly random and the state st is significantly shorter than R.
- Furthermore, as we assume c_1 is not part of the ciphertext, st is independent of k
- Hence by the extraction property of Ext the string $\mathsf{Ext}(k, R)$ is uniformly random in the adversary's view, and therefore m_b is statistically hidden.

Now, the main idea of [GWZ22] to make this approach work even though c_1 is part of the ciphertext is to encrypt k in such a way that c_1 *can be made independent* of the extractor seed k. This is achieved by choosing a suitable encryption scheme for which c_1 can be chosen independently of k, and a suitable secret key which decrypts c_1 to k can be chosen *after the fact*, i.e. after the leakage st has been computed. [GWZ22] provide an elegant construction of such a scheme from non-compact single-key functional encrytion, which can be built from any public key encryption scheme [GVW12].

5.2.2 The Big Picture

While our construction departs significantly from the blueprint of [GWZ22] we use the same high-level concept of an encryption scheme that allows delaying secret-key generation in the security proof. Rather than constructing incompressible PKE directly, we first tackle the intermediate and simpler task of realizing a rate-1 *incompressible symmetric-key encryption*. In a second step, we will then transform *any* incompressible SKE scheme into an incompressible PKE scheme in a rate-preserving way. It turns out that even constructing a rate-1 incompressible SKE from standard assumptions is a non-trivial task and does not follow, e.g. from the (low-rate) public-key construction of [GWZ22].

Since our two steps are independent of one another, improvements of either in future work will lead to better incompressible encryption schemes. For simplicity, in the following outline, we will focus only on CPA security, whereas in the main body, we present a CCA secure construction.

5.2.3 Rate-1 Incompressible Symmetric-Key Encryption

In the symmetric-key setting, the syntax and correctness of incompressible SKE are pretty much that of standard symmetric-key encryption, whereas the security notion is similar to

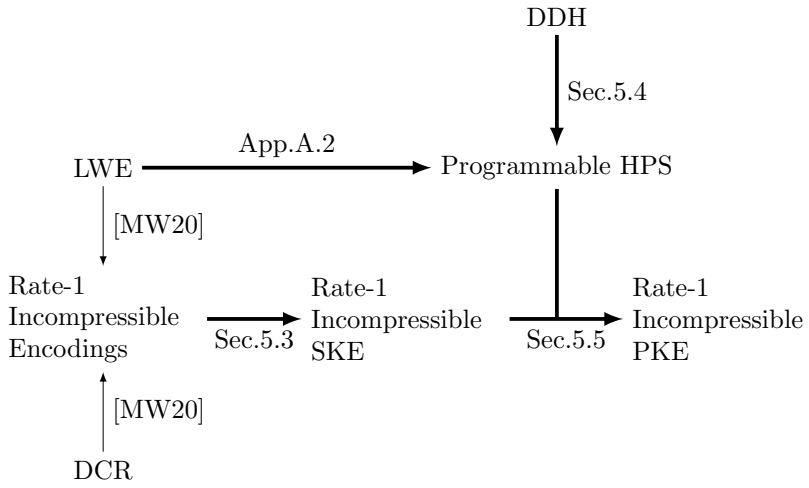

Figure 5.1: Overview of the results in this work, bold arrows are contributions of this work.

that of incompressible PKE, just with the difference that the first stage $\mathcal{A}_1$ of the adversary is not given a public key (as there is none). Thus, the security notion we consider here is the incompressible encryption-analogue of security against an eavesdropper (IND-EAV).

Failed Naive Attempts. As a (failed) very first attempt, one may try "make work" an incompressible SKE construction from the One-Time-Pad (OTP), i.e. the secret key k is a random bit-string as long as the message m and the ciphertext is $c = \mathsf{k} \oplus \mathsf{m}$. However, the obvious issue with this is that such a ciphertext c *decomposes* into many one-bit ciphertexts $c_i = \mathsf{k}_i \oplus \mathsf{m}_i$, and it is enough for $\mathcal{A}_2$ to leak a few bits of c to enable $\mathcal{A}_3$ to partially decrypt c and thus distinguish encryptions of m_0 from encryptions of m_1. As a next idea, one may try the following: Encryption chooses a (fresh) pseudorandom generator (PRG) seed s, encrypt m into $\hat{\mathsf{m}} = \mathsf{m} \oplus \mathsf{G}(\mathsf{s})$, use k to encrypt the seed s into a *header ciphertext* c, i.e. the encryption of m is $(\mathsf{Enc}(\mathsf{k}, \mathsf{s}), \mathsf{m} \oplus \mathsf{G}(\mathsf{s}))$. While this approach does look promising, we observe that it is stuck at either *leakage-rate* $1/2$ or ciphertext-rate $1/2$, that is as soon as $\mathcal{A}_3$ learns $\mathsf{Enc}(\mathsf{k}, \mathsf{s})$ in its entirety and a few bits of $\mathsf{m}_b \oplus \mathsf{G}(\mathsf{s})$, he will be able to distinguish encryptions of m_0 from m_1.

Introducing Circularity. Clearly, we need some kind of mechanism to *glue* the two ciphertexts components together, i.e. we want to make it such that if some parts of $\hat{\mathsf{m}}$ are missing, then c will be useless (and vice versa). As a first, heuristic "hands-on" approach to achieve this, we can try to use $\hat{\mathsf{m}}$ as a source of randomness from which we extract a key to mask the seed s. Thus, let $\mathsf{Ext}(\cdot, \cdot)$ be a *seeded* randomness extractor. We compute a ciphertext $(c, \hat{\mathsf{m}})$ by first computing $\hat{\mathsf{m}} = \mathsf{m} \oplus \mathsf{G}(\mathsf{s})$ for a random seed s as before, but then encrypt s into c via $c = \mathsf{s} \oplus \mathsf{Ext}(\mathsf{k}, \hat{\mathsf{m}})$, i.e. we use k as an extractor seed to extract a one-time-pad key $\mathsf{Ext}(\mathsf{k}, \hat{\mathsf{m}})$ from $\hat{\mathsf{m}}$. Clearly, given k and a ciphertext $(c, \hat{\mathsf{m}})$, we can decrypt by first computing $s = c \oplus \mathsf{Ext}(\mathsf{k}, \hat{\mathsf{m}})$ and then $\mathsf{m} = \hat{\mathsf{m}} \oplus \mathsf{G}(\mathsf{s})$. The rationale for why we hope this construction to be secure is that as soon as a significant fraction of the bits of $\hat{\mathsf{m}}$ are lost, the output of the extractor $\mathsf{Ext}(\mathsf{k}, \hat{\mathsf{m}})$ should look uniform, and thus $\hat{\mathsf{m}} = \mathsf{m} \oplus \mathsf{G}(\mathsf{s})$ should hide m by the pseudorandomness of G. However, this circularity backfires when trying to establish security of this construction just from the pseudorandomness of G and

the randomness-extraction property of Ext: In order to use pseudorandomness of G, we first need to *remove* the s from the view of the adversary, but $c = \mathsf{s} \oplus \mathsf{Ext}(\mathsf{k}, \hat{\mathsf{m}})$ is correlated with s given k. On the other hand, in order to use the randomness extraction property of Ext we need that $\hat{\mathsf{m}}$ has high entropy given st. But all the entropy of $\hat{\mathsf{m}} = \mathsf{m} \oplus \mathsf{G}(\mathsf{s})$ comes from the seed s, which is very small. Hence $\approx \lambda$ bits of $\hat{\mathsf{m}}$ suffice to information-theoretically determine s.

Consequently, while heuristically, this construction seems secure, it seems unlikely the individual security properties of G and Ext suffice to *prove* this construction secure.

Breaking Circularity Hence, what we need is a mechanism to break the circularity, which we have just introduced. Looking at where establishing security of the above construction gets stuck, a natural point to start is to make it such that $\hat{\mathsf{m}}$ looks like it has a large amount of real entropy once a few bits of $\hat{\mathsf{m}}$ are missing, i.e. $L(\hat{\mathsf{m}})$ being computationally indistinguishable from $L(\hat{r})$ for a high-entropy distribution $\hat{r}$ for any efficiently computable leakage function $L(\cdot)$ [2].

Incompressible encodings. Fortunately, an encoding mechanism achieving this notion called *incompressible encodings* was just recently introduced and constructed by Moran and Wichs [MW20]. As a technical tool, they introduced the notion of *HILL-entropic encoding* in their work, which will be sufficient, if not to say ideally suited for our construction. Such a scheme consists of an encoding algorithm En and a decoding algorithm De, both of which rely on a (large) common random string $\mathsf{crs} \xleftarrow{\$} \{0,1\}^t$:

- The encoding algorithm $\mathsf{En}_{\mathsf{crs}}(m)$ is a *randomized algorithm* which takes a message m and produces an encoding $\hat{m}$
- The decoding algorithm $\mathsf{De}_{\mathsf{crs}}(\hat{m})$ is a deterministic algorithm which takes an encoding $\hat{m}$ and returns a message m.

In terms of correctness, one naturally requires that encoding followed by decoding leads to the original message. Security-wise, we require that there exists a simulator Sim which on input a message m produces a pair $(\mathsf{crs}', \tilde{m})$, which is computationally indistinguishable from a real pair of crs and encoding of m, i.e.

$$(\mathsf{crs}, \mathsf{En}_{\mathsf{crs}}(m)) \approx_c \mathsf{Sim}(m),$$

where $\mathsf{crs} \xleftarrow{\$} \{0,1\}^t$. Additionally, we require that if $(\mathsf{crs}', \tilde{m}) \leftarrow \mathsf{Sim}(m)$, then $\tilde{m}$ has almost full *true* min-entropy given crs', i.e. $\tilde{H}_\infty(\tilde{m}|\mathsf{crs}') \geq (1-\epsilon)n$, where $\tilde{H}_\infty$ is the average conditional min-entropy. The (very) high level idea how this can be achieved is that in simulation the common random string and the encoded message *switch roles*, in the sense that the simulated common random string crs' encodes the message m, whereas the encoding $\tilde{m}$ now has room to have (very) high entropy.

Moran and Wichs [MW20] provide two instantiations of their construction, one from DCR and one from LWE. These constructions achieve rate-1, i.e., the encoding is only slightly larger than the encoded message. The conceptual idea behind the construction is rather elegant: The encoding function $\mathsf{En}_{\mathsf{crs}}(m)$ generates a pair of public key and trapdoor $(\mathsf{pk}, \mathsf{td})$ of *preimage-sampleable surjective lossy function* F (for which we have efficient constructions from DCR or LWE) and sets x to be a randomly sampled preimage of $m \oplus \mathsf{crs}$,

[2] In our case the leakage function L is described by the adversary's second stage $\mathcal{A}_2$

i.e. $x = F_{\mathsf{td}}^{-1}(m \oplus \mathsf{crs})$, and sets $\hat{m} = (\mathsf{pk}, x)$. To decode $\hat{m}$, one computes $m = F_{\mathsf{pk}}(x) \oplus \mathsf{crs}$. The simulator Sim chooses a *highly lossy public key* $\tilde{\mathsf{pk}}$, chooses x uniformly at random, and sets $\mathsf{crs}' = m \oplus F_{\tilde{\mathsf{pk}}}(x)$ and $\tilde{m} = (\tilde{\mathsf{pk}}, x)$. Given that F_{pk} is *regular* for surjective keys pk, meaning that if x is uniform then $F_{\mathsf{pk}}(x)$ is also (statisticalluy close to) uniform, we can routinely establish that real pairs $(\mathsf{crs}, \hat{m})$ are computationally indistinguishable from simulated $(\mathsf{crs}', \tilde{m})$ using the indistinguishability of surjective public keys pk and highly lossy public keys $\tilde{\mathsf{pk}}$. Moreover, for simulated pairs $(\mathsf{crs}' = m \oplus F_{\tilde{\mathsf{pk}}}(x), \tilde{m} = (\tilde{\mathsf{pk}}, x))$ we can easily argue that x (and hence $\tilde{m}$) has high min-entropy given $\mathsf{crs}' = m \oplus F_{\tilde{\mathsf{pk}}}(x)$, as $F_{\tilde{\mathsf{pk}}}$ is highly lossy and hence x has high min entropy given $F_{\tilde{\mathsf{pk}}}(x)$.

Moran and Wichs [MW20] go on to show that for any incompressible/HILL-entropic encoding, the common random string crs must be as long as the message, if one wants to establish security from a *falsifiable assumption* [Nao03] under a black-box reduction.

The Full Construction. We will now provide our complete construction of incompressible SKE and sketch the security proof. For our scheme, the secret key K is a uniformly random bit-string of suitable length which will be parsed as $\mathsf{K} = (\mathsf{crs}, \mathsf{k})$, where crs is the common random string for a HILL-entropic encoding $(\mathsf{En}, \mathsf{De})$, and k is the seed for a randomness extractor Ext. Encryption and decryption work as follows.

- $\mathsf{Enc}(\mathsf{K} = (\mathsf{crs}, \mathsf{k}), \mathsf{m})$: Choose a uniformly random PRG seed $\mathsf{s} \xleftarrow{\$} \{0,1\}^\lambda$ and compute $\hat{\mathsf{m}} = \mathsf{En}_{\mathsf{crs}}(m \oplus \mathsf{G}(\mathsf{s}))$. Compute $c = \mathsf{s} \oplus \mathsf{Ext}(\mathsf{k}, \hat{\mathsf{m}})$ and output the ciphertext $\mathsf{ct} = (c, \hat{\mathsf{m}})$.
- $\mathsf{Dec}(\mathsf{K} = (\mathsf{crs}, \mathsf{k}), \mathsf{ct} = (c, \hat{\mathsf{m}}))$: Compute $\mathsf{s} = c \oplus \mathsf{Ext}(\mathsf{k}, \hat{\mathsf{m}})$ and output $m = \mathsf{De}_{\mathsf{crs}}(\hat{\mathsf{m}}) \oplus \mathsf{G}(\mathsf{s})$.

Correctness of this scheme follows routinely.

Security of this scheme is established along the following lines. First we rely on the security of the HILL-entropic encoding to replace $(\mathsf{crs}, \hat{\mathsf{m}})$ with a *simulated pair* $(\mathsf{crs}', \tilde{\mathsf{m}}) = \mathsf{Sim}(\mathsf{m} \oplus \mathsf{G}(\mathsf{s}))$. By the security of the HILL-entropic encoding, this modification is (computationally) unnoticeable to the adversary. However, now the encoding $\tilde{\mathsf{m}}$ has true high min-entropy given crs'. Thus, using a min-entropy chain rule (e.g. by [DORS08]) we can argue that $\tilde{\mathsf{m}}$ still has sufficiently high min-entropy given both crs' and a leak $L(\tilde{\mathsf{m}})$. Hence, the randomness extraction property guarantees that $\mathsf{Ext}(\mathsf{k}, \tilde{\mathsf{m}})$ will extract uniform randomness (given crs' and $L(\tilde{\mathsf{m}})$). To establish this we need a mild extra property of the extractor Ext that given a (uniformly random) extractor output y and $\tilde{\mathsf{m}}$ we can sample a key k' *after the fact* such that $(\mathsf{k}', y) \approx (\mathsf{k}, \mathsf{Ext}(\mathsf{k}, \tilde{\mathsf{m}}))$. Hence in the next hybrid modification, we can thus replace $c = \mathsf{s} \oplus \mathsf{Ext}(\mathsf{k}, \tilde{\mathsf{m}})$ with a uniformly random and independent string c'. Now that c' is independent of s, we can use the pseudorandomness property of G to replace $\mathsf{m} \oplus \mathsf{G}(\mathsf{s})$ in $(\mathsf{crs}', \tilde{\mathsf{m}}) = \mathsf{Sim}(\mathsf{m} \oplus \mathsf{G}(\mathsf{s}))$ with a uniformly random string u, i.e. $(\mathsf{crs}', \tilde{\mathsf{m}}) = \mathsf{Sim}(u)$. We have finally arrived at an experiment where the ciphertext $\mathsf{ct} = (c', \tilde{\mathsf{m}})$ is independent of the message m, and hence the adversary's advantage is 0.

Concerning the rate of this scheme, note that a ciphertext $\mathsf{ct} = (c, \hat{\mathsf{m}})$ has rate 1, as c is just of size $\mathsf{poly}(\lambda)$ (independent of the message length n), and the HILL-entropic encoding $\hat{\mathsf{m}}$ is rate 1.

5.2.4 From Symmetric-Key to Public-Key Incompressible Encryption via Hash Proof Systems

Now that we have a construction of incompressible SKE, we need a way to establish a *long* key K between the sender and receiver. This is a job for a *key encapsulation mechanism*

(KEM) [CS03]. A key-encapsulation mechanism consists of:

- A key-encapsulation mechanism consists of a key-generation algorithm KeyGen which produces a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$.
- An encapsulation algorithm which takes a public key pk and produces a symmetric key K and a ciphertext header c_0 *encapsulating* K.
- A decapsulation algorithm Dec which takes a secret key sk and a ciphertext header c_0 and outputs a key K.

The correctness requirement is the obvious one, whereas the standard security requirement is that K is pseudorandom given pk and c_0. A symmetric key K generated via a KEM can now be used to encrypt a message m into a payload ciphertext c_1 using a symmetric key encryption scheme. The full ciphertext is $c = (c_0, c_1)$.

However, to transform an incompressible SKE into an incompressible PKE *not just any key encapsulation mechanism will do.* The simple reason is that in the incompressible (public key) encryption security game, the adversary gets to see the secret key sk in the end, which will allow him to decapsulate the (short) ciphertext header c_0 into the symmetric key K. But the standard security notion of KEMs discussed above does not require that the encapsulated key K follows a uniform distribution. Indeed, e.g. for simple PRG-based KEMs, the encapsulated key is statistically far from uniform. However, recall that in our construction of incompressible SKE above, we made critical use of the fact that the key K follows a uniform distribution and that the security reduction can *program* it in a suitable way.

Thus, we need a KEM which we can switch into a mode in which the ciphertext header c_0 encapsulates a truly uniform key K. As we need the ciphertext header c_0 to be substantially shorter than the encapsulated key K, the entropy of K in this mode *must* come from the secret key sk.

Enter Hash proof systems. This is where hash proof systems (HPS) [CS02] come into play [3]. Recall that HPS are defined relative to an NP-language $\mathcal{L} \subseteq \{0,1\}^k$. We have a key-generation algorithm KeyGen which generates a public or *projected* key pk, and a secret or *hashing* key sk. The hashing or decapsulation algorithm Decap takes the secret key sk and *any* $x \in \{0,1\}^k$ and produces a hash value K. The restricted hashing or encapsulation algorithm Encap takes a public key pk, an $x \in \mathcal{L}$ and a witness w (with respect to a fixed NP-relation for $\mathcal{L}$) for membership of x in $\mathcal{L}$ and produces a hash-value K.

In terms of correctness or completeness, we require that Decap and Encap agree on $\mathcal{L}$, i.e. if $x \in \mathcal{L}$ and w is a valid witness for x, then it holds that $\mathsf{Decap}(\mathsf{sk}, x) = \mathsf{Encap}(\mathsf{pk}, x, w)$.

In terms of security, we require *smoothness*, namely given that $x \notin \mathcal{L}$, it holds that $\mathsf{Decap}(\mathsf{sk}, x)$ is statistically close to uniform *given* pk.

HPS are especially useful for sparse pseudorandom languages $\mathcal{L}$, such as the decisional Diffie-Hellman (DDH) language) [CS02]. We define this language with respect to a pair of (randomly chosen) generators $g, h \in \mathbb{G}$, where $\mathbb{G}$ is a cryptographic group of prime order p. A pair $x = (g', h')$ is in $\mathcal{L}$, if there exists an $r \in \mathbb{Z}_p$ such that $g' = g^r$ and $h' = h^r$. The DDH assumption states that a random element in $\mathcal{L}$, i.e. a pair (g^r, h^r) is computationally indistinguishable from a pair of uniformly random group elements (u, v) [4]

[3] HPS have been instrumental in many prior works on leakage resilience cryptography e.g. [ADN+10, HLWW13]

[4] Note that such a pair is not in $\mathcal{L}$, except with negligible probability $1/p$.

In the Cramer-Shoup [CS02] scheme, the secret key $\mathsf{sk} = (\alpha, \beta)$ consists of two uniformly random values $\alpha, \beta \in \mathbb{Z}_p$, and the public key pk is computed as $\mathsf{pk} = g^\alpha h^\beta$. Given a public key pk an instance $c_0 = (g^r, h^r)$ with witness r, we compute a key $\mathsf{K} = \mathsf{pk}^r$. Given a secret key $\mathsf{sk} = (\alpha, \beta)$ and an instance $c_0 = (g', h')$ we compute a key $\mathsf{K} = g'^\alpha h'^\beta$. It follows routinely that encapsulation and decapsulation agree on $\mathcal{L}$. Moreover, for a $(g^*, h^*) \notin \mathcal{L}$ it holds $\mathsf{K}^* = g^{*\alpha} h^{*\beta}$ is uniformly random given $\mathsf{pk} = g^\alpha h^\alpha$ by a simple linear algebra argument.

Hash Proof Systems, and in particular the Cramer-Shoup HPS (almost) give us a KEM with the desired properties. Namely, given pk and (g, h), to encapsulate a key k we choose a uniformly random $r \in \mathbb{Z}_p$ and compute $c_0 = (g^r, h^r)$ and $\mathsf{K} = \mathsf{pk}^r$. To decapsulate K from $c_0 = (g', h')$ given $\mathsf{sk} = (\alpha, \beta)$, we compute $\mathsf{K} = g'^\alpha h'^\beta$.

A typical proof-strategy using HPS lets a reduction compute the encapsulated key (on the sender's side) via the decapsulation algorithm using the secret key. Correctness of the HPS ensures that this does not change K. Hence this modification will not be detected by an adversary. Now we don't need the witness r anymore. We can replace c_0 with a uniformly random c_0' and argue that this modification is computationally undetectable by the adversary, thanks to the DDH assumption. Since now c_0' is outside of $\mathcal{L}$ w.o.p, it holds that K is uniform even given pk, as desired.

However, this is still not enough to make our security reduction go through. It turns out we not only have to ensure that K is uniform given pk, but also that for *any given* K and fixed pk and c_0 we can find a secret key sk (compatible with pk) such that $\mathsf{Decap}(\mathsf{sk}, c_0) = \mathsf{K}$. Realizing this property using the Cramer-Shoup HPS directly seems hard, as in order to sample an $\mathsf{sk} = (\alpha, \beta)$ with $\mathsf{K} = g'^\alpha h'^\beta$ we would need to compute a discrete logarithm of K.

Programmable Hash Proof Systems For this purpose, we will consider a notion of *programmable* hash proof systems, which obey a stronger smoothness notion. In short, such an HPS has the following property. Given a public key pk, a (fake) ciphertext header c_0^* (not in $\mathcal{L}$) and secret auxiliary information aux depending on both pk and c_0, we can sample a uniformly random secret key sk^* such that $\mathsf{Decap}(\mathsf{sk}^*, c_0^*) = \mathsf{K}$, for which it holds that $(\mathsf{pk}, c_0^*, \mathsf{sk}^*) \approx_s (\mathsf{pk}, c_0^*, \mathsf{sk})$ if K is chosen uniformly random.

Our idea to achieve this is simple: We will concatenate Decap (and also Encap) with a *balanced small range* hash function $\mathsf{HC} : \mathbb{G} \to \{0,1\}$, i.e. we have $\mathsf{Decap}'(\mathsf{sk}, c_0) = \mathsf{HC}(\mathsf{Decap}(\mathsf{sk}, c_0))$ and $\mathsf{Encap}'(\mathsf{pk}, c_0, r) = \mathsf{HC}(\mathsf{Encap}(\mathsf{pk}, c_0, r))$. Here balanced means that if $h \in \mathbb{G}$ is a uniformly random group element, then $\mathsf{HC}(h)$ is statistically close to a uniformly random bit. While there exist deterministic constructions of such extractors for certain groups (e.g. [CFPZ09]) we can find such an HC for any group via the leftover-hash lemma [HILL99]. For such a hash function, we can efficiently sample a uniformly random pre-image $h \in \mathbb{G}$ of K for which we *do know* the discrete logarithm (with respect to a generator $g \in \mathbb{G}$). We achieve this via rejection sampling: Given a bit $\mathsf{K} \in \{0,1\}$, choose a uniformly random $z \in \mathbb{Z}_p$ and test whether $\mathsf{HC}(g^z) = \mathsf{K}$ (which happens with probability $1/2$), and reject and resample if the test fails.

Now let $h = g^y$, $\mathsf{pk} = g^t$ and $c_0^* = (g' = g^r, h' = g^s)$ be a public key and (fake) ciphertext, for which the auxiliary information is (y, t, r, s), i.e. the discrete logarithms of pk and c_0^*. Given a key $\mathsf{K} \in \{0,1\}$, we first sample a uniformly random $z \in \mathbb{Z}_p$ such that $\mathsf{HC}(g^z) = \mathsf{K}$. Now we have 2 linear constraints (over $\mathbb{Z}_p$) on $\mathsf{sk} = (\alpha, \beta) \in \mathbb{Z}_p^2$, namely

$$t = \alpha + \beta \cdot y$$

from $\mathsf{pk} = g^\alpha \cdot h^\beta$ and

$$z = \alpha r + \beta s$$

from $\mathsf{HC}(g^z) = \mathsf{HC}(g'^{\alpha} \cdot h'^{\beta})$. Since we now have two equations and two unknowns α and β, we can solve for α and β using basic linear algebra.

We do pay a price to get programmability: Instead of getting $\log(|\mathbb{G}|)$ key bits per public key pk, we only get a single bit. Naturally, this can be improved up to $\log(\lambda)$ key-bits while keeping the above rejection sampling procedure expected polynomial time.

The Full Construction We are now ready to present our fully-fledged construction. This construction will have a large public key. We will later discuss how the size of the public key can be reduced.

Assume thus that $(\mathsf{Enc}, \mathsf{Dec})$ is an incompressible SKE scheme, and that $(\mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ is a programmable HPS for a decision-membership-hard language $\mathcal{L}$, for concreteness assume the DDH language. Our incompressible PKE construction is given by the following algorithms.

- The key-generation algorithm KeyGen' generates random group elements $g, h \in \mathbb{G}$ and n pairs of public and secret keys $(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_n, \mathsf{sk}_n)$ using KeyGen (on g, h) and set the public key $\mathsf{PK} = (g, h, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ and the secret key $\mathsf{SK} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$.
- The encryption algorithm Enc' proceeds as follows, given a public key $\mathsf{PK} = (g, h, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ and a message m. First, generate a random DDH instance $c_0 = (g' = g^r, h' = h^r)$ using a random $r \xleftarrow{\$} \mathbb{Z}_p$. Now compute the key-bits $\mathsf{K}_1 = \mathsf{Encap}(\mathsf{pk}_1, c_0, r), \ldots, \mathsf{K}_n = \mathsf{Encap}(\mathsf{pk}_n, c_0, r)$ and set $\mathsf{K} = (\mathsf{K}_1, \ldots, \mathsf{K}_n)$. Next, we use K to encrypt m using the incompressible SKE scheme, i.e. we compute $c_1 = \mathsf{Enc}(\mathsf{K}, \mathsf{m})$ and output the ciphertext $c = (c_0, c_1)$.
- The decryption algorithm Dec' takes a secret key $\mathsf{SK} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ and a ciphertext $c = (c_0, c_1)$, and proceeds as follows. First, it decapsulates the key $\mathsf{K} = (\mathsf{K}_1, \ldots, \mathsf{K}_n)$ by computing $\mathsf{K}_1 = \mathsf{Decap}(\mathsf{sk}, c_0)$, $\ldots$, $\mathsf{K}_n = \mathsf{Decap}(\mathsf{sk}_n, c_0)$. Next, it decrypts c_1 to m via $\mathsf{m} = \mathsf{Dec}(\mathsf{K}, c_1)$.

Correctness of this scheme follows routinely from the correctness of its components.

Note that if the incompressible SKE scheme $(\mathsf{Enc}, \mathsf{Dec})$ is rate-1, then so is our public-key scheme $(\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$, as the only additional information in ciphertexts $c = (c_0, c_1)$ is the header c_0, which consists of just two group elements. On the other hand, note that the size of the public key of this scheme scales with the size n of the symmetric key K, which in our symmetric-key construction scales with the size of the message m.

Security of the Full Construction We will now turn to sketching the security proof for the main construction. In the first hybrid step (somewhat expectedly), we use the HPS Decap algorithm instead of the Encap algorithm to compute the key-bits K_i *in the encryption of the challenge ciphertext*. That is, in the encryption of the challenge ciphertext we replace $\mathsf{K}_i = \mathsf{Encap}(\mathsf{pk}_i, c_0, r)$ with $\mathsf{K}_i = \mathsf{Decap}(\mathsf{sk}_i, c_0)$ for all $i = 1, \ldots, n$. Due to the correctness property of the HPS, this modification does not change the distribution of K. Hence this hybrid change goes unnoticed by the adversary. In the second hybrid step, since we don't need r anymore, we replace $c_0 = (g^r, h^r)$ with a uniformly random c_0'. We can use the DDH assumption to argue that this modification goes unnoticed.

The next hybrid step is the critical one: We choose g, h, the pk_i and c_0' with auxiliary information, i.e. together with their discrete logarithms with respect to g, choose $\mathsf{K} \xleftarrow{\$} \{0,1\}^n$ uniformly at random and sample each sk_i such that $\mathsf{K}_i = \mathsf{Decap}'(\mathsf{sk}_i, c_0')$ using the programming algorithm of the programmable HPS. We can argue statistical indistinguishability

using the programmability property of HPS. The crucial observation now is that the public key $\mathsf{PK} = (g, h, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ and the ciphertext header c_0 are computed *independently* of K and SK, and in fact we choose SK depending on K, i.e. we can choose SK after everything else.

This now allows us to turn an adversary $\mathcal{A}$ with non-negligible advantage in this hybrid experiment into an adversary $\mathcal{A}'$ with the same advantage against the incompressible SKE scheme. $\mathcal{A}'$ first generates PK as in the hybrid experiment and provides PK to the first stage $\mathcal{A}_1$ of $\mathcal{A}$, which will output m_0, m_1. Now the second stage $\mathcal{A}'_2$ gets to see a symmetric-key encryption c_1 of m_b, and turns this into a public-key encryption by setting $c = (c_0, c_1)$, where c_0 computed as in the hybrid experiment. This ciphertext c is then given $\mathcal{A}_2$, which outputs a state/leak st, and $\mathcal{A}'_2$ outputs the same state st.

Finally, $\mathcal{A}'_3$ given a symmetric key K and the state st proceeds as follows. Using the auxiliary information aux[5]. and the key K, it samples a secret key $\mathsf{SK} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ such that for all $i = 1, \ldots, n$ it holds that $\mathsf{K}_i = \mathsf{Decap}'(\mathsf{sk}_i, c_0)$, as in the hybrid experiment. Then, $\mathcal{A}'_3$ runs $\mathcal{A}_3$ on SK and st and outputs whatever $\mathcal{A}_3$ outputs.

It is not hard to see that from the view of $\mathcal{A}$, $\mathcal{A}'$ simulates the hybrid experiment perfectly. Hence, the advantage of $\mathcal{A}'$ against the incompressible symmetric-key security experiment is the same as that of $\mathcal{A}$ against the hybrid experiment, and we derive the desired contradiction.

Reducing the Public-Key-Size. As mentioned above, the construction we discussed in the last two paragraphs has a near-optimal ciphertext size (i.e. increasing the size of the symmetric-key ciphertext only by two group elements). In contrast, it has a very large public key which scales *linearly* with the size of the encrypted messages/the ciphertexts.

We will now discuss a tradeoff which achieves a better balance between ciphertext size and public key size. Concretely, we will provide a tradeoff which achieves a ciphertext size of $n + n^{\epsilon}\mathsf{poly}(\lambda)$ for an $0 < \epsilon < 1$ and public key size $n^{\epsilon'}\mathsf{poly}(\lambda)$ for an $1/2 < \epsilon' < 1$. I.e. we achieve ciphertext rate $1 - n^{\epsilon-1}\mathsf{poly}(\lambda)$, which approaches 1 for sufficiently large n, while having a key of sublinear size.

In order to declutter the presentation, we will switch from multiplicative notation of group operations in $\mathbb{G}$ to additive notion in the following discussion. That is we will denote group elements g^x by $[x]$, and write $\alpha \cdot [x]$ instead of $(g^x)^\alpha$. Furthermore, we will consider vectors and matrices of group elements, i.e. if $\mathbf{x} \in \mathbb{Z}_p^k$ is a vector, then $[\mathbf{x}]$ is its element-wise encoding in the group $\mathbb{G}$. Likewise, we write an encoding of a matrix $\mathbf{A} \in \mathbb{Z}_p^{k \times l}$ as $[\mathbf{A}]$.

In our discussion above we considered a HPS for the *two-dimensional* DDH language, i.e. the language consisting of all $r \cdot [\mathbf{v}]$ given two $[\mathbf{v}]$, where $\mathbf{v} \in \mathbb{Z}_p^2$ is a randomly chosen 2-dimensional vector over $\mathbb{Z}_p$.

Thus let $\mathbf{v} \in \mathbb{Z}_p^k$ be a randomly chosen k-dimensional vector. The goal of the k-dimensional DDH problem is to distinguish $([\mathbf{v}], t \cdot [\mathbf{v}])$ from $([\mathbf{v}], [\mathbf{u}])$, where $\mathbf{v}$ and $\mathbf{u}$ are chosen uniformly random from $\mathbb{Z}_p^k$ and r is chosen uniformly from $\mathbb{Z}_p$. It follows routinely via a standard rerandomization argument that the k-dimensional DDH problem is hard, given that the 2-dimensional DDH problem is hard.

We can construct an HPS for k-DDH analogously to the 2-dimensional case: Fix a vector $[\mathbf{v}] \in \mathbb{G}^k$. The secret key sk is a random vector $\boldsymbol{\alpha} \in \mathbb{Z}_p^k$, whereas the public key is given by

[5]There is a technical subtlety in the security definition of incompressible SKE which we omitted before: We allow the first stage $\mathcal{A}'_1$ of a symmetric-key adversary $\mathcal{A}'$ to produce a large state (i.e. scaling with the message size), which is provided to *both* $\mathcal{A}'_2$ and $\mathcal{A}'_3$. This is to *communicate* a potentially large public key PK from $\mathcal{A}_1$ to $\mathcal{A}_3$ without putting a burden on the leakage-budget of the leaker-stage $\mathcal{A}'_2$. One could consider an alternative definition where this communication from $\mathcal{A}'_1$ to $\mathcal{A}'_3$ is not allowed. In such a setting we could still prove our construction secure by *compressing* the auxiliary information aux from which PK and c_0 are generated using a PRG

$[\mathsf{pk}] = \boldsymbol{\alpha}^\top[\mathbf{v}]$, i.e. the inner product of $\boldsymbol{\alpha}$ and $[\mathbf{v}]$. Given a vector $[\mathbf{w}] = r \cdot [\mathbf{v}]$ and a witness r, the Encap algorithm computes $[\mathsf{K}] = r \cdot [\mathsf{pk}]$. On the other hand, given any vector $[\mathbf{w}] \in \mathbb{G}^k$ and a secret key $\mathsf{sk} = \boldsymbol{\alpha}$, the Decap algorithm computes $\boldsymbol{\alpha}^\top \cdot [\mathbf{w}]$. Arguing correctness and smoothness are again simple exercises in linear algebra. Furthermore, this HPS satisfies a stronger notion of $k-1$-smoothness: Given uniformly random $[\mathbf{w}_1], \ldots, [\mathbf{w}_{k-1}]$, it holds that

$$(\mathsf{pk}, \boldsymbol{\alpha}^\top[\mathbf{w}_1], \ldots, \boldsymbol{\alpha}^\top[\mathbf{w}_{k-1}]) \approx_s (\mathsf{pk}, [u_1], \ldots, [u_{k-1}]),$$

where the $[u_1], \ldots, [u_{k-1}]$ are uniformly random in $\mathbb{G}$. Establishing this is again routine linear algebra.

We will first briefly discuss how the HPS can be made programmable. In essence, we follow the same idea as above: We take a balance function $\mathsf{HC} : \mathbb{G} \to \{0,1\}$ and define the Decap algorithm to compute $\mathsf{HC}(\boldsymbol{\alpha}^\top[\mathbf{w}])$. We claim this construction is $k-1$-programmable. That is, given $[\mathbf{v}]$, $[\mathsf{pk}] = [t]$, uniformly random $[\mathbf{w}_1], \ldots [\mathbf{w}_{k-1}]$ together with the witnesses $\mathbf{v}$, t and $\mathbf{w}_1, \ldots, \mathbf{w}_{k-1}$, and a random $\mathsf{K} = (\mathsf{K}_1, \ldots, \mathsf{K}_{k-1}) \in \{0,1\}^{k-1}$, we can efficiently sample a uniformly random $\boldsymbol{\alpha} \in \mathbb{Z}_p^k$ such that $t = \boldsymbol{\alpha}^\top\mathbf{v}$ and $\mathsf{K}_i = \mathsf{HC}(\boldsymbol{\alpha}^\top[\mathbf{w}_i])$ for $i = 1, \ldots, n$. We proceed as above: First we choose uniformly random $z_i \in \mathbb{Z}_p$ such that $\mathsf{K}_i = \mathsf{HC}([z_i])$ for all i. Then we get the linear equation system

$$\begin{aligned}
\boldsymbol{\alpha}^\top\mathbf{v} &= t \\
\boldsymbol{\alpha}^\top\mathbf{w}_1 &= z_1 \\
&\vdots \\
\boldsymbol{\alpha}^\top\mathbf{w}_{k-1} &= z_{k-1}.
\end{aligned}$$

Since the $\mathbf{w}_i$ are chosen uniformly random, this system has full rank w.o.p., and hence we can find a matching secret key $\boldsymbol{\alpha}$ via simple linear algebra.

Now, plugging this programmable HPS into our construction of incompressible PKE, we obtain the following parameters.

- A single public pk consisting of one group element can be used to encapsulate k key bits. Hence, to encapsulate n key bits we need n/k public keys amounting to n/k group elements.
- The ciphertext header now contains $k \cdot (k-1) \leq k^2$ group elements (in the above notation the vectors $[\mathbf{w}_1], \ldots, [\mathbf{w}_{k-1}]$).

Hence, if we want to strike a balance where the (additive) ciphertext overhead is of the same size as the public key, we obtain the relation

$$\frac{n}{k} = k^2,$$

which yields to $k = n^{1/3}$. Hence, for this choice of parameters the public key consists of a $n^{2/3}$ group elements (which is sublinear), and the size of the ciphertext is $n + n^{2/3}\log(|\mathbb{G}|) = n(1 - n^{-1/3}\log(|\mathbb{G}|))$ bits, which approaches rate 1.

5.2.5 Extension to CCA security

The scheme described so far achieves IND-CPA incompressible security. This work also considers an IND-CCA2 incompressible security definition where the adversary gets oracle access to a decryption oracle.

To achieve IND-CCA2 security, we follow the framework of [CS02]. We add a second hash proof system that acts as integrity proof for ciphertexts. The second hash proof system does not need to be programmable but universal$_2$ [CS02] or 2-smooth [ABP15]. It allows the decryption oracle to only answer queries to honestly generated ciphertexts. This mechanism ensures that the decryption oracle does not give up entropy of the programmable HPS's secret key.

In the main body of this work, we provide the full construction that achieves this level of security.

5.2.6 Incompressible Encryption in the ICM

Finally, we present a scheme that is secure in the ideal cipher model (ICM) but insecure when we use a concrete keyed permutation. The scheme is a simple hybrid encryption scheme where the symmetric encryption is one huge block cipher. In previous versions of this paper we claimed that the ICM is equivalent to the random oracle model. This, however, only holds in single stage security games as pointed out by [GWZ23].

More concretely, the public key is composed of a pk of an IND-CPA scheme. To encrypt a message, we first sample two strings r, k of size λ and compute $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{k})$. Then we compute $d = \mathsf{P}_\mathsf{k}((r, \mathsf{m}))$ where P is modelled as an ideal cipher oracle.

The scheme is secure in the ICM by observing that: i) The adversary cannot query the ideal cipher oracle with key k before receiving the secret key as it would break the IND-CPA security of the underlying PKE scheme; and ii) Given the secret key, the last stage adversary cannot query the ideal cipher oracle $\mathsf{P}_\mathsf{k}^{-1}$ on d because the limitations of the state size make sure d has high min-entropy. The adversary can also not query P_k on (r, m) as it would have to guess r. Therefore, the adversary has almost no information about the message.

However, if we use a fully-homomorphic encryption (FHE) scheme as a PKE and if we instantiate the ideal cipher oracle with a specific block cipher P, it is the scheme becomes breakable. The key idea is that as soon as the ideal cipher oracle is instantiated with a block cipher P, the adversary has access to the code of P and can thus run it homomorphically under the FHE. Concretely, the adversary chooses two messages m_0, m_1 and, after receiving the challenge encryption of m_b, it can unmask m_b inside the FHE and compare it with m_0, m_1. The resulting evaluated ciphertext contains a single bit which is much smaller than the original ciphertext. After receiving the secret key, it can decrypt b and break the IND-CPA incompressible security of the scheme.

[GWZ23] provide a simple incompressible SKE in the ROM (instead of the ICM) that can be used to make the same point in the ROM.

5.3 Incompressible Symmetric-Key Encryption

In this section, we define incompressible symmetric-key encryption (SKE) and give a construction from entropic encodings.

5.3.1 Definition

First, we recall the notion of forward-secure storage [Dzi06b] under the name of incompressible symmetric-key encryption. For our purposes we only need IND-EAV style security but this could be extended similar to what we did with incompressible public-key encryption.

Definition 5.3.1 (Incompressible SKE)**.** An incompressible symmetric-key encryption scheme is a tuple of PPT algorithms using uniformly random keys k

$c \leftarrow \mathsf{Enc}(\mathsf{k}, \mathsf{m})$**:** Given a symmetric key k and a message m encryption it outputs a ciphertext c.

$\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$**:** Given a symmetric key k and a ciphertext c decryption it outputs a message m.

We require size of message space, size of key space, and size of ciphertext space to be polynomials over the security parameter λ and the space bound S; that is, $n = n(\lambda, S)$, $k = k(\lambda, S)$, and $l = l(\lambda, S)$ respectively.

Correctness For all $\lambda, S \in \mathbb{N}$, messages m and keys $\mathsf{k} \in \{0,1\}^k$ we have that $\mathsf{m} = \mathsf{Dec}(\mathsf{k}, \mathsf{Enc}(\mathsf{k}, \mathsf{m}))$

Security For security parameter λ and space bound S, a symmetric-key encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ has incompressible SKE security if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ the probability of winning the following experiment is $\leq \frac{1}{2} + \mathsf{negl}[\lambda]$.

$\mathsf{Dist}^{\mathsf{IncomSKE}}_{\mathcal{A},\Pi}(\lambda, S)$ **Experiment** :

- Run the adversary $(\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1) \leftarrow \mathcal{A}_1(1^\lambda)$ to receive two messages m_0 and m_1
- Sample a bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random
- Sample $\mathsf{k} \xleftarrow{\$} \{0,1\}^{n(\lambda,S)}$ uniformly at random
- Run $c \leftarrow \mathsf{Enc}(\mathsf{k}, \mathsf{m}_b)$ to encrypt m_b
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{st}_1, c)$ to produce a state st_2 smaller than S
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{k}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$
- The adversary wins if $b = b'$

5.3.2 Construction

Now we show how to build incompressible symmetric-key encryption using HILL-entropic encodings, extractors, and pseudorandom generators.

Construction 5.3.2. Let λ be the security parameter, S be the space bound of the adversary and n be the size of the message space. Let $(\mathsf{En}, \mathsf{De})$ be an (α, β)-HILL-entropic encoding, $\mathsf{Ext} : \{0,1\}^{\alpha(\lambda,n)} \times \{0,1\}^{d(\lambda)} \to \{0,1\}^\lambda$ be a $(\beta(\lambda, n) - S, \mathsf{negl}[\lambda])$ strong average-case min-entropy extractor where $d(\lambda)$ is a polynomial and $G : \{0,1\}^\lambda \to \{0,1\}^n$ be a PRG.

$\mathsf{Enc}(\mathsf{k}, \mathsf{m})$**:**

- Parse $\mathsf{k} = (\mathsf{k}_1, \mathsf{k}_2, \mathsf{crs})$.
- Sample $\mathsf{s} \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random.
- Let $c_1 \leftarrow \mathsf{En}_{\mathsf{crs}}(1^\lambda, G(\mathsf{s}) \oplus \mathsf{m})$.
- Let $c_2 \leftarrow \mathsf{s} \oplus \mathsf{Ext}(c_1, \mathsf{k}_1) \oplus \mathsf{k}_2$.
- Return $c = (c_1, c_2)$.

$\mathsf{Dec}(\mathsf{k}, c)$**:**

- Parse $\mathsf{k} = (\mathsf{k}_1, \mathsf{k}_2, \mathsf{crs})$.
- Parse $c = (c_1, c_2)$.
- Let $\mathsf{s} \leftarrow \mathsf{Ext}(c_1, \mathsf{k}_1) \oplus c_2 \oplus \mathsf{k}_2$.
- Return $\mathsf{De}_{\mathsf{crs}}(c_1) \oplus G(\mathsf{s})$.

Parameters. The ciphertexts are of size $\lambda + \alpha(\lambda, n)$. The keys are of size $d(\lambda) + t(\lambda, n)$, where $t(\lambda, n)$ is the size of the encoding's crs. Notice that the extractor exists if $\beta(\lambda, n) - S - 2\log\left(\frac{1}{\mathsf{negl}[\lambda]} + 2\right) \geq \lambda$ according to Lemma 2.2.4. So, the adversary is allowed a leakage of size $S \leq \beta(\lambda, n) - \lambda - 2\log\left(\frac{1}{\mathsf{negl}[\lambda]} + 2\right)$.

Therefore, if we choose a "good" entropic encoding we get a rate of

$$\frac{n}{n(1 + o(1)) + \mathsf{poly}[\lambda]},$$

allowed leakage of $S = n(1 - o(1)) - \mathsf{poly}[\lambda]$, and keysize of $k = n(1 + o(1)) + \mathsf{poly}[\lambda]$.

Correctness. By the correctness of the entropic encoding

$$\mathsf{De}_{\mathsf{crs}}(\mathsf{En}_{\mathsf{crs}}(1^\lambda, G(\mathsf{s}) \oplus \mathsf{m})) = G(\mathsf{s}) \oplus \mathsf{m}.$$

Since Ext is deterministic under a fixed key k_1 then

$$\mathsf{Ext}(c_1, \mathsf{k}_1) \oplus c_2 \oplus \mathsf{k}_2 = \mathsf{Ext}(c_1, \mathsf{k}_1) \oplus \mathsf{s} \oplus \mathsf{Ext}(c_1, \mathsf{k}_1) = \mathsf{s}.$$

Therefore, $\mathsf{De}_{\mathsf{crs}}(c_1) \oplus G(\mathsf{s}) = \mathsf{m}$.

Theorem 5.3.3 (Security)**.** *The incompressible SKE presented in Construction 5.3.2 has incompressible SKE security if (*En,De*) is an* (α, β)*-HILL-entropic encoding,* Ext *is a* $(\beta(\lambda, n) - S, \mathsf{negl}[\lambda])$ *strong average-case min-entropy extractor, and* G *is a pseudorandom generator each with the listed parameters.*

Proof. We prove security via hybrids. First we list the hybrid and then argue their indistinguishability. In each hybrid we highlight the changes compared to the previous one.

H_0 :

- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(1^\lambda)$ to receive two messages m_0 and m_1.
- Sample bit $b \xleftarrow{\$} \{0, 1\}$ uniformly at random.
- Sample $\mathsf{k} \xleftarrow{\$} \{0, 1\}^n$ uniformly at random.
- Run $c \leftarrow \mathsf{Enc}(\mathsf{k}, \mathsf{m}_b)$ to encrypt m_b.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{st}_1, c)$ to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{k}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

In H_1 we explicitly represent what happens in Enc.

H_1 :

- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(1^\lambda)$ to receive two messages m_0 and m_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Sample $\mathsf{k}_1 \xleftarrow{\$} \{0,1\}^{d(\lambda,n)}$ uniformly at random.
- Sample $\mathsf{k}_2 \xleftarrow{\$} \{0,1\}^{\lambda}$ uniformly at random.
- Sample $\mathsf{crs} \xleftarrow{\$} \{0,1\}^{t(\lambda,n)}$ uniformly at random.
- Sample $\mathsf{s} \xleftarrow{\$} \{0,1\}^{\lambda}$ uniformly at random.
- Let $c_1 \leftarrow \mathsf{En}_{\mathsf{crs}}(1^\lambda, G(\mathsf{s}) \oplus \mathsf{m}_b)$.
- Let $c_2 \leftarrow \mathsf{s} \oplus \mathsf{Ext}(c_1, \mathsf{k}_1) \oplus \mathsf{k}_2$.
- Let $c \leftarrow (c_1, c_2)$ and $\mathsf{k} \leftarrow (\mathsf{k}_1, \mathsf{k}_2, \mathsf{crs})$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{st}_1, c)$ to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{k}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$

In H_2 we switch the entropic encoding to the simulated code that has a lot of entropy.

H_2 :

- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(1^\lambda)$ to receive two messages m_0 and m_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Sample $\mathsf{k}_1 \xleftarrow{\$} \{0,1\}^{d(\lambda,n)}$ uniformly at random.
- Sample $\mathsf{k}_2 \xleftarrow{\$} \{0,1\}^{\lambda}$ uniformly at random.
-
- Sample $\mathsf{s} \xleftarrow{\$} \{0,1\}^{\lambda}$ uniformly at random.
- Let $(\mathsf{crs}, c_1) \leftarrow \mathsf{SimEn}(1^\lambda, G(\mathsf{s}) \oplus \mathsf{m}_b)$.
- Let $c_2 \leftarrow \mathsf{s} \oplus \mathsf{Ext}(c_1, \mathsf{k}_1) \oplus \mathsf{k}_2$.
- Let $c \leftarrow (c_1, c_2)$ and $\mathsf{k} \leftarrow (\mathsf{k}_1, \mathsf{k}_2, \mathsf{crs})$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{st}_1, c)$ to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{k}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

I H_3 we switch the order in which we sample c_2 and k_2.

H_3 :

- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(1^\lambda)$ to receive two messages m_0 and m_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.

- Sample $\mathsf{k}_1 \xleftarrow{\$} \{0,1\}^{d(\lambda,n)}$ uniformly at random.
-
- Sample $\mathsf{s} \xleftarrow{\$} \{0,1\}^{\lambda}$ uniformly at random.
- Let $(\mathsf{crs}, c_1) \leftarrow \mathsf{SimEn}(1^{\lambda}, G(\mathsf{s}) \oplus \mathsf{m}_b)$.
- Sample $c_2 \xleftarrow{\$} \{0,1\}^{\lambda}$ uniformly at random.
- Let $c \leftarrow (c_1, c_2)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{st}_1, c)$ to produce a state st_2 smaller than S.
- Let $\mathsf{k}_2 \leftarrow c_2 \oplus \mathsf{Ext}(c_1, \mathsf{k}_1) \oplus \mathsf{s}$.
- Let $\mathsf{k} \leftarrow (\mathsf{k}_1, \mathsf{k}_2, \mathsf{crs})$.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{k}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

In H_4 we replace the output of the extractor Ext by a uniformly random value.

H_4 :

- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(1^{\lambda})$ to receive two messages m_0 and m_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Sample $\mathsf{k}_1 \xleftarrow{\$} \{0,1\}^{d(\lambda,n)}$ uniformly at random.
- Sample $\mathsf{s} \xleftarrow{\$} \{0,1\}^{\lambda}$ uniformly at random.
- Let $(\mathsf{crs}, c_1) \leftarrow \mathsf{SimEn}(1^{\lambda}, G(\mathsf{s}) \oplus \mathsf{m}_b)$.
- Sample $c_2 \xleftarrow{\$} \{0,1\}^{\lambda}$ uniformly at random.
- Let $c \leftarrow (c_1, c_2)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{st}_1, c)$ to produce a state st_2 smaller than S.
- Sample $\mathsf{k}_2 \xleftarrow{\$} \{0,1\}^{\lambda}$ uniformly at random.
- Let $\mathsf{k} \leftarrow (\mathsf{k}_1, \mathsf{k}_2, \mathsf{crs})$.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{k}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

Finally we replace the output of $G(\mathsf{s})$ by a uniformly random value.

H_5 :

- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(1^{\lambda})$ to receive two messages m_0 and m_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Sample $\mathsf{k}_1 \xleftarrow{\$} \{0,1\}^{d(\lambda,n)}$ uniformly at random.
- Sample $\mathsf{s} \xleftarrow{\$} \{0,1\}^{\lambda}$ uniformly at random.

- Sample $r \xleftarrow{\$} \{0,1\}^n$ uniformly at random.
- Let $(\mathsf{crs}, c_1) \leftarrow \mathsf{SimEn}(1^\lambda, r)$.
- Sample $c_2 \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random.
- Let $c \leftarrow (c_1, c_2)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{st}_1, c)$ to produce a state st_2 smaller than S.
- Sample $\mathsf{k}_2 \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random.
- Let $\mathsf{k} \leftarrow (\mathsf{k}_1, \mathsf{k}_2, \mathsf{crs})$.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{k}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

$H_0 \approx H_1$:
The differences between H_0 and H_1 are purely syntactical. In H_1 we just show more detail of Enc.

$H_1 \approx_c H_2$:
Instead of sampling the common random string for the entropic encoding uniformly at random and then encoding $G(\mathsf{s}) \oplus \mathsf{m}$ we simulate both steps using SimEn. Assume there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ that can distinguish the two hybrids H_1 and H_2 with a non-negligible advantage of ϵ. From this we construct a PPT adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ that can break the β-HILL-entropy of $(\mathsf{En}, \mathsf{De})$ with advantage ϵ.

$\mathcal{A}'_1(1^\lambda)$:

- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(1^\lambda)$ to receive two messages m_0 and m_1
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random
- Sample $\mathsf{k}_1 \xleftarrow{\$} \{0,1\}^{d(\lambda,n)}$ uniformly at random
- Sample $\mathsf{s} \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random
- Return $G(\mathsf{s}) \oplus \mathsf{m}_b$

$\mathcal{A}'_2(\mathsf{crs}, c_1)$:

- Let $c_2 \leftarrow \mathsf{s} \oplus \mathsf{Ext}(c_1, \mathsf{k}_1)$
- Let $c \leftarrow (c_1, c_2)$
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{st}_1, c)$ to produce a state st_2 smaller than S
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{k}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$
- Return b'

If $\mathcal{A}$ can distinguish H_1 from H_2 then $\mathcal{A}'$ can distinguish a uniformly random $\mathsf{crs} \xleftarrow{\$} \{0,1\}^{t(\lambda,n)}$ and $c_1 \leftarrow \mathsf{En}(1^\lambda, G(\mathsf{s}) \oplus \mathsf{m}_b)$ from $(\mathsf{crs}, c_1) \leftarrow \mathsf{SimEn}(1^\lambda, G(\mathsf{s}) \oplus \mathsf{m}_b)$ as it perfectly simulates H_2 in the case that $(\mathsf{crs}, c_1) \leftarrow \mathsf{SimEn}(1^\lambda, G(\mathsf{s}) \oplus \mathsf{m}_b)$ and perfectly simulates H_1 in the other case.

$H_2 \approx H_3$:
In H_3 we switch the order in which we sample c_2 and k_2. From the view of the adversary this is statistically identical.

$H_3 \approx_s H_4$:

Let C_1, C_2, CRS, K_1, K_2, and ST_2 denote the random variables for the corresponding values in the experiment and U_λ independent uniform randomness of length λ. By the β-HILL entropy of the entropic encoding we know that $\tilde{H}_\infty(C_1|CRS) \geq \beta$. Using Lemma 2.2.2 we deduce that $\tilde{H}_\infty(C_1|(CRS, K_2, ST_2, C_2)) \geq \beta - 2\lambda - log(S)$. Therefore, the extractor gives us that $(K_1, K_2, CRS, ST_2, U_\lambda)$ is statistically close to $(K_1, K_2, CRS, ST_2, \mathsf{Ext}(C_1, K_1))$ which is exactly the view of $\mathcal{A}_3$.

$H_3 \approx_c H_4$:

In H_4 we encode a uniformly random string instead of $G(\mathsf{s}) \oplus \mathsf{m}_b$. Assume there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ that can distinguish the two hybrids H_3 and H_4 with a non-negligible advantage of ϵ. From this we construct a PPT adversary $\mathcal{A}'$ that can break the pseudorandomness of G with advantage ϵ.

$\mathcal{A}'(r')$:

- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(1^\lambda)$ to receive two messages m_0 and m_1
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random
- Sample $\mathsf{k}_1 \xleftarrow{\$} \{0,1\}^{d(\lambda,n)}$ uniformly at random
- Sample $\mathsf{s} \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random
- Sample $r \leftarrow r' \oplus \mathsf{m}_b$ uniformly at random
- Let $(\mathsf{crs}, c_1) \leftarrow \mathsf{SimEn}(1^\lambda, r)$
- Sample $c_2 \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random
- Let $c \leftarrow (c_1, c_2)$
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{st}_1, c)$ to produce a state st_2 smaller than S
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{k}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$
- Return b'

If $\mathcal{A}$ can distinguish H_3 from H_4 then $\mathcal{A}'$ can distinguish $G(\mathsf{s})$ with uniformly random $\mathsf{s} \xleftarrow{\$} \{0,1\}^\lambda$ from uniformly random $r' \xleftarrow{\$} \{0,1\}^n$ as it perfectly simulates H_3 in the case that $r' \leftarrow G(\mathsf{s})$ and perfectly simulates H_4 in the other case.

H_4 :

In H_4 the winning probability of the adversary is $\frac{1}{2}$ as it gets no information about b at all.

□

5.4 Programmable Hash Proof Systems

In this work we think of a hash proof systems as a key encapsulation mechanism where the encapsulated key is independent of the public key and the ciphertext under certain conditions. This allows us to later resample the secret key in the incompressibility experiments.

For our construction we need two different hash proof systems. One that we call Y-programmable and one that we call 2-smooth both using the same language.

5.4.1 Definitions

First we define hash proof system that we will use as a mask in our encryption scheme.

Definition 5.4.1 (Y-Programmable Hash Proof System [CS02, Kal05])**.** A Y-programmable hash proof system is defined over a NP language $\mathcal{L} \subset X$, where each element x in the language $\mathcal{L}$ has a witness w. Additionally there exist a subset $Y \subset X \setminus \mathcal{L}$ and efficient ways to sample a language $\mathcal{L}$ with a corresponding trapdoor $\mathsf{td}_{\mathcal{L}}$, an $x \in \mathcal{L}$ with its witness w and an $x \in Y$ with a corresponding trapdoor td_x

- $(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^k)$: Given the security parameter λ, the encapsulated key size k the language generation algorithm that outputs public parameters pp defining a language $\mathcal{L}$ and a trapdoor $\mathsf{td}_{\mathcal{L}}$ to that language.
- $(x \in \mathcal{L}, w) \leftarrow \mathsf{samp}\mathcal{L}(\mathsf{pp})$: Given the public parameters, it outputs an element $x \in \mathcal{L}$ with the corresponding witness w.
- $(x \in Y, \mathsf{td}_x) \leftarrow \mathsf{samp}Y(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$: Given the public parameters and a trapdoor $\mathsf{td}_{\mathcal{L}}$, it outputs $x \in Y$ and the corresponding trapdoor td_x.

The hash proof system itself consists of these algorithms:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$: Given the public parameters, the key generation algorithm outputs a public key pk and a secret key sk.
- $\mathsf{k} \leftarrow \mathsf{Encap}(\mathsf{pk}, x, w)$: Given the public lye pk, en element x and a witness w. the key encapsulation algorithm outputs an encapsulated key k.
- $\mathsf{k} \leftarrow \mathsf{Decap}(\mathsf{sk}, x)$: Given the secret key sk and any $x \in X$, the key decapsulation algorithm outputs an encapsulated key. k. Notice x can be outside $\mathcal{L}$.
- $\mathsf{sk}' \leftarrow \mathsf{Program}(\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x, \mathsf{sk}, x, \mathsf{k})$ Given two trapdoors $\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x$, a secret key sk, an element $x \in Y$, and an encapsulated key k, the programming algorithm outputs a new secret key sk'.

Correctness. For all $\lambda, k \in \mathbb{N}$, $(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$ in the range of $\mathsf{Gen}(1^\lambda, 1^k)$, $(\mathsf{pk}, \mathsf{sk})$ in the range of $\mathsf{KeyGen}(\mathsf{pp})$, $x \in \mathcal{L}$ and for $\mathsf{k} \leftarrow \mathsf{Encap}(\mathsf{pk}, x, w)$, we have $\mathsf{k} = \mathsf{Decap}(\mathsf{sk}, x)$ with $|\mathsf{k}| = k$.

Language Indistinguishability. For all $\lambda, k \in \mathbb{N}$ if we sample $(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^k)$, $\mathcal{L} \ni x \leftarrow \mathsf{samp}\mathcal{L}(\mathsf{pp})$, and $(x^* \in Y, \mathsf{td}_{x^*}) \leftarrow \mathsf{samp}Y(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$, we have the computational indistinguishability: $x \approx_c x^*$.

Programmability. For all $\lambda, k \in \mathbb{N}$, $(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$ in the range of $\mathsf{samp}\mathcal{L}(1^\lambda, 1^k)$, $(\mathsf{pk}, \mathsf{sk})$ in the range of $\mathsf{KeyGen}(\mathsf{pp})$, $\mathsf{k} \in \{0,1\}^m$, and for (x, td_x) in the range of $\mathsf{samp}Y(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$, $\mathsf{sk}' \leftarrow \mathsf{Program}(\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x, \mathsf{sk}, x, \mathsf{k})$, we have $\mathsf{Decap}(\mathsf{sk}', x) = \mathsf{k}$.

Y-Programmable Smoothness. For all $\lambda, k \in \mathbb{N}$, $(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$ in the range of $\mathsf{Gen}(1^\lambda, 1^k)$, $(\mathsf{pk}, \mathsf{sk})$ in the range of $\mathsf{KeyGen}(\mathsf{pp})$, (x, td_x) in the range of $\mathsf{samp}Y(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$, $\mathsf{k} \in \{0,1\}^m$, and $\mathsf{sk}' \leftarrow \mathsf{Program}(\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x, \mathsf{sk}, x, \mathsf{k})$ we have statistical indistinguishability $(\mathsf{pk}, \mathsf{sk}, x) \approx_s (\mathsf{pk}, \mathsf{sk}', x)$.

Notice, if $Y = X \setminus \mathcal{L}$ then Y-programmable smoothness implies smoothness.

Next we recall 2-smooth hash proof systems with our adjusted notation.

Definition 5.4.2 (2-Smooth Hash Proof System [CS02, ABP15])**.** A 2-smooth hash proof system is defined over a NP language $\mathcal{L} \subset X$ as above The hash proof system itself consists of the following algorithms:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$: Given the public parameters, the key generation algorithm that outputs a public key pk and a secret key sk.
- $\mathsf{k} \leftarrow \mathsf{Encap}(\mathsf{pk}, x, w, \tau)$: Given public key pk, an element of the language $x \in \mathcal{L}$, its witness w, and a tag τ, the key encapsulation algorithm outputs an encapsulated key k.
- $\mathsf{k} \leftarrow \mathsf{Decap}(\mathsf{sk}, x, \tau)$: Given the secret key sk, any $x \in X$, and a tag τ. the key decapsulation algorithm outputs an encapsulated key k. Notice x can be outside $\mathcal{L}$.

Correctness. For all $\lambda, k \in \mathbb{N}$, $(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$ in the range of $\mathsf{Gen}(1^\lambda, 1^k)$, $(\mathsf{pk}, \mathsf{sk})$ in the range of $\mathsf{KeyGen}(\mathsf{pp})$, $x \in \mathcal{L}$, tags τ, and for $\mathsf{k} \leftarrow \mathsf{Encap}(\mathsf{pk}, x, w, \tau)$, we have $\mathsf{k} = \mathsf{Decap}(\mathsf{sk}, x, \tau)$ with $|\mathsf{k}| = k$.

Language Indistinguishability. Exactly as above.

2-Smoothness. For all $\lambda, k \in \mathbb{N}$, $(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$ in the range of $\mathsf{Gen}(1^\lambda, 1^k)$, $x, x' \in X \setminus \mathcal{L}$, two tags τ, τ' such that $(x, \tau) \neq (x', \tau')$, let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ and sample $\mathsf{k} \xleftarrow{\$} \{0,1\}^k$ we have computational indistinguishability between $(\mathsf{pk}, \mathsf{Decap}(\mathsf{sk}, x, \tau), \mathsf{Decap}(\mathsf{sk}, x', \tau'))$ and $(\mathsf{pk}, \mathsf{Decap}(\mathsf{sk}, x, \tau), \mathsf{k})$.

5.4.2 Programmable Hash Proof System from DDH

In our protocols we need programmable HPS with a big encapsulated key space (for classic notation [CS02] this would be called the hash space).

Some smooth hash proof systems are easily transformed into programmable HPS with big encapsulated keys by generating more public keys and using them on the same $x \in X$. These HPS include the one from weak pseudorandom effective group actions [ADMP20]. That transformation causes the public key size to scale linearly with the size of the encapsulated key and leave the size of the ciphertext indepent of the encapsulated key size.

We present a variant of the original [CS02] HPS with an interesting trade off. Here both public key size and ciphertext size scale in the 2/3-power with k, the size of the encapsulated key.

Construction 5.4.3. Let $\mathsf{HC} : \mathbb{G} \times \{0,1\}^{log(|\mathbb{G}|)} \to \{0,1\}$ denote a 1-bit randomness extractor over a group element; if this function is applied over a matrix of group elements, then it means that the function is applied entry-wise with the same randomness. In the following let $\ell, s \in \mathbb{N}$ such that $\ell \cdot s = k$. We get an interesting tradeoff for our application when $\ell = k^{1/3}$ and $s = k^{2/3}$.

$\mathsf{Gen}(1^\lambda, 1^k)$:

- $(\mathbb{G}, p, g) \xleftarrow{\$} \mathcal{G}(1^\lambda)$.
- Sample $\mathbf{h} \xleftarrow{\$} \mathbb{Z}_p^\ell \setminus \{\mathbf{0}\}$ uniformly at random.

- Return $\mathsf{pp} = (\mathbb{G}, p, g, [\mathbf{h}])$ and $\mathsf{td}_{\mathcal{L}} = \mathbf{h}$.

$\mathsf{samp}\mathcal{L}(\mathsf{pp})$:

- Parse $\mathsf{pp} = (\mathbb{G}, p, g, [\mathbf{h}])$.
- Sample $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^{\ell-1}$ uniformly at random.
- Return $x = [\mathbf{y}]$ and $w = \mathbf{y}$.

$\mathsf{samp}Y(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$:

- Parse $\mathsf{pp} = (\mathbb{G}, p, g, [\mathbf{h}])$.
- Let $\mathsf{td}_{\mathcal{L}} = \mathbf{h}$.
- Sample $\mathbf{E} \xleftarrow{\$} \mathbb{Z}_p^{\ell \times (\ell-1)}$ such that $\begin{pmatrix}\mathbf{h} & \mathbf{E}\end{pmatrix}$ is invertible uniformly at random.
- Return $x = [\mathbf{E}]$ and $w = \mathbf{E}$.

$\mathsf{KeyGen}(\mathsf{pp})$**:**

- Parse $\mathsf{pp} = (\mathbb{G}, p, g, [\mathbf{h}])$.
- Sample $r \xleftarrow{\$} \{0,1\}^{log(|\mathbb{G}|)}$ the public randomness for a extractor.
- Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{s \times \ell}$ uniformly at random.
- Return $\mathsf{pk} = (\mathbf{A}[\mathbf{h}], r)$ and $\mathsf{sk} = \mathbf{A}$.

$\mathsf{Encap}\ \ (\mathsf{pk}, c = [\mathbf{h}\mathbf{y}^t], w = \mathbf{y})$:

- Parse $\mathsf{pp} = (\mathbb{G}, p, g, [\mathbf{h}] \in \mathbb{G}^{\ell})$.
- Parse $\mathsf{pk} = ([\mathbf{f}] \in \mathbb{G}^s, r)$.
- Let $\mathbf{K} \leftarrow \mathsf{HC}([\mathbf{f}]\mathbf{y}^t, r)$ the component-wise extractor of the outer product between $\mathbf{f}$ and $\mathbf{y}$.
- Return $\mathsf{k} = \mathbf{K}$.

$\mathsf{Decap}\ \ (\mathsf{sk}, x = [\mathbf{E}] \in \mathbb{G}^{\ell \times (\ell-1)})$:

- Parse $\mathsf{pk} = ([\mathbf{f}], r)$
- Parse $\mathsf{sk} = \mathbf{A} \in \mathbb{Z}_p^{s \times \ell}$.
- Let $\mathbf{K} \leftarrow \mathsf{HC}(\mathbf{A}[\mathbf{E}], r)$ the component-wise extractor of the product between $\mathbf{A}$ and $[\mathbf{E}]$.
- Return $\mathsf{k} = \mathbf{K}$.

$\mathsf{Program}(\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x, \mathsf{sk}, x, \mathsf{k})$**:**

- Parse $\mathsf{pk} = ([\mathbf{f}], r)$, $\mathsf{td}_{\mathcal{L}} = \mathbf{h} \in \mathbb{Z}_p^{\ell}$, $\mathsf{td}_x = \mathbf{E} \in \mathbb{Z}_p^{\ell \times (\ell-1)}$, $\mathsf{sk} = \mathbf{A}$, and $\mathsf{k} = \mathbf{K} \in \{0,1\}^{s \times (\ell-1)}$.
- For each $i \in [\ell - 1], j \in [s]$ sample $B_{i,j} \xleftarrow{\$} \mathbb{Z}_p$ such that $K_{i,j} = \mathsf{HC}([B_{i,j}], r)$ via rejection sampling.
- Set $\mathbf{B} = (B)_{i,j}$. Let $\mathbf{A}' \leftarrow \begin{pmatrix}\mathbf{A}\mathbf{h} & \mathbf{B}\end{pmatrix}\begin{pmatrix}\mathbf{h} & \mathbf{E}\end{pmatrix}^{-1}$.
- Return $\mathsf{sk}' = \mathbf{A}'$.

Correctness. For any $(\mathsf{pp} = (\mathbb{G}, p, g, [\mathbf{h}]), \mathsf{td}_{\mathcal{L}})$ in the range of Gen, $(\mathsf{pk} = ([\mathbf{Ah}], r), \mathsf{sk} = \mathbf{A})$ in the range of KeyGen, and $[\mathbf{hy}^t] \in \mathcal{L}$ we have $\mathsf{Encap}(\mathsf{pk}, [\mathbf{hy}^t])$ outputs $\mathsf{k} = \mathsf{HC}([\mathbf{Ah}]\mathbf{y}^{\mathbf{t}}, r) = \mathsf{HC}([\mathbf{Ahy}^t], r)$. Decapsulation then outputs $\mathsf{k} = \mathsf{HC}(\mathbf{A}[\mathbf{hy}^t], r) = \mathsf{HC}([\mathbf{Ahy}^t], r)$.

Programmability. Since we choose $\mathbf{h}$ and $\mathbf{E}$ s.t. $\begin{pmatrix}\mathbf{h} & \mathbf{E}\end{pmatrix}$ is invertible $\mathsf{Program}$ always outputs a matrix $\mathbf{A}'$ with the property that $\mathbf{A}'\mathbf{E} = \mathbf{B}$ and $\mathsf{k} = \mathsf{HC}([\mathbf{B}], r)$.

Programmable Smoothness. If we first sample k uniformly random and then program for the key k $\mathsf{Program}(\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x, \mathsf{sk}, x, \mathsf{k})$ the resulting distribution over $\mathbf{B}$ will be uniformly random. And because $\begin{pmatrix}\mathbf{h} & \mathbf{E}\end{pmatrix}$ is invertible then $\mathbf{A}'$ is a uniformly random under the condition that $\mathbf{A}'\mathbf{h} = \mathbf{Ah}$. The same holds for $\mathbf{A}$. Therefore, $(\mathsf{pk}, \mathsf{sk} = \mathbf{A}, x)$ and $(\mathsf{pk}, \mathsf{sk}' = \mathbf{A}', x)$ are identically distributed.

Theorem 5.4.4 (Language Indistinguishability)**.** *If DDH is hard for $\mathcal{G}$ then elements from $\mathcal{L} = \{[\mathbf{h}]\mathbf{y}^t | \mathbf{y} \in \mathbb{Z}_p^{\ell-1}\}$ and*

$$Y = \{[\mathbf{E}] | \mathbf{E} \in \mathbb{Z}_p^{\ell \times (\ell-1)} \wedge \begin{pmatrix}\mathbf{h} & \mathbf{E}\end{pmatrix} \text{ is invertible}\}$$

of construction 5.4.3 are indistinguishable.

Proof. We prove security via hybrids. First we list the hybrids and then argue their indistinguishability. In each hybrid we highlight the changes compared to the previous one.

H_0 :

- Let $(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^k)$.
- Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$.
- Let $(x, w) \leftarrow \mathsf{samp}\mathcal{L}(\mathsf{pp})$.
- Let $\mathsf{k} \leftarrow \mathsf{Encap}(\mathsf{pk}, x, w)$.
- Run the adversary $\mathcal{A}(\mathsf{pk}, \mathsf{sk}, x)$.

H_1 :

- Sample a group $(\mathbb{G}, p, g) \xleftarrow{\$} \mathcal{G}(1^\lambda)$.
- Sample $r \xleftarrow{\$} \{0,1\}^{log(|\mathbb{G}|)}$ the randomness for the extractor .
- Sample $\mathbf{h} \xleftarrow{\$} \mathbb{Z}_p^\ell \setminus \{\mathbf{0}\}$ uniformly at random .
- Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{s \times \ell}$ uniformly at random .
- Let $\mathsf{pp} = (\mathbb{G}, p, g, [\mathbf{h}])$.
- Let $\mathsf{pk} = ([\mathbf{Ah}], r)$ and $\mathsf{sk} = \mathbf{A}$.
- Sample $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^{\ell-1}$ uniformly at random .
- Let $x = [\mathbf{hy}^t] = [\mathbf{C}]$.
- Run the adversary $\mathcal{A}(\mathsf{pk}, \mathsf{sk}, x)$.

$H_{2,i}$:

- Sample a group $(\mathbb{G}, p, g) \xleftarrow{\$} \mathcal{G}(1^\lambda)$.
- Sample $r \xleftarrow{\$} \{0,1\}^{\log(|\mathbb{G}|)}$ the randomness for the extractor.
- Sample $\mathbf{h} \xleftarrow{\$} \mathbb{Z}_p^\ell \setminus \{\mathbf{0}\}$ uniformly at random.
- Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{s\times\ell}$ uniformly at random.
- Let $\mathsf{pp} = (\mathbb{G}, p, g, [\mathbf{h}])$.
- Let $\mathsf{pk} = ([\mathbf{Ah}], r)$ and $\mathsf{sk} = \mathbf{A}$.
- Sample $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^{\ell-1}$ uniformly at random.
- Let $[\mathbf{C}] = [\mathbf{h}\mathbf{y}^t]$.
- Sample $\mathbf{E} \xleftarrow{\$} \mathbb{Z}_p^{l\times(l-1)}$ uniformly at random .
- Replace the first i entries of $[\mathbf{C}]$ by the first i entries in $[\mathbf{E}]$.
- Let $x = [\mathbf{C}]$.
- Run the adversary $\mathcal{A}(\mathsf{pk}, \mathsf{sk}, x)$.

H_3 :

- Sample a group $(\mathbb{G}, p, g) \xleftarrow{\$} \mathcal{G}(1^\lambda)$.
- Sample $r \xleftarrow{\$} \{0,1\}^{\log(|\mathbb{G}|)}$ the randomness for the extractor.
- Sample $\mathbf{h} \xleftarrow{\$} \mathbb{Z}_p^\ell \setminus \{\mathbf{0}\}$ uniformly at random.
- Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{s\times\ell}$ uniformly at random.
- Let $\mathsf{pp} = (\mathbb{G}, p, g, [\mathbf{h}])$.
- Let $\mathsf{pk} = ([\mathbf{Ah}], r)$ and $\mathsf{sk} = \mathbf{A}$.
- Sample $\mathbf{E} \xleftarrow{\$} \mathbb{Z}_p^{l\times(l-1)}$ uniformly at random such that $\begin{pmatrix}\mathbf{h} & \mathbf{E}\end{pmatrix}$ is invertible .
- Let $x = [\mathbf{E}]$.
- Run the adversary $\mathcal{A}(\mathsf{pk}, \mathsf{sk}, x)$.

$H_0 \approx H_1$:
The differences between H_0 and H_1 are purely syntactical. In H_1 we just show more detail of Gen and Encap.

$H_1 \approx H_{2,0}$:
The differences between H_1 and $H_{2,0}$ are purely syntactical.

$H_{2,i} \approx_c H_{2,i+1}$:
In $H_{2,i+1}$ we replace the $n+1$st element of $\mathbf{C}$ by a random one. Assume there exists a PPT adversary $\mathcal{A}$ that can distinguish the two hybrids $H_{2,i}$ and $H_{2,i+1}$ with a non-negligible advantage of ϵ. From this we construct a PPT adversary $\mathcal{A}'$ that can break DDH with advantage ϵ.

$\mathcal{A}'$ $((\mathbb{G}, p, g), ([a], [b], [\rho]))$:

- Let $u \leftarrow i \mod l$
- Let $v \leftarrow \lfloor i/l \rfloor$
- Sample $r \xleftarrow{\$} \{0,1\}^{\log(|\mathbb{G}|)}$ the randomness for the extractor
- Sample $\mathbf{h} \xleftarrow{\$} \mathbb{Z}_p^\ell \setminus \{\mathbf{0}\}$ uniformly at random
- Replace $[x_u]$ by $[a]$
- Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{s \times \ell}$ uniformly at random
- Let $\mathsf{pp} = (\mathbb{G}, p, g, [\mathbf{h}])$
- Let $\mathsf{pk} = ([\mathbf{Ah}], r)$ and $\mathsf{sk} = \mathbf{A}$
- Sample $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^{\ell-1}$ uniformly at random
- For $u' \in [l]$ and $v' \in [l-1]$:

  Let $C_{u',v'} \leftarrow \begin{cases} [\rho] & \text{if } u' = u, v' = v \\ [b]x_{u'} & \text{if } u' \neq u, v' = v \\ [x_{u'}]y_{v'} & \text{else} \end{cases}$
- Sample $\mathbf{E} \xleftarrow{\$} \mathbb{Z}_p^{l \times (l-1)}$
- Replace the first i entries of $[\mathbf{C}]$ by the first i entries in $[\mathbf{E}]$
- Let $x = [\mathbf{C}]$
- Run the adversary $b' \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk}, x)$
- Return b'

If $\mathcal{A}$ distinguishes between $H_{2,i}$ and $H_{2,i+1}$ then $\mathcal{A}'$ distinguishes between $\rho = ab$ and ρ being uniformly random as $\mathcal{A}'$ perfectly simulates $H_{2,i}$ in the case that $\rho = ab$ and $H_{2,i+1}$ if r is uniformly random.

$H_{2,m} \approx_s H_3$:
$H_{2,m}$ is statistically close to H_3 because with probability $1 - \mathsf{negl}[\lambda]$ we have $\begin{pmatrix} \mathbf{h} & \mathbf{E} \end{pmatrix}$ is invertible.

□

Parameters. For an encapsulated key of size k this scheme roughly gets us public parameters of size $k^{1/3} \cdot \mathsf{poly}[\lambda]$, public key of size $k^{2/3} \cdot \mathsf{poly}[\lambda]$ and elements from X of size $k^{2/3} \cdot \mathsf{poly}[\lambda]$.

5.4.3 2-Smooth Hash Proof System from DDH

The above hash proof system only is programmable if $x \in Y$. To make our encryption scheme CCA secure we need a efficient way to check whether $x \in \mathcal{L}$ or $x \in X \setminus \mathcal{L}$. To do this we construct the 2-smooth hash proof system below that is defined over the same language.

Construction 5.4.5. We construct a 2-smooth hash proof system with a output size of λ using an extractor $\mathsf{Ext} : \mathbb{G}^{\ell-1} \times \{0,1\}^p \to \{0,1\}^\lambda$ and a collision resistant hash function CRHF that maps into $\mathbb{Z}_p$. As a language description we use the same as in Construction 5.4.3.

$\mathsf{KeyGen}(\mathsf{pp})$**:**

- Parse $\mathsf{pp} = (\mathbb{G}, p, g, [\mathbf{h}])$.

- Sample $r \xleftarrow{\$} \{0,1\}^{log(|\mathbb{G}|)}$ uniformly at random.
- Sample $\mathsf{s} \xleftarrow{\$} \{0,1\}^{\lambda}$ uniformly at random.
- Sample $\mathbf{a}, \mathbf{b} \xleftarrow{\$} \mathbb{Z}_p^{\ell}$ uniformly at random.
- Return $\mathsf{pk} = (\mathbf{a}^t[\mathbf{h}], \mathbf{b}^t[\mathbf{h}], r, \mathsf{s})$ and $\mathsf{sk} = (\mathbf{a}, \mathbf{b})$.

Encap $(\mathsf{pk}, x = [\mathbf{h}\mathbf{y}^t] \in \mathbb{G}^{\ell \times (\ell-1)}, w = \mathbf{y} \in \mathbb{Z}_p^{\ell-1}, \tau)$:

- Parse $\mathsf{pp} = (\mathbb{G}, p, g, [\mathbf{h}] \in \mathbb{G}^{\ell})$ and $\mathsf{pk} = ([f], [f'] \in \mathbb{G}, r, \mathsf{s})$.
- Let $[\mathbf{d}] = ([f]\mathbf{y}) + (\mathsf{CRHF}_{\mathsf{s}}(x, \tau)[f']\mathbf{y})$.
- Return $\mathsf{k} = \mathsf{Ext}([\mathbf{d}^t], r)$.

Decap $(\mathsf{sk}, x = [\mathbf{E}] \in \mathbb{G}^{\ell \times (\ell-1)}, \tau)$:

- Parse $\mathsf{pp} = (\mathbb{G}, p, g, [\mathbf{h}] \in \mathbb{G}^{\ell})$ and $\mathsf{pk} = ([f], [f'] \in \mathbb{G}, r, \mathsf{s})$.
- Parse $\mathsf{sk} = (\mathbf{a} \in \mathbb{Z}_p^{\ell}, \mathbf{b} \in \mathbb{Z}_p^{\ell})$.
- Parse $x = [\mathbf{E}] \in \mathbb{G}^{\ell \times (\ell-1)}$.
- Return $\mathsf{k} = \mathsf{Ext}(\mathbf{a}^t[\mathbf{E}] + \mathsf{CRHF}_{\mathsf{s}}(x, \tau)\mathbf{b}^t[\mathbf{E}], r)$.

Correctness. For any $(\mathsf{pp} = (\mathbb{G}, p, g, [\mathbf{h}]), \mathsf{td}_{\mathcal{L}})$ in the range of Gen, $(\mathsf{pk} = (\mathbf{a}^t[\mathbf{h}], \mathbf{b}^t[\mathbf{h}], r, \mathsf{s})$, $\mathsf{sk} = (\mathbf{a}, \mathbf{b}))$ in the range of KeyGen, and $[\mathbf{h}\mathbf{y}^t] \in \mathcal{L}$ we have $\mathsf{Encap}(\mathsf{pk}, [\mathbf{h}\mathbf{y}^t])$ outputs

$$\begin{aligned}\mathsf{k} =& \mathsf{Ext}\left(\left(([f]\mathbf{y}) + ([f]\mathsf{CRHF}_{\mathsf{s}}(x, \tau)\mathbf{y})\right)^t, r\right) \\ =& \mathsf{Ext}\left([(\mathbf{a}^t\mathbf{h})\mathbf{y}^t + \mathsf{CRHF}_{\mathsf{s}}(x, \tau)(\mathbf{b}^t\mathbf{h})\mathbf{y}^t], r\right).\end{aligned}$$

On the other hand, decapsulation outputs

$$\begin{aligned}\mathsf{k} &= \mathsf{Ext}(\mathbf{a}^t[\mathbf{h}\mathbf{y}^t] + \mathsf{CRHF}_{\mathsf{s}}(x, \tau)\mathbf{b}^t[\mathbf{h}\mathbf{y}^t], r) \\ &= \mathsf{Ext}\left([(\mathbf{a}^t\mathbf{h})\mathbf{y}^t + \mathsf{CRHF}_{\mathsf{s}}(x, \tau)(\mathbf{b}^t\mathbf{h})\mathbf{y}^t], r\right).\end{aligned}$$

Language Indistinguishability. Since we use the same language as in construction 5.4.3 the language indistinguishability holds by the same argument.

2-Smoothness. For all $\lambda, n \in \mathbb{N}$, $(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$ in the range of $\mathsf{Gen}(\mathsf{pp})$, $x, x' \in X \setminus \mathcal{L}$, two tags τ, τ' such that $(x, \tau) \neq (x', \tau')$, let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ and sample $\mathsf{k} \xleftarrow{\$} \{0,1\}^m$. Let $\gamma \leftarrow \mathsf{CRHF}_{\mathsf{s}}(x, \tau)$ and $\gamma' \leftarrow \mathsf{CRHF}_{\mathsf{s}}(x', \tau')$. Using $(x, \tau) \neq (x', \tau')$ and the collision resistance of CRHF we can assume that $\gamma \neq \gamma'$.

In the following let $\mathbf{d} = \mathbf{a}^t[\mathbf{E}] + \gamma\mathbf{b}^t[\mathbf{E}]$ (computed in $\mathsf{Decap}(\mathsf{sk}, x, \tau)$) and $\mathbf{d'} = \mathbf{a}^t[\mathbf{E}'] + \gamma'\mathbf{b}^t[\mathbf{E}']$ (computed in $\mathsf{Decap}(\mathsf{sk}, x', \tau')$). Then the following equation holds:

$$\begin{pmatrix} f & f' & \mathbf{d}^t & \mathbf{d}'^t \end{pmatrix} = \begin{pmatrix} \mathbf{a}^t & \mathbf{b}^t \end{pmatrix} \begin{pmatrix} \mathbf{h} & \mathbf{0} & \mathbf{E} & \mathbf{E}' \\ \mathbf{0} & \mathbf{h} & \gamma\mathbf{E} & \gamma'\mathbf{E}' \end{pmatrix}$$

If $x, x' \in X \setminus \mathcal{L}$ then there exists a column $\mathbf{z}$ with index i in $\mathbf{E}$ s.t. $\mathbf{z}$ is linearly independent of $\mathbf{h}$ and $\mathbf{z}'$ with index i' in $\mathbf{E}'$ s.t. $\mathbf{z}'$ is l.i. of $\mathbf{h}$. Then the following equation also holds:

$$\begin{pmatrix} f & f' & d_i & d'_{i'} \end{pmatrix} = \begin{pmatrix} \mathbf{a}^t & \mathbf{b}^t \end{pmatrix} \begin{pmatrix} \mathbf{h} & \mathbf{0} & \mathbf{z} & \mathbf{z}' \\ \mathbf{0} & \mathbf{h} & \gamma\mathbf{z} & \gamma'\mathbf{z}' \end{pmatrix}$$

Now, we argue that the matrix on the right side has rank 4. We have that $\begin{pmatrix}\mathbf{h}\\\mathbf{0}\end{pmatrix}$ and $\begin{pmatrix}\mathbf{0}\\\mathbf{h}\end{pmatrix}$ are linearly independent. Moreover, $\begin{pmatrix}\mathbf{z}\\\gamma\mathbf{z}\end{pmatrix}$ is outside the span of $\begin{pmatrix}\mathbf{h}\\\mathbf{0}\end{pmatrix}$ and $\begin{pmatrix}\mathbf{0}\\\mathbf{h}\end{pmatrix}$ because $\mathbf{h}$ and $\mathbf{z}$ are linearly independent. Finally, $\begin{pmatrix}\mathbf{z'}\\\gamma'\mathbf{z'}\end{pmatrix}$ is outside the span of $\begin{pmatrix}\mathbf{z}\\\gamma\mathbf{z}\end{pmatrix}$, $\begin{pmatrix}\mathbf{h}\\\mathbf{0}\end{pmatrix}$, and $\begin{pmatrix}\mathbf{0}\\\mathbf{h}\end{pmatrix}$. To see this, assume that this is not the case, i.e., that there exists a linear combination

$$\begin{pmatrix}\mathbf{z'}\\\gamma'\mathbf{z'}\end{pmatrix} = c_1\begin{pmatrix}\mathbf{z}\\\gamma\mathbf{z}\end{pmatrix} + c_2\begin{pmatrix}\mathbf{h}\\\mathbf{0}\end{pmatrix} + c_3\begin{pmatrix}\mathbf{0}\\\mathbf{h}\end{pmatrix}.$$

Assume there exist c_1,c_2,$c_3 \in \mathbb{N}^+$ such that $\mathbf{z'} = c_1\mathbf{z} + c_2\mathbf{h}$ and $\gamma'\mathbf{z'} = c_1\gamma\mathbf{z} + c_3\mathbf{h}$. Then we replace $\mathbf{z'}$ in the second equation

$$\begin{aligned}&\gamma'(c_1\mathbf{z} + c_2\mathbf{h}) = c_1\gamma\mathbf{z} + c_3\mathbf{h}\\ \Leftrightarrow\ &(\gamma' - \gamma)c_1\mathbf{z} = (c_3 - \gamma' c_2)\mathbf{h}\end{aligned}$$

This however can only be true if $\gamma' - \gamma = 0$ because $\mathbf{z}$ is linearly independent of $\mathbf{h}$.

Since $\mathbf{a}$ and $\mathbf{b}$ are chosen uniformly at random then so are f,f', d_i, and d'_i. If d_i and d'_i are uniformly random then $\mathsf{Decap}(\mathsf{sk}, x, \tau) = \mathsf{Ext}(\mathbf{d}^t, r)$ and $\mathsf{Decap}(\mathsf{sk}, x', \tau') = \mathsf{Ext}(\mathbf{d'^t}, r)$ are statistically close to uniformly random by the extractor property.

Parameters. For the same language as in Construction 5.4.3 with public parameters of size $k^{1/3} \cdot \mathsf{poly}[\lambda]$ and elements of $k^{2/3} \cdot \mathsf{poly}[\lambda]$ construction 5.4.5 roughly results in public keys of size $2k^{1/3} \cdot \mathsf{poly}[\lambda]$ and an encapsulated key of size λ.

5.5 Incompressible PKE

In this section, we show how to build incompressible PKE from incompressible SKE and a hash proof system with extra properties.

First we extend the incompressible encryption security notion [GWZ22] to the chosen ciphertext scenario and then we show a new construction paradigm using hash proof systems and incompressible symmetric-key encryption.

5.5.1 CCA Incompressible Encryption

We use the definition of incompressible encryption by Guan et al.[GWZ22]. It defines a public-key encryption scheme where the adversary has to know most of the ciphertext to decrypt it even with access to the secret key.

Definition 5.5.1 (Incompressible PKE)**.** An incompressible public-key encryption scheme is a triple of PPT algorithms

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^S)$**:** Given the security parameter λ and a space bound S the key-generation algorithm outputs a public key pk and a secret key sk.

$c \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m})$**:** Given a public key pk and a message m the encryption algorithm outputs a ciphertext c.

$\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$**:** Given a secret key sk and a ciphertext c the decryption algorithm outputs a message m.

Both size of message space and size of ciphertext space are polynomials over security parameter λ and space bound S, that is, $n = n(\lambda, S)$ and $l = l(\lambda, S)$ respectively.

Correctness. For all $\lambda, S \in \mathbb{N}$, messages m and $(\mathsf{pk}, \mathsf{sk})$ in the range of KeyGen we have that $\mathsf{m} = \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{m}))$.

CCA Incompressible Security. Similar to standard IND-CCA (sometimes referred to as IND-CCA2) security we extend incompressible encryption such that the adversary has access to an encryption oracle.

For security parameter λ and space bound S, a public key encryption scheme (KeyGen, Enc, Dec) has incompressible CCA PKE security if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ wins the following experiment with probability $\leq \frac{1}{2} + \mathsf{negl}[\lambda]$.

$\mathsf{Dist}^{\mathsf{CCAIncomPKE}}_{\mathcal{A},\Pi}(\lambda, S)$ **Experiment** :

- Run key generation algorithm $\mathsf{KeyGen}(1^\lambda, 1^S)$ to obtain $(\mathsf{pk}, \mathsf{sk})$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk})$ on public key pk with oracle access to $\mathsf{Dec}(\mathsf{sk}, \cdot)$ to receive two messages m_0, m_1 and state st_1.
- Sample bit $b \xleftarrow{\$} \{0, 1\}$ uniformly at random.
- Run $c \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_b)$ to encrypt m_b.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk}, c, \mathsf{st}_1)$ with access to the decryption oracle $\mathsf{Dec}(\mathsf{sk}, \cdot)$ for all inputs but c to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

Rate We define the rate by $\frac{|\mathsf{m}|}{|\mathsf{Enc}(\mathsf{pk},\mathsf{m})|}$ the size of a message divided by a ciphertext encrypting the message. We say a scheme has rate-1 when the rate is $1 - o(1)$.

5.5.2 Construction

We construct a encryption scheme that very much resembles the classic Cramer-Shoup [CS02] scheme. Instead of masking the ciphertext with the randomness that comes out of the hash proof system we use it as a key for an incompressible symmetric-key encryption scheme.

Construction 5.5.2 (Incompressible PKE)**.** Given security parameter λ, space bound S, and message length n let (KeyGen', Encap', Decap', $\mathsf{Program}'$) be a Y-programmable hash proof system for a language $\mathcal{L} \subset X$ (where you can sample x with according witness from $\mathcal{L}$ and sample x with according trapdoor from Y) where the representation size of X is $p(\lambda, S, n)$ and encapsulated keys of size $k(\lambda, S_{\mathsf{sym}}, n)$, ($\mathsf{KeyGen}''$, Encap'', Decap'') is a 2-smooth hash proof system for the same language with encapsulation key size of λ and public key size $p'(\lambda, S, n)$, and ($\mathsf{Enc}_{\mathsf{sym}}$, $\mathsf{Dec}_{\mathsf{sym}}$) be an incompressible SKE with messages of size n, keys of size $k(\lambda, S_{\mathsf{sym}}, n)$ and ciphertexts of size $l(\lambda, S_{\mathsf{sym}}, n)$ with incompressible SKE adversary being allowed to leak a state of size $S_{\mathsf{sym}} = S + p(\lambda, S, n) + p'(\lambda, S, n)$.

$\mathsf{KeyGen}(1^\lambda, 1^S)$**:**

- Generate language and corresponding trapdoor

$$(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^n).$$

- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.

- Let $(\mathsf{pk}'', \mathsf{sk}'') \leftarrow \mathsf{KeyGen}''(\mathsf{pp})$.
- Return $\mathsf{pk} = (\mathsf{pk}', \mathsf{pk}'')$ and $\mathsf{sk} = (\mathsf{sk}', \mathsf{sk}'')$.

$\mathsf{Enc}(\mathsf{pk}, \mathsf{m})$**:**

- Parse $\mathsf{pk} = (\mathsf{pk}', \mathsf{pk}'')$
- Let $(x, w) \leftarrow \mathsf{samp}\mathcal{L}(\mathsf{pp})$.
- Let $\mathsf{k} \leftarrow \mathsf{Encap}'(\mathsf{pk}', x, w)$.
- Let $c_{\mathsf{sym}} \leftarrow \mathsf{Enc}_{\mathsf{sym}}(\mathsf{k}, \mathsf{m})$.
- Let $\pi \leftarrow \mathsf{Encap}''(\mathsf{pk}'', x, w, c_{\mathsf{sym}})$.
- Return $c = (x, c_{\mathsf{sym}}, \pi)$.

$\mathsf{Dec}(\mathsf{sk}, c)$**:**

- Parse $\mathsf{sk} = (\mathsf{sk}', \mathsf{sk}'')$.
- Parse $c = (x, c_{\mathsf{sym}}, \pi)$.
- If $\pi = \mathsf{Decap}''(\mathsf{sk}'', x, c_{\mathsf{sym}})$
  - Let $\mathsf{k} \leftarrow \mathsf{Decap}'(\mathsf{sk}', x)$
  - Return $\mathsf{m} = \mathsf{Dec}_{\mathsf{sym}}(\mathsf{k}, c_{\mathsf{sym}})$.
- Return $\perp$.

Parameters. $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is an incompressible PKE with messages of size n, ciphertexts of size $l(\lambda, S_{\mathsf{sym}}, n) + p(\lambda, S, n) + p'(\lambda, S, n)$, the adversary is allowed a leak of size $S = S_{\mathsf{sym}} - p(\lambda, S, n) - p'(\lambda, S, n)$, and the public key is of size $p(\lambda, S, n) + p'(\lambda, S, n)$.

When instantiating the two hash proof systems with constructions 5.4.3,5.4.5 and the incompressible SKE with construction 5.3.2 then $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is an incompressible PKE with messages of size n, ciphertexts of size $(n + n^{2/3}\mathsf{poly}[\lambda])(1 + o(1))$, the adversary is allowed a leak of size $S = n(1 - o(1)) - \mathsf{poly}[\lambda](n(1 + o(1)))^{2/3}$, the public key is of size $n^{2/3}(1 + o(1))\mathsf{poly}[\lambda]$, and the secret key is of size $n(1 + o(1))\mathsf{poly}[\lambda]$.

Correctness. Follows from the correctness of the hash proof systems $(\mathsf{KeyGen}', \mathsf{Encap}', \mathsf{Decap}', \mathsf{Program}')$, $(\mathsf{KeyGen}'', \mathsf{Encap}'', \mathsf{Decap}'')$, and symmetric-key encryption $(\mathsf{Enc}_{\mathsf{sym}}, \mathsf{Dec}_{\mathsf{sym}})$.

Theorem 5.5.3 (Security)**.** *The PKE construction 5.5.2 has incompressible CCA PKE security if* $(\mathsf{KeyGen}', \mathsf{Encap}', \mathsf{Decap}', \mathsf{Program}')$ *is a programmable hash proof system with the listed parameters,* $(\mathsf{KeyGen}'', \mathsf{Encap}'', \mathsf{Decap}'')$ *is a 2-smooth hash proof system with the listed parameters, and* $(\mathsf{Enc}_{\mathsf{sym}}, \mathsf{Dec}_{\mathsf{sym}})$ *is an incompressible secure SKE with the listed parameters.*

Proof. We prove security via hybrids. First we list the hybrids and then argue their indistinguishability. In each hybrid we highlight the changes compared to the previous one.

$H_0(\lambda, S)$:

- Run key generation algorithm $\mathsf{KeyGen}(1^\lambda, 1^S)$ to obtain $(\mathsf{pk}, \mathsf{sk})$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk})$ on public key pk with oracle access to $\mathsf{Dec}(\mathsf{sk}, \cdot)$ to receive two messages m_0, m_1 and state st_1.

- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Run $c \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_b)$ to encrypt m_b.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk}, c, \mathsf{st}_1)$ with access to the decryption oracle $\mathsf{Dec}(\mathsf{sk}, \cdot)$ for all inputs but c to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

In H_1 we explicitly represent what happens in KeyGen and Enc.

$H_1(\lambda, S)$:

- Generate language and corresponding trapdoor $(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^n)$.
- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.
- Let $(\mathsf{pk}'', \mathsf{sk}'') \leftarrow \mathsf{KeyGen}''(\mathsf{pp})$.
- Let $\mathsf{pk} = (\mathsf{pk}', \mathsf{pk}'')$ and $\mathsf{sk} = (\mathsf{sk}', \mathsf{sk}'')$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk})$ on public key pk with oracle access to $\mathsf{Dec}(\mathsf{sk}, \cdot)$ to receive two messages m_0, m_1 and state st_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Let $(x, w) \leftarrow \mathsf{samp}\mathcal{L}(\mathsf{pp})$.
- Let $\mathsf{k} \leftarrow \mathsf{Encap}'(\mathsf{pk}', x, w)$.
- Let $c_{\mathsf{sym}} \leftarrow \mathsf{Enc}_{\mathsf{sym}}(\mathsf{k}, \mathsf{m}_b)$.
- Let $\pi \leftarrow \mathsf{Encap}''(\mathsf{pk}'', x, w, c_{\mathsf{sym}})$.
- Let $c = (x, c_{\mathsf{sym}}, \pi)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk}, c, \mathsf{st}_1)$ with access to the decryption oracle $\mathsf{Dec}(\mathsf{sk}, \cdot)$ for all inputs but c to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

In H_2 we use the decapsulation mechanisms to encrypt the challenge message instead of encapsulation.

$H_2(\lambda, S)$:

- Generate language and corresponding trapdoor

$$(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^n).$$

- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.
- Let $(\mathsf{pk}'', \mathsf{sk}'') \leftarrow \mathsf{KeyGen}''(\mathsf{pp})$.
- Let $\mathsf{pk} = (\mathsf{pk}', \mathsf{pk}'')$ and $\mathsf{sk} = (\mathsf{sk}', \mathsf{sk}'')$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk})$ on public key pk with oracle access to $\mathsf{Dec}(\mathsf{sk}, \cdot)$ to receive two messages m_0, m_1 and state st_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Let $(x, w) \leftarrow \mathsf{samp}\mathcal{L}(\mathsf{pp})$.
- Let $\mathsf{k} \leftarrow \mathsf{Decap}'(\mathsf{sk}', x)$.
- Let $c_{\mathsf{sym}} \leftarrow \mathsf{Enc}_{\mathsf{sym}}(\mathsf{k}, \mathsf{m}_b)$.

- Let $\pi \leftarrow \mathsf{Decap}''(\mathsf{sk}'', x, c_{\mathsf{sym}})$.
- Let $c = (x, c_{\mathsf{sym}}, \pi)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk}, c, \mathsf{st}_1)$ with access to the decryption oracle $\mathsf{Dec}(\mathsf{sk}, \cdot)$ for all inputs but c to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

In H_3 we sample x from $Y \subset X \setminus \mathcal{L}$ instead of $\mathcal{L}$.

$H_3(\lambda, S)$:

- Generate language and corresponding trapdoor

$$(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^n).$$

- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.
- Let $(\mathsf{pk}'', \mathsf{sk}'') \leftarrow \mathsf{KeyGen}''(\mathsf{pp})$.
- Let $\mathsf{pk} = (\mathsf{pk}', \mathsf{pk}'')$ and $\mathsf{sk} = (\mathsf{sk}', \mathsf{sk}'')$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk})$ on public key pk with oracle access to $\mathsf{Dec}(\mathsf{sk}, \cdot)$ to receive two messages m_0, m_1 and state st_1.
- Sample bit $b \xleftarrow{\$} \{0, 1\}$ uniformly at random.
- Let $(x, \mathsf{td}_x) \leftarrow \mathsf{samp}Y(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$.
- Let $\mathsf{k} \leftarrow \mathsf{Decap}'(\mathsf{sk}', x)$.
- Let $c_{\mathsf{sym}} \leftarrow \mathsf{Enc}_{\mathsf{sym}}(\mathsf{k}, \mathsf{m}_b)$.
- Let $\pi \leftarrow \mathsf{Decap}''(\mathsf{sk}'', x, c_{\mathsf{sym}})$.
- Let $c = (x, c_{\mathsf{sym}}, \pi)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk}, c, \mathsf{st}_1)$ with access to the decryption oracle $\mathsf{Dec}(\mathsf{sk}, \cdot)$ for all inputs but c to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

In H_4 we change the behaviour of the decryption oracle.

$H_4(\lambda, S)$:

- Generate language and corresponding trapdoor

$$(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^n).$$

- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.
- Let $(\mathsf{pk}'', \mathsf{sk}'') \leftarrow \mathsf{KeyGen}''(\mathsf{pp})$.
- Let $\mathsf{pk} = (\mathsf{pk}', \mathsf{pk}'')$ and $\mathsf{sk} = (\mathsf{sk}', \mathsf{sk}'')$.
- Define an inefficient decryption algorithm Dec'.

  $\mathsf{Dec}'(c)$:
  - Parse $c = (x, c_{\mathsf{sym}}, \pi)$.
  - If $x \in \mathcal{L}$
    - Let w be the witness for x.
    - Let $\mathsf{k} \leftarrow \mathsf{Encap}(\mathsf{pk}', x, w)$.
    - Return $\mathsf{Dec}_{\mathsf{sym}}(\mathsf{k}, c_{\mathsf{sym}})$.

  - Else
    - Return $\perp$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1^{\mathsf{Dec}'}(\mathsf{pk})$ on public key pk with oracle access to Dec' to receive two messages m_0, m_1 and state st_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Let $(x, \mathsf{td}_x) \leftarrow \mathsf{samp}Y(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$.
- Let $\mathsf{k} \leftarrow \mathsf{Decap}'(\mathsf{sk}', x)$.
- Let $c_{\mathsf{sym}} \leftarrow \mathsf{Enc}_{\mathsf{sym}}(\mathsf{k}, \mathsf{m}_b)$.
- Let $\pi \leftarrow \mathsf{Decap}''(\mathsf{sk}'', x, c_{\mathsf{sym}})$.
- Let $c = (x, c_{\mathsf{sym}}, \pi)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2^{\mathsf{Dec}'}(\mathsf{pk}, c, \mathsf{st}_1)$ with oracle access to Dec' for all inputs but c to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

In H_5 we program the secret key given to the adversary to decapsulate the ciphertext to the randomly chosen key k.

$H_5(\lambda, S)$:

- Generate language and corresponding trapdoor

$$(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^{\lambda}, 1^{n}).$$

- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.
- Let $(\mathsf{pk}'', \mathsf{sk}'') \leftarrow \mathsf{KeyGen}''(\mathsf{pp})$.
- Let $\mathsf{pk} = (\mathsf{pk}', \mathsf{pk}'')$ and $\mathsf{sk} = (\mathsf{sk}', \mathsf{sk}'')$.
- Define an inefficient decryption algorthim Dec'.
  $\mathsf{Dec}'(c)$:
  - Parse $c = (x, c_{\mathsf{sym}}, \pi)$.
  - If $x \in \mathcal{L}$
    - Let w be the witness for x.
    - Let $\mathsf{k} \leftarrow \mathsf{Encap}(\mathsf{pk}', x, w)$.
    - Return $\mathsf{Dec}_{\mathsf{sym}}(\mathsf{k}, c_{\mathsf{sym}})$.
  - Else
    - Return $\perp$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1^{\mathsf{Dec}'}(\mathsf{pk})$ on public key pk with oracle access to Dec' to receive two messages m_0, m_1 and state st_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Let $(x, \mathsf{td}_x) \leftarrow \mathsf{samp}Y(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$.
- Sample $\mathsf{k} \xleftarrow{\$} \{0,1\}^{k(\lambda, S_{\mathsf{sym}}, n)}$.
- Let $c_{\mathsf{sym}} \leftarrow \mathsf{Enc}_{\mathsf{sym}}(\mathsf{k}, \mathsf{m})$.
- Let $\pi \leftarrow \mathsf{Decap}''(\mathsf{sk}'', x, c_{\mathsf{sym}})$.
- Let $c = (x, c_{\mathsf{sym}}, \pi)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2^{\mathsf{Dec}'}(\mathsf{pk}, c)$ with oracle access to Dec' for all inputs but c to produce a state st_2 smaller than S.
- Let $\mathsf{sk}'_{prog} \leftarrow \mathsf{Program}(\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x, \mathsf{sk}', x, \mathsf{k})$.

- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk} = (\mathsf{sk}'_{prog}, \mathsf{sk}''), \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

In H_6 we switch the decryption oracle back to the original behaviour.

$H_6(\lambda, S)$:

- Generate language and corresponding trapdoor

$$(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^{\lambda}, 1^{n}).$$

- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.
- Let $(\mathsf{pk}'', \mathsf{sk}'') \leftarrow \mathsf{KeyGen}''(\mathsf{pp})$.
- Let $\mathsf{pk} = (\mathsf{pk}', \mathsf{pk}'')$ and $\mathsf{sk} = (\mathsf{sk}', \mathsf{sk}'')$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk})$ on public key pk with oracle access to $\mathsf{Dec}(\mathsf{sk}, \cdot)$ to receive two messages m_0, m_1 and state st_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Let $(x, \mathsf{td}_x) \leftarrow \mathsf{samp}Y(\mathsf{pp}, \mathsf{td}_L)$.
- Sample $\mathsf{k} \xleftarrow{\$} \{0,1\}^{k(\lambda, S_{\mathsf{sym}}, n)}$.
- Let $c_{\mathsf{sym}} \leftarrow \mathsf{Enc}_{\mathsf{sym}}(\mathsf{k}, \mathsf{m})$.
- Let $\pi \leftarrow \mathsf{Decap}''(\mathsf{sk}'', x, c_{\mathsf{sym}})$.
- Let $c = (x, c_{\mathsf{sym}}, \pi)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk}, c, \mathsf{st}_1)$ with access to the decryption oracle $\mathsf{Dec}(\mathsf{sk}, \cdot)$ for all inputs but c to produce a state st smaller than S.
- Let $\mathsf{sk}'_{prog} \leftarrow \mathsf{Program}(\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x, \mathsf{sk}', x, \mathsf{k})$.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk} = (\mathsf{sk}'_{prog}, \mathsf{sk}''), \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

$H_0 \approx H_1$:

The differences between H_0 and H_1 are purely syntactical. In H_1 we just show more detail of KeyGen and Enc.

$H_1 \approx H_2$:

In H_2 we merely change how the challenge ciphertext is calculated. By the correctness of the hash proof system these two hybrids look identical to the adversary.

$H_2 \approx_c H_3$:

In H_3 sample x from $Y \subset X \setminus \mathcal{L}$ instead of $\mathcal{L}$. These two hybrids are computationally indistinguishable by the language indistinguishability. Assume there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ that can distinguish the two hybrids H_2 and H_3 with a non-negligible advantage of ϵ. From this we construct a statistical adversary $\mathcal{A}'$ that can break language indistinguishability of the HPS with advantage ϵ.

$\mathcal{A}'(\mathsf{pp}, x)$:

- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.
- Let $(\mathsf{pk}'', \mathsf{sk}'') \leftarrow \mathsf{KeyGen}''(\mathsf{pp})$.
- Let $\mathsf{pk} = (\mathsf{pk}', \mathsf{pk}'')$ and $\mathsf{sk} = (\mathsf{sk}', \mathsf{sk}'')$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk})$ on public key pk with oracle access to $\mathsf{Dec}(\mathsf{sk}, \cdot)$ to receive two messages m_0, m_1 and state st_1.

- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Let $\mathsf{k} \leftarrow \mathsf{Decap}'(\mathsf{sk}', x)$.
- Let $c_{\mathsf{sym}} \leftarrow \mathsf{Enc}_{\mathsf{sym}}(\mathsf{k}, \mathsf{m}_b)$.
- Let $\pi \leftarrow \mathsf{Decap}''(\mathsf{sk}'', x, c_{\mathsf{sym}})$.
- Let $c = (x, c_{\mathsf{sym}}, \pi)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk}, c, \mathsf{st}_1)$ with oracle access to $\mathsf{Dec}(\mathsf{sk}, \cdot)$ for all inputs but c to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

If $\mathcal{A}$ can distinguish H_2 from H_3 with advantage ε then the advantage of $\mathcal{A}'$ of distinguishing a x sampled from $\mathcal{L}$ and sampling x from $Y \subset X \setminus \mathcal{L}$ is also ϵ as it perfectly simulates H_2 in the case that $x \in \mathcal{L}$ and perfectly simulates H_3 in the other case.

$H_3 \approx_s H_4$:
According to 2-smoothness of $(\mathsf{KeyGen}'', \mathsf{Encap}'', \mathsf{Decap}'')$ in the decryption oracle for $c = (x, c_{\mathsf{sym}}, \pi)$ if $x \notin \mathcal{L}$ the decryption oracle fails with $2^{-\lambda}$ probability even when the adversary has access to the challenge ciphertext $(x^*, c^*_{\mathsf{sym}}, \pi^*)$ where $x^* \in X \setminus \mathcal{L}$. Simply not answering the query if $x \in X \setminus \mathcal{L}$ is statistically close to answering with probability $2^{-\lambda}$.

$H_4 \approx_s H_5$:
According to programmable smoothness of $(\mathsf{KeyGen}', \mathsf{Encap}', \mathsf{Decap}')$ if $x \notin \mathcal{L}$ then $(\mathsf{pk}', \mathsf{sk}', x)$ is statistically close to previous distribution $(\mathsf{pk}', \mathsf{Program}(\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x, \mathsf{sk}', x, \mathsf{k}), x)$ for uniformly random k. Because this is exactly what we switch we get that H_4 and H_5 are statistically close.

$H_5 \approx_c H_6$:
Again, According to 2-smoothness of $(\mathsf{KeyGen}'', \mathsf{Encap}'', \mathsf{Decap}'')$ in the decryption oracle for $c = (x, c_{\mathsf{sym}}, \pi)$ if $x \notin \mathcal{L}$ the decryption oracle fails with $2^{-\lambda}$ probability even when the adversary has access to the challenge ciphertext $(x^*, c^*_{\mathsf{sym}}, \pi^*)$ where $x^* \in X \setminus \mathcal{L}$. Simply not answering the query if $x \in X \setminus \mathcal{L}$ is statistically close to answering with probability $2^{-\lambda}$.

$H_6 \approx \mathsf{Dist}^{\mathsf{IncomSKE}}_{\mathcal{A}', \Pi_{\mathsf{sym}}}$:
Finally, given a mulit-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ that wins experiment $H_6(\lambda, S)$ with probability ϵ we construct an adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2, \mathcal{A}'_3)$ that wins the experiment $\mathsf{Dist}^{\mathsf{IncomSKE}}_{\mathcal{A}', \Pi_{\mathsf{sym}}}(\lambda, S + p(\lambda) + p'(\lambda))$ with probability ϵ.

$\mathcal{A}'_1(1^\lambda, 1^S)$:

- Generate language and corresponding trapdoor

$$(\mathsf{pp}, \mathsf{td}_L) \leftarrow \mathsf{Gen}(1^\lambda, 1^n).$$

- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.
- Let $(\mathsf{pk}'', \mathsf{sk}'') \leftarrow \mathsf{KeyGen}''(\mathsf{pp})$.
- Let $\mathsf{pk} = (\mathsf{pk}', \mathsf{pk}'')$ and $\mathsf{sk} = (\mathsf{sk}', \mathsf{sk}'')$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk})$ on public key pk with oracle access to $\mathsf{Dec}(\mathsf{sk}, \cdot)$ to receive two messages m_0, m_1 and state st_1.

- Let $\mathsf{st}'_1 \leftarrow (\mathsf{pk}, \mathsf{sk}, \mathsf{td}_{\mathcal{L}}, \mathsf{st}_1)$.
- Return m_0, m_1, and st'_1

$\mathcal{A}'_2(\mathsf{st}'_1, c_{\mathsf{sym}})$:

- Parse $\mathsf{st}'_1 = (\mathsf{pk}, \mathsf{sk}, \mathsf{td}_{\mathcal{L}}, \mathsf{st}_1)$
- Let $(x, \mathsf{td}_x) \leftarrow \mathsf{samp}Y(\mathsf{pp}, \mathsf{td}_L)$
- Let $\pi \leftarrow \mathsf{Decap}''(\mathsf{sk}'', x)$
- Let $c \leftarrow (x, c_{\mathsf{sym}}, \pi)$
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk}, c, \mathsf{st}_1)$ with oracle access to $\mathsf{Dec}(\mathsf{sk}, \cdot)$ for all inputs but c to produce a state st_2 smaller than S
- Return state $\mathsf{st}'_2 \leftarrow (x, \mathsf{td}_x, \mathsf{st}_2)$ smaller than $S_{\mathsf{sym}} = S + p(\lambda) + p'(\lambda)$

$\mathcal{A}'_3(\mathsf{k}, \mathsf{st}'_1, \mathsf{st}'_2, \mathsf{m}_0, \mathsf{m}_1)$:

- Parse $\mathsf{st}_1 = (\mathsf{pk}, \mathsf{sk} = (\mathsf{sk}', \mathsf{sk}''), \mathsf{td}_{\mathcal{L}}, \mathsf{st}'_1)$
- Parse $\mathsf{st}_2 = (x, \mathsf{td}_x, \mathsf{st}'_2)$
- Program $\mathsf{sk}'_{prog} \leftarrow \mathsf{Program}(\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x, \mathsf{sk}', x, \mathsf{k})$
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{pk}, (\mathsf{sk}'_{prog}, \mathsf{sk}''), \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$
- Return b'

$\mathcal{A}'$ wins $\mathsf{Dist}^{\mathsf{IncomSKE}}_{\mathcal{A}', \Pi_{\mathsf{sym}}}(\lambda, S + p(\lambda))$ iff $\mathcal{A}$ wins in $H_6(\lambda, S)$ because $\mathcal{A}'$ perfectly simulates H_6 from the perspective of $\mathcal{A}$.

□

5.6 Dangers of Using Idealized Models

In this section, we show that there is are very simple incompressible encryption scheme that are secure in the ideal cipher model (ICM) or the random oracle model (ROM). However, as soon as we instantiate the ideal cipher with a keyed permutation with a succinct description or the random oracle with a hash function, the scheme is not secure anymore. Through this we demonstrate a proof in the ICM/ROM might be meaningless in reality and extra precautions must be taken before designing schemes in the idealized models. This provides a fairly natural example on the uninstantiability of ideal models. The technique has similarities to observations by [DM04] about initial key generation for the bounded storage model.

5.6.1 Construction

First we show a construction of incompressible PKE in the ideal cipher model.

Construction 5.6.1. Let $(\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ be an IND-CPA encryption scheme with a secret key size $p(\lambda)$ and $\mathsf{P}_{\mathsf{k}} : \{0,1\}^{\mu} \to \{0,1\}^{\mu}$ be a random permutation indexed by k modelled as an ideal cipher, where $\mu = p(\lambda) + \lambda + S$.

$\mathsf{KeyGen}(1^{\lambda})$**:**

- Compute $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}'(1^{\lambda})$.
- Return pk and sk.

$\mathsf{Enc}(\mathsf{pk}, \mathsf{m} \in \{0,1\}^S)$**:**

- Sample $\mathsf{k} \xleftarrow{\$} \{0,1\}^\lambda$
- Encrypt $c' \leftarrow \mathsf{Enc}'(\mathsf{pk}', \mathsf{k})$.
- Sample $r \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random
- Let $d \leftarrow \mathsf{P}_\mathsf{k}((r, \mathsf{m}))$.
- Return $c = (c', d)$.

$\mathsf{Dec}(\mathsf{sk}, c)$**:**

- Parse $c = (c', d)$
- Decrypt $\mathsf{k} \leftarrow \mathsf{Dec}'(\mathsf{sk}', c')$.
- Compute $(r, \mathsf{m}) = \mathsf{P}_\mathsf{k}^{-1}(d)$.
- Output m.

Correctness Correctness follows trivially from the correctness of the underlying PKE and by the fact that $\mathsf{P}_\mathsf{k}^{-1}(\mathsf{P}_\mathsf{k}(r, \mathsf{m})) = (r, \mathsf{m})$. We now analyze the security of the scheme.

Theorem 5.6.2. *The scheme presented in Construction 5.6.1 has incompressible PKE security in the ideal cipher model.*

Proof. We prove the theorem via the following hybrids.

H_0 : This is the incompressible PKE security game.

- Run key generation algorithm $\mathsf{KeyGen}'(1^\lambda)$ to obtain $(\mathsf{pk}, \mathsf{sk})$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{pk})$ to receive two messages m_0 and m_1 with oracle access to P and P^{-1}.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Sample $\mathsf{k} \xleftarrow{\$} \{0,1\}^\lambda$.
- Encrypt $c' \leftarrow \mathsf{Enc}'(\mathsf{pk}, \mathsf{k})$.
- Sample $r \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random.
- Compute $d \leftarrow \mathsf{P}_\mathsf{k}(r, \mathsf{m}_b)$.
- Let $c = (c', d)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{st}_1, c)$ to produce a state st_2 smaller than S with oracle access to P and P^{-1}.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$ with oracle access to P and P^{-1}.
- The adversary wins if $b = b'$.

In this hybrid, the experiment aborts if P_k is ever queried by the first stage adversary.

H_1 :

- Run key generation algorithm $\mathsf{KeyGen}'(1^\lambda)$ to obtain $(\mathsf{pk}, \mathsf{sk})$.

- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{pk})$ to receive two messages m_0 and m_1 with oracle access to P and P^{-1}.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Sample $\mathsf{k} \xleftarrow{\$} \{0,1\}^\lambda$.
- Encrypt $c' \leftarrow \mathsf{Enc}'(\mathsf{pk}, \mathsf{k})$.
- Sample $r \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random.
- Compute $d \leftarrow \mathsf{P}_\mathsf{k}(r, \mathsf{m}_b)$.
- Let $c = (c', d)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{st}_1, c)$ to produce a state st_2 smaller than S with oracle access to P and P^{-1}.
- If P_k is queried by $\mathcal{A}_1$ or $\mathcal{A}_2$, abort.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$ with oracle access to P and P^{-1}.
- The adversary wins if $b = b'$.

In this hybrid, the experiment aborts if the final stage adversary ever queries $\mathsf{P}_\mathsf{k}^{-1}$ on the value d.

H_2 :

- Run key generation algorithm $\mathsf{KeyGen}'(1^\lambda)$ to obtain $(\mathsf{pk}, \mathsf{sk})$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{pk})$ to receive two messages m_0 and m_1 with oracle access to P and P^{-1}.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Sample $\mathsf{k} \xleftarrow{\$} \{0,1\}^\lambda$.
- Encrypt $c' \leftarrow \mathsf{Enc}'(\mathsf{pk}, \mathsf{k})$.
- Sample $r \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random.
- Compute $d \leftarrow \mathsf{P}_\mathsf{k}(r, \mathsf{m}_b)$.
- Let $c = (c', d)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{st}_1, c)$ to produce a state st_2 smaller than S with oracle access to P and P^{-1}.
- If P_k is queried by $\mathcal{A}_1$ or $\mathcal{A}_2$ abort.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$ with oracle access to P and P^{-1}.
- If $\mathcal{A}_3$ queries $\mathsf{P}_\mathsf{k}^{-1}(d)$ before ever querying $\mathsf{P}_\mathsf{k}((r, \mathsf{m}_b))$ abort.
- The adversary wins if $b = b'$.

In this hybrid, the experiment aborts if the adversary queried P_k on (r, m_b). This removes almost all information about m_b.

H_3 :

- Run key generation algorithm $\mathsf{KeyGen}'(1^\lambda)$ to obtain $(\mathsf{pk}, \mathsf{sk})$.

- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{pk})$ to receive two messages m_0 and m_1 with oracle access to P and P^{-1}.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Sample $\mathsf{k} \xleftarrow{\$} \{0,1\}^\lambda$.
- Encrypt $c' \leftarrow \mathsf{Enc}'(\mathsf{pk}, \mathsf{k})$.
- Sample $r \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random.
- Compute $d \leftarrow \mathsf{P}_\mathsf{k}(r, \mathsf{m}_b)$.
- Let $c = (c', d)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{st}_1, c)$ to produce a state st_2 smaller than S with oracle access to P and P^{-1}.
- If P_k is queried by $\mathcal{A}_1$ or $\mathcal{A}_2$, abort.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$ with oracle access to P and P^{-1}.
- If $\mathcal{A}_3$ queries $\mathsf{P}_\mathsf{k}^{-1}(d)$ before ever querying $\mathsf{P}_\mathsf{k}((r, \mathsf{m}_b))$ abort.
- If $\mathcal{A}_3$ queries $\mathsf{P}_\mathsf{k}((r, \mathsf{m}_b))$ abort.
- The adversary wins if $b = b'$.

We now show indistinguishability of hybrids.

$H_0 \approx_c H_1$:

The hybrids are identical except for the abort condition. The experiment aborts the protocol if the ideal cipher P_k or $\mathsf{P}_\mathsf{k}^{-1}$ is ever queried by $\mathcal{A}_1$ or $\mathcal{A}_2$. We argue that the winning probability in H_0 and H_1 are negligibly close if (KeyGen', Enc', Dec') is an IND-CPA secure PKE.

Assume there is an adversary $\mathcal{A}_1$, $\mathcal{A}_2$ that queries k with probability ε. From this, we construct an adversary $\mathcal{A}'_1$, $\mathcal{A}'_2$ that breaks the IND-CPA security of the encryption scheme (KeyGen', Enc', Dec').

$\mathcal{A}'_1(\mathsf{pk})$:

- Sample $\mathsf{k}_0, \mathsf{k}_1 \xleftarrow{\$} \{0,1\}^\lambda$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{pk})$ to receive two messages m_0 and m_1 with oracle access to P and P^{-1}.
- Return k_0 and k_1 as challenge messages.

$\mathcal{A}'_2(\mathsf{pk}, c')$:

- Sample $r \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random.
- Sample $b' \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Let $\mathsf{k} = \mathsf{k}_{b'}$
- Let $d \leftarrow \mathsf{P}_\mathsf{k}((r, \mathsf{m}_{b'}))$.
- Let $c = (c', d)$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{st}_1, c)$ with oracle access to P and P^{-1}.
- If $\mathcal{A}_1$ or $\mathcal{A}_2$ ever queried P_k or $\mathsf{P}_\mathsf{k}^{-1}$ return 0 else 1.

If $b = b'$ then $(\mathcal{A}'_1, \mathcal{A}'_2)$ outputs 0 with probability ε as this perfectly simulates H_0 to the adversary $(\mathcal{A}_1, \mathcal{A}_2)$.

If $b \neq b'$ then $(\mathcal{A}'_1, \mathcal{A}'_2)$ will output 1 with probability $1 - \mathsf{negl}[\lambda]$ as c contains no information about k beyond $d = \mathsf{P}_\mathsf{k}((r, \mathsf{m}_{b'}))$.

Combining the two cases the probability of winning $\mathcal{A}'$ winning the IND-CPA experiment is $\frac{1}{2}\varepsilon + \frac{1}{2}(1 - \mathsf{negl}[\lambda])$. Which is a non-negligible advantage if ε is non-negligible.

$H_1 \approx H_2$:

The hybrids are identical except for the new abort condition. We argue that the winning probability in H_1 and H_2 are negligibly close if the adversary can only polynomially many queries $q(\lambda)$.

Let D, SK, ST_2 be the random variables that reflect the occurrence of d, sk, st_2 in the execution of the experiment.

We know that $H_\infty(D) = S + \lambda + p(\lambda)$ because without access to the permutation P_k (which $\mathcal{A}_1$ and $\mathcal{A}_2$ don't have) we have d is a uniformly random string.

We are going to show that $\tilde{H}_\infty(D|SK, ST_2) \geq \lambda$. We have that

$$\begin{aligned}\tilde{H}_\infty(D|SK, ST_2) &\geq H_\infty(D) - p(\lambda) - S \\ &\geq S + \lambda + p(\lambda) - p(\lambda) - S = \lambda\end{aligned}$$

where the first inequality follows from Lemma 2.2.2 and the last step follows from the fact that D is a uniform vector over $\mu = S + \lambda + p(\lambda)$.

Hence the adversary with $q(\lambda)$ queries can at most query $\mathsf{P}_\mathsf{k}^{-1}$ on d with probability $q(\lambda)/2^\lambda$. Therefore, the winning probability of H_1 only be better than that of H_2 by $q(\lambda)/2^\lambda$.

$H_2 \approx H_3$:

The only difference between H_2 and H_3 is additional abort condition if $\mathcal{A}_3$ queries $\mathsf{P}_k((r, \mathsf{m}_b))$. We argue that the winning probability in H_2 and H_3 are negligibly close if the adversary can only polynomially many queries $q(\lambda)$.

The abort condition, however, can only be reached if previous abort conditions are not met. So the adversary needs to guess r, which is uniformly random in $\{0,1\}^\lambda$. Therefore, if the adversary has $q(\lambda)$ queries to the ideal cipher, she has at most reached the new abort condition of querying $\mathsf{P}_k((r, \mathsf{m}_b))$ with probability of $q(\lambda)/2^\lambda$.

H_3 :

In H_3 the adversary learns no information about m_b because it can never query $\mathsf{P}_k((r, \mathsf{m}_b))$ or $\mathsf{P}_k^{-1}(d)$ and all other queries are uniformly random subject to being different from all other queries and $\mathsf{P}_k((r, \mathsf{m}_b)) = d$.

□

We also present this construction of an incompressible PKE in the ROM as described in [GWZ23] Construction 9+10.

Construction 5.6.3. Let $(\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ be an IND-CPA encryption scheme with a secret key size $p(\lambda)$ and $\mathsf{H}_n : \{0,1\}^* \to \{0,1\}^n$ be a random function indexed by n modelled as a hash function.

$\mathsf{KeyGen}(1^\lambda)$**:**

- Compute $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}'(1^\lambda)$.
- Return pk and sk.

$\mathsf{Enc}(\mathsf{pk}, \mathsf{m} \in \{0,1\}^S)$**:**

- Sample $\mathsf{k} \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random
- Sample $r \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random
- Encrypt $c' \leftarrow \mathsf{Enc}'(\mathsf{pk}', \mathsf{k})$.
- Let $d \leftarrow \mathsf{H}_S(\mathsf{k}, r) \oplus \mathsf{m}$.
- Let $h \leftarrow \mathsf{H}_\lambda(\mathsf{k}, d) \oplus r$
- Return $c = (c', d, h)$.

$\mathsf{Dec}(\mathsf{sk}, c)$**:**

- Parse $c = (c', d, h)$
- Decrypt $\mathsf{k} \leftarrow \mathsf{Dec}'(\mathsf{sk}', c')$.
- Let $r \leftarrow \mathsf{H}_\lambda(\mathsf{k}, d) \oplus h$.
- Compute $\mathsf{m} \leftarrow \mathsf{H}_S(\mathsf{k}, r) \oplus d$.
- Output m.

Correctness and security of this scheme follows by the same proof as the incompressible SKE in [GWZ23].

5.6.2 Attack

We instantiate the ideal cipher in Construction 5.6.1 with a specific keyed permutation P. Moreover, the only condition we have on the underlying PKE is that it is IND-CPA secure. Hence, we can instantiate it with an FHE scheme $(\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Eval}', \mathsf{Dec}')$. The idea of the attack is that, as soon as the ideal cipher is a permutation P_k with a succinct description, the adversary can make non-black-box use of the permutation. It homomorphically evaluates the permutation inside the FHE allowing it to compress the ciphertext into an encryption of a single bit.

We now provide the description of the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$.

$\mathcal{A}_1(\mathsf{pk})$: Output any distincit two messages $\mathsf{m}_0, \mathsf{m}_1$.

$\mathcal{A}_2(c)$:

- Parse $c = (c', d)$.
- Consider the circuit $\mathcal{C}(x) = \mathcal{C}(x)_{d,\mathsf{m}_0}$ which computes the following:
  1. Evaluate $w = \mathsf{P}_\mathsf{k}^{-1}(d)$.
  2. If $w = \mathsf{m}_0$, output 0. Otherwise, output 1.
- Compute $\tilde{c} \leftarrow \mathsf{Eval}'(\mathsf{pk}, \mathcal{C}, c')$.
- Output $\mathsf{st} = \tilde{c}$.

$\mathcal{A}_3(\mathsf{st}, \mathsf{sk})$: Compute $b' \leftarrow \mathsf{Dec}'(\mathsf{sk}, \tilde{c})$. Output b'.

We can establish that $b' = b$ by the homomorphic correctness of the FHE.

Size of st. The state st of the adversary is composed by a single FHE ciphertext encrypting one bit. Hence, by the compactness of the FHE $|\mathsf{st}| = \mathsf{poly}[\lambda]$.

Attack on Construction 5.6.3 The same attack concept also works for Construction 5.6.3. We the sketch attack below

$\mathcal{A}_1(\mathsf{pk})$: Output any distincit two messages $\mathsf{m}_0, \mathsf{m}_1$.

$\mathcal{A}_2(c)$:

- Parse $c = (c', d, h)$.
- Consider the circuit $\mathcal{C}(x) = \mathcal{C}(x)_{d,h,\mathsf{m}_0}$ which computes the following:
  1. Let $r \leftarrow \mathsf{H}_\lambda(x, d) \oplus h$.
  2. Let $w \leftarrow \mathsf{H}_S(x, r) \oplus d$.
  3. If $w = \mathsf{m}_0$, output 0. Otherwise, output 1.
- Compute $\tilde{c} \leftarrow \mathsf{Eval}'(\mathsf{pk}, \mathcal{C}, c')$.
- Output $\mathsf{st} = \tilde{c}$.

$\mathcal{A}_3(\mathsf{st}, \mathsf{sk})$: Compute $b' \leftarrow \mathsf{Dec}'(\mathsf{sk}, \tilde{c})$. Output b'.

Remark 5.6.4. Notice encrypting the key of any (plain model) incompressible encryption scheme in an FHE and adding this to the public key removes the incompressibility by the same argument. This observation demonstrates the dangers of using the ICM in incompressible encryption and that incompressible encryption is a fairly delicate notion.

Chapter 6

Space-Hard Functions

6.1 Introduction

Timed Cryptography. Traditionally, in public key cryptography [DH76], the ability to decrypt ciphertexts which have been generated with a public key pk is tied to the possession of a secret key sk corresponding to pk. Likewise, generation of signatures with respect to a verification key vk is tied to the possession of corresponding signing key. Timed cryptography [May93, CLSY93, RSW96] adds a twist to this rigid paradigm: Rather than the possession of a secret key, *investing time* facilitates the decryption of a ciphertext or generation of signature. In other words, time-lock encryption allows to *encrypt to the future*.

This enables new applications both in theory and practice: Timed commitments [BN00] facilitate e.g. fair exchange and fair coin-toss in the two party setting, notions which have been shown to be beyond reach of standard cryptographic notions [Cle86]; Likewise, from a more practical angle time-lock puzzles play a crucial role in the design of public randomness beacons, a crucial component in the design of distributed ledgers (see e.g. [KWJ24]).

Verfiable Delay Functions Boneh et al. [BBBF18] introduced the notion of *verifiable delay functions* (VDFs), which can be loosely thought of as the timed analogue of digital signatures: Computing a VDF output together with a certificate of its validity takes a long time T, whereas verification of a certificate can be performed rapidly, that is in time $\mathsf{poly}(\lambda, \log(T))$. VDFs are likewise powerful tools in the construction of randomness beacons and consensus protocols, as they e.g. facilitate techniques such as *self-selection* [CM16] and proofs of replication [ABBK16].

Boneh et al. [BBBF18] provide both generic and concrete constructions of VDFs. They obtain generic constructions by combining specific sequential functions, such as iterated hashing, with incrementally verifiable computation [Val08, BCCT13].

Alas, when it comes to concrete assumptions, VDFs in particular and timed cryptography in general rest on a rather narrow foundation; most candidates of time-lock puzzles and verifiable delay functions are tied to the sequential squaring assumption in groups of unknown order and related problems [Pie19, Wes19, DMPS19]. Bitansky et al. [BGJ$^+$16] showed that by relying on indistinguishability obfuscation, timed primitives can be realized assuming the minimal assumption that inherently sequential problems exist. As this construction relies on very heavy theoretical tools, its appeal is currently limited to the domain of pure theory.

Proofs of sequential work [MMV13, CP18, DLM19] can be seen as a more lightweight alternative to VDFs and are achievable from potentially weaker assumptions. However, PoSW lack a uniqueness property, which makes them unsuitable for many of the more advanced applications of VDFs.

The concrete VDF candidates given in [BBBF18] constitute a notable exception from the sequential squaring blueprint. These candidates are based on a novel family of hardness assumptions relating to the inversion of rational functions of high degree.

Space-Hard Cryptography. Conceptually, there is nothing intrinsically special about the computational resource of *time*. Hence, a natural conceptual next step is to consider more general computational resources. In fact, there is a growing body of works investigating the notion of *memory or space-hard functions* [Per09, AS15a, AB16, ACP+17, BP17, ABB22, AGP24].

In this work, we are concerned with both *space-lock puzzles*, the space-analogue of time-lock puzzles, and verifiable space-hard functions, the analogue of VDFs.

Syntactically, we define a space-lock puzzle to consist of two algorithm Gen and Solve. Gen takes as input a space parameter S and a message m and outputs a puzzle p, whereas Solve takes a puzzle p and outputs a message m. In terms of efficiency, we require that Gen runs in time and space $\mathsf{poly}(\lambda, \log(S))$, whereas Solve runs in space S. In terms of security, we require that any algorithm running in time $\mathsf{poly}(\lambda, S)$ having access to space of size at most $S^{1-\epsilon}$ has at most negligible advantage guessing an encrypted bit.

A verifiable space-hard function syntactically consists of two algorithms Eval and Verify (potentially along with a setup algorithm producing public parameters). Eval takes a space parameter S and a value x and outputs a value y and a certificate π, whereas Verify takes inputs x, y and a certificate π and outputs either accept or reject. In terms of efficiency, we require that Eval runs in space S, whereas Verify runs in time and space $\mathsf{poly}(\lambda, \log(S))$. In terms of security, we require *computational uniqueness* and *space-hardness*. Computational uniqueness requires that no algorithm running in time and space $\mathsf{poly}(\lambda, S)$ can produce a verifying tuple x, y', π' with $y' \neq y$, where $(y, \pi) = \mathsf{Eval}(S, x)$. Space-hardness requires that no algorithm running in time $\mathsf{poly}(\lambda, S)$ and space $S^{1-\epsilon}$ finds y with non-negligible probability.

In terms of assumptions and constructions, the design-space of space-hard cryptography is comparatively much less explored than that of timed cryptography. In terms of generic constructions, a closer look at the time-lock puzzle construction given in [BGJ+16] reveals that this construction can be adapted to space-lock puzzles, i.e. we can construct a space-lock puzzles assuming iO and, additionally, the minimal assumption that inherently space-hard computations exist.

However, critically, there are currently no algebraically structured candidates for efficient space-lock puzzles.

6.1.1 Our Results

In this work, we take a first step in studying efficient space-hard primitives from algebraic assumptions relating to the solvability of sparse univariate polynomials of large degree. The contributions of our work are two-fold:

1. [DDJ24][1] provide an efficient attack against the specific proposal of the "Inverting

[1] This is the paper that corresponds to this chapter. The attack is left out of the chapter as I did not significantly contribute to it. The high level description of the attack remains in the chapter to outline the problems with designing permutation polynomials.

Injective Rational Maps" Assumption of Boneh et al. [BBBF18]. While we do not break the assumption in its most general form, we demonstrate a full break on their suggested candidate, which indicates that the assumption stands on brittle ground. A major challenge towards instantiating the general assumption is finding suitable families of injective rational maps, and [BBBF18] suggested a family of rational functions of (large) degree d constructed by [GM97]. In [BBBF18] it was conjectured that this family cannot be inverted in time and space $\mathsf{poly}(\log(d))$ (i.e. by circuits of size $\mathsf{poly}(\log(d))$). [DDJ24] provides and implements an algorithm which inverts these rational functions in time and space $\mathsf{poly}(\log(d))$, thus falsifying the main candidate instantiation of the Inverting Injective Rational Maps assumption.

We remark that this VDF was a *weak* VDF to begin with, i.e. algorithms that run in time $\mathsf{poly}(\log(d))$ and space $O(d^c)$ for some $c \geq 1$ were known and discussed in [BBBF18]. The main innovation of this part of our work is that our attack runs in *both* time and space $\mathsf{poly}(\log(d))$.

2. In Section 6.2 we introduce and discuss a new algebraically structured space-hardness assumption which we refer to as the *sparse root finding* (SRF) assumption. Building on this, in Section 6.3 we provide a construction of spacelock puzzles from the SRF assumption, whereas in Section 6.4 we construct a verifiable space-hard function from the SRF assumption.

6.1.2 Our Techniques

Inverting Guralnick Müller Polynomials As mentioned above, Boneh et al. [BBBF18] provided a concrete candidate for a VDF based on Guralnick-Müller permutation polynomials [GM97]. These are defined via rational functions $f_{\mu,q}$ over a finite field $\mathbb{F}_{p^m}$ and parametrized by an element $\mu \in \mathbb{F}_{p^m}$ and a (large) degree parameter $q = p^r$ (for some $r < m$). Both μ and q need to obey some additional constraints to ensure that the function is a permutation. The function $f_{\mu,q}(X)$ is then given by

$$f_{\mu,q}(X) = \frac{(X^q - \mu X - \mu)(X^q - \mu X + \mu)^q + ((X^q - \mu X + \mu)^2 + 4\mu^2 X)^{(q+1)/2}}{2X^q}$$

Notice, that this function is neither linear nor affine, consequently at a first glance one may reasonably conjecture that it takes space proportional to q to invert it on random inputs. In fact, [BBBF18] provide a survey of cryptanalytic techniques to invert rational functions and argue why these techniques fail for the case of Guralnick-Müller polynomials. This includes inversion of *extremely* sparse polynomials, linear algebraic attacks, as well as attacks against so-called *exceptional polynomials*, which remain permutations when considered as rational functions over the extension field $\mathbb{F}_{p^{m'}}$ *for infinitely many choices of the degree* m'. Boneh et al. [BBBF18] conjecture that inverting a function $f_{\mu,q}$ for a randomly chosen $\mu \in \mathbb{F}_{p^m}$ (under some constraints) takes time polynomial in the degree parameter q.

Yet, [DDJ24] fully break of this assumption. Interestingly, they draw the mathematical tools for this attack from the original work of Guralnick and Müller [GM97]. They observe the following: While the function $f_{\mu,q}$ itself is not affine, the problem of inverting $f_{\mu,q}$ on a target $t \in \mathbb{F}_{p^m}$ can be *embedded into* a linear system of higher degree. Specifically, [GM97] provides us with the following property of $f_{\mu,q}$: If θ is $q-1$-st root of μ in the algebraic closure of $\mathbb{F}_{p^m}$, then there exist efficiently computable coefficients A_0, B_0, B_1, B_2 (depending

on μ and t) in an extension of $\mathbb{F}_{p^m}$ such that

$$\prod_{i \in \mathbb{F}_q} (f_{\mu,q}(X + i\theta) - t) = X^{q^3} + B_2 X^{q^2} + B_1 X^q + B_0 X + A_0. \tag{6.1}$$

Consequently, any solution $\xi \in \mathbb{F}_{p^m}$ of $f_{\mu,q}(\xi) = t$ is also a solution to the right-hand side of equation (6.1), i.e. such a solution satisfies

$$\xi^{q^3} + B_2 \xi^{q^2} + B_1 \xi^q + B_0 \xi + A_0 = 0. \tag{6.2}$$

Now observe that equation (6.2) is in fact a linear equation system (as exponentiation with q is a Frobenius action). Hence, we can efficiently compute a solution space using standard linear algebra techniques. [DDJ24] provide a proof that this attack works with overwhelming probability and a implementation of the attack in MAGMA that solves big instances in milliseconds.

Space-Hardness from Inverting Sparse Polynomials Guralnick-Müller polynomials are one specific instance of the more general problem of finding roots of sparse polynomials [2], a problem which we will refer to as Sparse Root Finding (SRF).

As [BBBF18] note, their root-finding-based candidate achieves only a mild form of sequentiality to begin with. In fact, a moderate polynomial increase in parallel computation power will enable a solver to find roots significantly faster. On the other hand, the space-hardness of these problems seems to be much more robust, as all known algorithms for this type of problem consume a large amount of space. In fact, the amount of memory scales linearly with the degree of the polynomial.

This is the starting point for the constructive results in this work. In a nutshell, we consider the problem of inverting sparse, high degree polynomials but drop the requirement that the polynomial needs to act as a permutation. Hence, the resulting problem carries significantly less structure than e.g. inverting Guralnick-Müller polynomials and does not provide an obvious angle for cryptanalysis.

More importantly, by basing our constructions on the problem of root-finding for general sparse polynomials, we can achieve a win-win scenario:

- If our assumptions hold, we obtain practically efficient candidates for space-hard cryptography
- While we do not provide a worst-to-average case reduction, refuting our assumptions would constitute a considerable advance in the algorithmic state-of-the art of polynomial factorization algorithms, as it is a long open problem to design polynomial factorization algorithms which leverage sparsity (in a non-extreme parameter regime).

Space-Lock Puzzle Building Space-Lock Puzzles from the assumption that SRF is space-hard in sparse high-degree polynomials is fairly straight forward. To generate a puzzle for a random message m, generate a random sparse polynomial $f(X)$ with high degree and a random constant coefficient. Now, we know that $f(X) - f(m)$ has a root at m and can be output as a space-lock puzzle for m. There are two minor problems with this construction. First, we might want to create a puzzle for a non-random message, which we can resolve by using hybrid encryption. Second, there might be polynomials $f(X) - f(m)$ with multiple roots. We can fix this problem by padding the message and checking for the correct padding after solving the puzzle.

[2] More generally, we can consider this as finding roots of structured polynomials which can be evaluated quickly.

Verifiable Space-Hard Functions. We start by discussing our construction of verifiable space-hard functions from SRF. As we let go of the permutation requirement of the polynomials, we need to work harder to make this function verifiable. Our technical tool to achieve this is a novel and efficient special-purpose proof system for certifying the greatest common divisor (gcd) between the polynomial $f(X)$ and $X^p - X$. This is sufficient, as given this gcd one can quickly and space-efficiently find the roots of $f(X)$.

For the purpose of this outline, assume that we have cheap way to prove equations over high-degree and possibly dense polynomials. We will later explain how to carry out these checks. We make use of the fact that the greatest common divisor between a polynomial $f(X)$ and $X^p - X$ is constant degree with high probability over the choice of a random sparse polynomial $f(X)$. Our proof system establishes that some constant degree polynomial $g(X)$ is the gcd of $f(X)$ and $X^p - X$ in two phases.

In the first phase, the prover computes $f'(X) = X^p - X \mod f(X)$ via square and multiply. Each step of this computation is defined by a simple polynomial equation

$$(X^{2^n} \mod f(X)) \cdot (X^{2^n} \mod f(X)) = (X^{2^{n+1}} \mod f(X)) + h(X)f(X)$$

for some polynomial $h(X)$.

In the second phase, we use that

$$gcd(X^p - X, f(X)) = gcd(f'(X), f(X))$$

and compute $g(X) = gcd(f'(X), f(X))$ together with its Bézout coefficients $a(X)$, $b(X)$ via the extended Euclidean algorithm. The greatest common divisor is unique, up to normalization, hence we require the prover to normalize this polynomial. Bézouts identity guarantees that for all $\bar{a}(X)$, $\bar{b}(X)$ we have $\bar{a}(X)f'(X) + \bar{b}(X)f(X)$ is a multiple of $gcd(f'(X), f(X))$. Further, the verifier can check whether $g(X) = a(X)f'(X) + b(X)f(X)$ is a divisor, by making sure that $f(X) \mod g(X) = 0$ and $X^p - X \mod g(X) = 0$. Now we have verified that $g(X)$ is a divisor of $f(X)$ and $f'(X)$ and that $g(X)$ is a multiple of the their greatest common divisor, therefore, it is a greatest common divisor.

So far we have skipped over the issue of how we can verify polynomial equations, when the polynomials have representations that are bigger than the verifier's space. Instead of the verifier checking these equations over the polynomials the prover commits to the polynomials using a polynomial commitment scheme. The verifier then checks these equations by evaluating the polynomials at a random location. The verifier can then check the equations on the evaluated polynomials. Soundness follows from the Schwartz-Zippel lemma.

These polynomial commitment schemes can be instantiated with the popular pairing based [KZG10], lattice based schemes like [CMNW24], or IOP based polynomial commitment schemes [BBHR18, ACY23, ACFY24a, ACFY24b].

6.1.3 Open Problems

We consider it to be an interesting question to investigate whether it is possible to intrinsically and flexibly tie the resources of space and time in a puzzle or verifiable time-space hard function. Specifically, is it possible to force the puzzle solver to spend S space for T time? Here S and T are adjustable parameters. This concept may be most closely captured by the concept of sustained space complexity [ABP18].

We believe any solution to this problem that goes beyond taking a sequential function that has a scalable domain and generically applying incrementally verifiable computation to it might be of big interest. An example for such a function is sequential squaring over a

modulus that scales with the space parameter. More specifically the function could be on input $x \in [2^\lambda]$ compute $h((N^s - x)^{2^T} \mod N^s)$ where h is some compressing function to reduce the size of the result.

6.1.4 Related Work

There are many works in memory restricted cryptography such as memory-hard functions and various forms of proof of space. Memory-hard functions are functions that are only computable with a large amount of memory accesses. The measure that many works use is called cumulative memory complexity. These functions are used to reduce the effectiveness of building application specific integrated circuits (ASICs) or field programmable gate arrays (FPGA) for brute force attacks because these excell at computation and do not have a faster way of accessing memory than off-the-shelf CPUs. Memory-hard functions are used in password hashing, proof of work, and other applications where the goal is to make computation expensive. The first memory-hard function was proposed by Percival in 2009 [Per09]. So far, all memory hard functions [Per09, AS15a, AB16, ACP+17] use graphs with special structure and iteration of a function to force anyone trying to evaluate the function to do a lot of memory accesses.

The notion of space-hardness we consider was coined "Transient Space" in the works of Ren and Devadas [RD16]. We are trying to increase the amount of maximum storage that is necessary to compute the function, which we call space hardness. A space-lock puzzle closely resemble trapdoor memory-hard function [AGP24], asymmetrically memory-hard functions [BP17] and memory-hard puzzles [ABB22], but under the notion of memory hardness.

We introduce space-hard functions and their verifiable counter part. Verifiable space-hard functions are a space-analogue of verifiable delay functions. We heavily deviate from the design space of memory-hard functions as most constructions are based on the random oracle model. Therefore, using incrementally verifiable computation to verify their evaluation requires proving statements over random oracle, which is concretely inefficient and conceptually unsatisfying. Indeed, [ABFG14] show how to verify that the function used much memory, but not that the output is correct. [DFKP15] further extend the notion to proof of space, where the prover executes a memory-hard function and then regularly gets queried to prove that he maintains a large amount of the computation in his memory. For an excellent overview on the topic, we refer to [RD16], as they detail the different notions and their relations.

Our verifiable space-hard function follows a similar design principle as the weak verifiable-delay function suggested by [BBBF18]. They suggest that inverting a fast to evaluate high degree permutation polynomial requires a lot of sequential computation. We instead conjecture and use the space-hardness of the same computation. Because we, however, want to move away from permutation polynomials to less structured polynomials our constructions require a special purpose proof system.

Indeed, verifiable space-hard functions can be though of as a space-analogue of verifiable delay functions [LW17, BBBF18, Wes19, Pie19, HHK+22, HHKK23] and space-lock puzzles as a space-analogue of time-lock puzzles [RSW96].

6.2 Space-Hardness of Root-Finding

We conjecture that root-finding for polynomials over a big finite field requires a lot of space. As far as we are aware of, all root finding algorithms [Ber70, CZ81]

[VZGS92, Sho93, KS95, KU11, GNU16] in a finite field $\mathbb{F}_p$ for large p and comparatively smaller degree d, start by computing $X^p - X \mod f(X)$. For a discussion of recent results, see [GNU16]. In general, this polynomial $X^p - X \mod f(X)$ is a dense polynomial of degree $d-1$, whose representation requires d elements in $\mathbb{F}_p$. For this reason, we conjecture a minimal space of d for any algorithm with a runtime $o(p)$. Proving this conjecture wrong would greatly advance the state of the art concerning polynomial factorization.

Assumption 6.2.1 (Sparse Root-Finding (SRF))**.** We define the space-hardness of finding a root in a polynomial from distribution $\mathcal{D}_{\lambda,S}$ as follows: Root-Finding is hard with a gap $\varepsilon < 1$ if there exists a polynomial $\tilde{S}(\cdot)$ such that for all polynomials $S(\cdot) \geq \tilde{S}(\cdot)$ and PPT adversaries $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ with space bound $S^\varepsilon(\lambda)$ there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$:

$$\Pr\left[f(x^*) = 0 \middle| \begin{matrix} f(X) \leftarrow \mathcal{D}_{\lambda,S(\lambda)} \\ x^* \leftarrow \mathcal{A}_\lambda(f(X)) \end{matrix}\right] = \mathsf{negl}(\lambda)$$

We also define a space-hardness assumption for the problem of computing the greatest common divisor of a polynomial and $X^p - X$.

Assumption 6.2.2 (Sparse GCD Computation)**.** We define the space-hardness of gcd computation from distribution $\mathcal{D}_{\lambda,S}$ as follows: gcd computation is hard with a gap $\varepsilon < 1$ if there exists a polynomial $\tilde{S}(\cdot)$ such that for all polynomials $S(\cdot) \geq \tilde{S}(\cdot)$ and PPT adversaries $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ with space bound $S^\varepsilon(\lambda)$ there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$:

$$\Pr\left[g(X) = gcd(f(X), X^p - X) \middle| \begin{matrix} f(X) \leftarrow \mathcal{D}_{\lambda,S(\lambda)} \\ g(X) \leftarrow \mathcal{A}_\lambda(f(X)) \end{matrix}\right] = \mathsf{negl}(\lambda)$$

Lemma 6.2.3. *If root-finding is hard with gap $\varepsilon < 1$ then gcd computation is hard with gap $\varepsilon < 1$.*

Proof. Given an adversary $\mathcal{A}$ that breaks Assumption 6.2.2 we construct an adversary $\mathcal{A}'$ that breaks Assumption 6.2.1.

$\mathcal{A}'(f(X))$:

- Let $g(X) \leftarrow \mathcal{A}(f(X))$ where $g(X)$ is an n-degree polynomial.
- Factor $g(X)$ into degree 1 polynomials $X - h_1, \ldots, X - h_n$ using the Cantor-Zassenhaus algorithm.
- Return h_1.

The factors of $g(X)$ can only be of degree 1 because $X^p - X$ only factors of degree 1. The Cantor-Zassenhaus [CZ81] algorithm has space-complexity $O(n \log p)$ [Sho93] and runs in $\mathsf{poly}(n, \log p)$.

□

We require these assumptions for two different but related applications, which we will detail in later chapters. One is a space lock puzzle and the other a verifiable space-hard function. For the space lock puzzle we only really require that the polynomials by the distributions are fast to evaluate and that root-finding is space-hard.

In general, we will stick with prime order fields because they tend to have less structure, which might protect them against structural attacks. We will also try to impose as little structure as possible on the polynomials. In order to make proofs over the these fields better we choose FFT-friendly fields.

A natural candidate is the distribution of random sparse polynomials. We make sure that the lowest two monomials are random for better estimation of number of roots. More formally, we define the distribution $\mathcal{D}_{\lambda,S}$ as follows:

Definition 6.2.4 (Random Sparse Polynomial (with Uniform Constant and Linear Coefficient))**.** Pick a prime $p \in \Omega(2^\lambda)$, degree $d \in \Omega(S)$, and number of non-zero monomials $k \in \Omega(\lambda)$. Operations happen over $\mathbb{F}_p$. Output the following univariate (the variable is X) polynomial

$$a_0 + a_1 X + \sum_{i \in [k-2]} a_i X^{e_i} + X^d$$

For uniformly random $a_i \xleftarrow{\$} \mathbb{F}_q$ and $e_i \xleftarrow{\$} [d-1]$.

We prove these polynomials define a family of strongly universal$_2$ hash functions [WC81].

Lemma 6.2.5. *For any polynomial $h(X)$, any distict $x_1, \ldots, x_t \in \mathbb{F}_p$, and any possibly non-distinct $y_1, \ldots, y_t \in \mathbb{F}_p$, we have that*

$$\Pr_{a_0,\ldots,a_{t-1} \xleftarrow{\$} \mathbb{F}_p} [h(x_1) + l(x_1) = y_1, \ldots, h(x_t) + l(x_t) = y_t] = p^{-t}$$

where $l(X) = \sum_{i \in [t]} a_{i-1} X^{i-1}$.

Proof. The statment $h(x_1)+l(x_1) = y_1, \ldots, h(x_t)+l(x_t) = y_t$ is equivalent to the this linear system of equations:

$$\begin{pmatrix} 1 & x_1 & \ldots & x_1^{t-1} & h(x_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_t & \ldots & x_t^{t-1} & h(x_t) \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_{t-1} \\ 1 \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_t \end{pmatrix} \qquad \Leftrightarrow$$

$$\begin{pmatrix} 1 & x_1 & \ldots & x_1^{t-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_t & \ldots & x_t^{t-1} \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_{t-1} \end{pmatrix} = \begin{pmatrix} y_1 - x_1^d \\ \vdots \\ y_k - x_k^d \end{pmatrix}$$

The matrix is a Vandermonde matrix, which is invertible. Therefore, multiplication by it is a bijection. Because $a_0, \ldots, a_{t-1}$ are uniformly random, the probability of the system of equations to hold is p^{-t}. □

Via a inclusion-exclusion argument, we can upper bound the probability of the polynomial having no roots.

Lemma 6.2.6. *For any polynomial $h(X)$ and uniform $a_0, a_1 \xleftarrow{\$} \mathbb{F}_p$ we have $a_0 + a_1 X + h(X)$ no roots with probability $\leq 1/2$.*

Proof. We have

$$\begin{aligned}
&\Pr_{a_0,a_1}[\vee_{x\in\mathbb{F}_p}(a_0+a_1x+h(x)=0)]\\
\geq&\sum_{x\in\mathbb{F}_p}\Pr_{a_0,a_1}[a_0+a_1x+h(x)=0] &(6.3)\\
&-\sum_{x_1<x_2\in\mathbb{F}_p}\Pr_{a_0,a_1}[\wedge_{x\in\{x_1,x_2\}}(a_0+a_1x+h(x)=0)]\\
=&1-\sum_{x_1<x_2\in\mathbb{F}_p}1/p^2 &(6.4)\\
\geq&1/2
\end{aligned}$$

Inequality 6.3 follows from a Bonferroni inequality 2.2.7 and equality 6.4 follows from Lemma 6.2.5. □

We need the following basic fact.

Lemma 6.2.7. *The size of the image of a polynomial $h(X)$ corresponds to the number of $a_0 \in \mathbb{F}_p$ for which the polynomial $h(X) - a_0$ has a root in $\mathbb{F}_p$.*

Proof. If there exists an $x \in \mathbb{F}_p$ such that $h(x) = a_0$ then $h(X) - a_0$ has a root in $\mathbb{F}_p$ and vice versa. □

Lemma 6.2.8. *For any polynomial $h(X)$ and uniform $a_1 \xleftarrow{\$} \mathbb{F}_p$ it holds with probability $> 1 - 1/\sqrt{2}$ that $h(X) + a_1X$ has a image of size $> p(1 - 1/\sqrt{2})$.*

Proof. Fix a polynomial $h(X)$. By Lemma 6.2.6 we know that the number of $a_0, a_1 \in \mathbb{F}_p$ for which the polynomial $f(X) = a_0 + a_1X + h(X)$ has no root is at most $p^2/2$. Therefore, by a pidgeon-hole argument there are $< p/\sqrt{2}$ choices of for a_1 such that there exist $> p/\sqrt{2}$ choices of a_0 such that $a_0 + a_1X + h(X)$ has no root. This means that there are $> p - p/\sqrt{2}$ choices for a_1 such that $\leq p/\sqrt{2}$ choices for a_0 such that $a_0 + a_1X + h(X)$ has no root. Therefore, there are $> p - p/\sqrt{2}$ choices for a_1 such that $> p - p/\sqrt{2}$ choices for a_0 such that $a_0 + a_1X + h(X)$ has a root. The statement follows by Lemma 6.2.7. □

Lemma 6.2.9. *Fix a polynomial $h(X)$. The statistical distance between*

$$(a_1, h(x) + a_1x) \text{ and } (a_1, y)$$

where a_1, x is uniformly random over $\mathbb{F}_p$ and y is uniformly random from the image of $h(X) + a_1X$ is $< \sqrt{2} - 0.5$.

Proof. By Lemma 6.2.8 $h(X) + a_1X$ has a image of size $> p(1 - 1/\sqrt{2})$ with probability $1 - 1/\sqrt{2}$. For the rest of this analysis we assume to be in this case.

Picking a random x and evaluating $h(X) + a_1X$ on it is the same as sampling the output uniformly random from the multiset image of $h(X) + a_1X$ (the multiset where each element y has the multiplicity of the number of elements such that the element evaluates to y). This multiset has size p.

We now convert the multiset image to set by enumerating the multiplicities of each element and call this set $\mathcal{M}$. E.g. a multiset $\{8, 8, 8, 13, 55, 55\}$ would turn into a set of tuples $\{(8, 1), (8, 2), (8, 3), (13, 1), (55, 1), (55, 2)\}$. We do the same thing to the image of $h(X) + a_1X$ and call it $\mathcal{D}$. However because it is a set it only ever has multiplicity one.

So we would turn the set $\{8, 13, 55\}$ into $\{(8, 1), (13, 1), (55, 1)\}$. By the definition of these sets $\mathcal{D} \subseteq \mathcal{M}$. Because $\mathcal{M}$ is of size p and $\mathcal{D}$ is of size $> p(1 - 1/\sqrt{2})$ sampling a random element from $\mathcal{M}$ will be in $\mathcal{D}$ with probability $> 1 - 1/\sqrt{2}$. Therefore, for a $1 - 1/\sqrt{2}$ fraction of random choices in sampling x at random and then evaluating $h(x) + a_1 x$ behaves exactly as sampling uniformly random from the image. Thus, the statistical distance is $< 1 - (1 - 1/\sqrt{2})^2 = 1 - (1.5 - 2/\sqrt{2}) = \sqrt{2} - 0.5$. □

For our verifiable space-hard functions, we also want to have tight bound on the number of roots of these polynomials. A natural candidate for this is a distribution over permutation polynomials. In previous chapters we showed how to efficiently invert the specific set of Guralnick-Müller permutation polynomials [GM97], which [BBBF18] suggested as a time-lock puzzle. As we leveraged the specific structure of these polynomials for our attack, we believe it to be prudent to stay away from permutation polynomials.

If we instead use random sparse polynomials in our verfiable space-hard functions, we would want to have a good bound on the number of roots of these polynomials. The work of [Kel16] conjectures that the number of roots in a random sparse polynomial is $O(k \log p)$, which would be good enough for as this also implies that the probability of sampling a polynomial without roots is not overwhelming.

To have a lower probability of sampling a polynomial without roots that we can even prove, we suggest the distribution of polynomials which are the sum of a dense low-degree polynomial and a single high-degree monomial. We define the distribution $\mathcal{D}_{\lambda,S}$ as follows:

Definition 6.2.10 (Low-Degree Dense)**.** Pick a prime $p \in \Omega(2^\lambda)$, degree $d \in \Omega(S)$, and number of non-zero monomials k such that $k \log k - 2k \geq \lambda$. Operations happen over $\mathbb{F}_p$.

$$a_0 + \sum_{i \in [k-1]} a_i X^i + X^d$$

For uniformly random $a_i \xleftarrow{\$} \mathbb{F}_q$.

Lemma 6.2.11. *For a polynomial $f(X)$ sampled from distribution $\mathcal{D}_{\lambda,S}$ we have $f(X)$ has at least k (as in Definition 6.2.10) roots with probability $\leq 2^{-\lambda}$.*

Proof. A polynomial having k roots equivalent to the statement that there exists a set of λ distinct points $x_1, \ldots, x_k \in \mathbb{F}_p$ such that the polynomial evaluates to zero at these points. There are $\binom{p}{k}$ much sets. For each of those sets, the probability that the polynomial evaluates to zero at these points is p^{-k} by Lemma 6.2.5. Therefore, by union bound the probability that the polynomial evaluates to zero at any set of k distinct points is $\leq \frac{\binom{p}{k}}{p^k} \leq \frac{p(p-1)\cdots(p-k+1)}{k! \cdot p^k} \leq \frac{1}{k!} \leq 2^{-\lambda}$. The last inequality follows from Stirling's approximation and our choice of k relative to λ. □

6.3 Space-Lock Puzzle from SRF

We define space-lock puzzles analogously to time-lock puzzles but the resource we restrict is not sequential time but space.

Definition 6.3.1 (Space-Lock Puzzle)**.** A space-lock puzzle (SLP) with message space $\{0,1\}^n$ is a tuple of three algorithms $\mathsf{SLP} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Solve})$ defined as follows:

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, S)$: The setup algorithm Setup takes as input a security parameter 1^λ and a space bound S and outputs public parameters pp.

$\mathsf{p} \leftarrow \mathsf{Gen}(\mathsf{pp}, \mathsf{m})$: The puzzle generation algorithm Gen takes as input public parameters pp and a message m and outputs a puzzle p.

$\mathsf{m} \leftarrow \mathsf{Solve}(\mathsf{pp}, \mathsf{p})$: The solving algorithm Solve takes as input a puzzle p and outputs a message m.

Statistical Correctness : $\mathsf{SLP} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Solve})$ is statistically correct if for all polynomials $S(\cdot)$ there exists a negligible function negl s.t. for all $n, \lambda \in \mathbb{N}$ and $\mathsf{m} \in \{0,1\}^n$:

$$\Pr\left[\mathsf{m} \neq \mathsf{Solve}(\mathsf{pp}, \mathsf{p}) \middle| \begin{matrix} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, S(\lambda)) \\ \mathsf{p} \leftarrow \mathsf{Gen}(\mathsf{pp}, \mathsf{m}) \end{matrix}\right] \leq \mathsf{negl}(\lambda)$$

Efficiency : There exists a polynomial poly such that for all $n, \lambda, S \in \mathbb{N}$, and $\mathsf{m} \in \{0,1\}^n$ the runtime (and therefore space usage) of $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, S)$ and $\mathsf{p} \leftarrow \mathsf{Gen}(\mathsf{pp}, \mathsf{m})$ is $\leq \mathsf{poly}(\lambda, n, \log S)$.

Security : SLP is secure with gap $\varepsilon < 1$ if there exists a polynomial $\tilde{S}(\cdot)$ such that all polynomials $S(\cdot) \geq \tilde{S}(\cdot)$ and PPT adversaries $\{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ with space bound $S^\varepsilon(\lambda)$ there exists a negligible function negl s.t. for all $n, \lambda \in \mathbb{N}$:

$$\Pr\left[b = \mathcal{A}_\lambda(\mathsf{st}, \mathsf{p}) \middle| \begin{matrix} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, S(\lambda)) \\ (\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}) \leftarrow \mathcal{A}_\lambda(\mathsf{pp}) \\ b \xleftarrow{\$} \{0,1\} \\ \mathsf{p} \leftarrow \mathsf{Gen}(\mathsf{pp}, m_b) \end{matrix}\right] \leq 1/2 + \mathsf{negl}(\lambda)$$

We present the first space-lock puzzle based on the space hardness of finding roots of polynomials.

Construction 6.3.2 (Space-Lock Puzzle)**.** Let $\mathcal{D}_{\lambda,S}$ be a distribution of polynomial that are fast to evaluate and where finding roots is space-hard. Further, $\mathsf{H} : \mathbb{F}_q \to \{0,1\}^{\lambda+n}$, where n is the size of the message space then the following defines a space-lock puzzle.

$\mathsf{Setup}(1^\lambda, S)$: Return (λ, S).

$\mathsf{Gen}(\mathsf{pp}, \mathsf{m})$:

- Sample a degree S polynomial $f(X) \in \mathcal{D}_{\lambda,S}$.
- Sample a uniformly random element $z \xleftarrow{\$} \mathbb{F}_q$.
- Let $y = f(z)$.
- Return $(f(X), y, \mathsf{H}(z) \oplus (0^\lambda || m))$.

$\mathsf{Solve}(\mathsf{p} = (f(x), y, c))$:

- Compute Z, the set of roots of the polynomial $f(X) - y$.
- For $z^* \in Z$:
  - Compute $\tilde{m} \leftarrow \mathsf{H}(z^*) \oplus c$.
  - If the first λ bits of $\tilde{m}$ are all 0 return the rest of $\tilde{m}$

Theorem 6.3.3. *Construction 6.3.2 is a space-lock puzzle under the assumption that the SRF assumption 6.2.1 holds for polynomials with uniform linear coefficient.*

Proof. The proof follows from lemmas 6.3.4 to 6.3.6. □

Lemma 6.3.4 (Statistical Correctness)**.** *Construction 6.3.2 is statistically correct.*

Proof. Because $f(X) - y$ has degree S it has at most S roots. We know that $f(z) = y$, therefore, z is a root of $f(X) - y$. Because H is a random oracle we have that for all $z^* \in Z \setminus \{z\}$ the probability that the first λ many bits of $\tilde{m}$ are 0^λ is $2^{-\lambda}$. Therefore, Solve outputs m with all but probability $\leq S(\lambda) \cdot 2^{-\lambda}$, which is negligible in λ. □

Lemma 6.3.5 (Efficiency)**.** *Construction 6.3.2 is efficient.*

Proof. The properties of $\mathcal{D}_{\lambda,S}$ tell us that evaluating $f(X)$ on z can be done in $\mathsf{poly}(\lambda, \log S)$. It follows that Gen runs in time and space $\mathsf{poly}(\lambda, \log S)$. □

Lemma 6.3.6 (Security)**.** *Construction 6.3.2 is secure under the SRF assumption 6.2.1 for polynomials with uniform linear coefficient.*

Proof. To break security of the Construction 6.3.2 the adversary has to compute z given $f(X)$ and y otherwise he has no way of computing $\mathsf{H}(z)$.

We fix an $f(X)$. In Construction 6.3.2 the adversary has to compute the root of a polynomial following the distribution $f(X) - f(x)$ for uniformly random x. In the assumption the adversary has to compute a root of a polynomial $f(X) - y$, where y is uniformly at random. Because of the polynomial identity lemma we have $f(X) - f(x)$ can have at most d roots, where d is the degree of $f(X) - f(x)$. Therefore, for every y^* we have $\Pr_{x \xleftarrow{\$} \mathbb{F}}[f(x) = y^*] \leq d/|\mathbb{F}| = d \cdot \Pr[y = y^*]$. Therefore, if an adversary compute a root for $f(X) - f(x)$ with probability ε then it can compute a root for $f(X) - y$ with probability $\geq \varepsilon/d$. □

6.4 Verifiable Space-Hard Function from SRF

The definition of verifiable space hard function is similar to the definition of verifiable delay function but instead of a sequential time bound we have a space bound.

Definition 6.4.1 (Verifiable Space-Hard Function)**.** A verifiable space-hard function (VSHF) with domain space $\mathbb{X}$, codomain $\mathbb{Y}$, and proof space Π has the following algorithms:

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, S)$: The setup algorithm Setup takes as input a security parameter λ and a space bound S and outputs public parameters pp.

$(y, \pi) \leftarrow \mathsf{Eval}(\mathsf{pp}, x)$: The evaluation deterministic algorithm Eval takes as input public parameters pp and outputs a function output y and a proof $\pi \in \Pi$.

$b \leftarrow \mathsf{Verify}(\mathsf{pp}, x, y, \pi)$: The verification algorithm Verify takes as input public parameters pp, a function input $x \in \mathbb{X}$, a function output y and a proof $\pi \in \Pi$ and outputs a bit b.

it has the following properties:

Correctness : $\mathsf{VSHF} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ is correct if for all $\lambda, S \in \mathbb{N}$ and $x \in \mathbb{X}$ we have $\mathsf{Verify}(\mathsf{pp}, x, y, \pi) = 1$ for $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, S)$ and $(y, \pi) \leftarrow \mathsf{Eval}(\mathsf{pp}, x)$.

Space Hardness : $\mathsf{VSHF} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ is sound with entropy E and a gap $\varepsilon < 1$ if there exists a polynomial $\tilde{S}(\cdot)$ such that for all polynomials $S(\cdot) \geq \tilde{S}(\cdot)$ and PPT adversaries $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ with space bound $S^\varepsilon(\lambda)$ there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$, $x \in \mathbb{X}$:

$$\Pr\left[\mathsf{Eval}(\mathsf{pp}, x) = (y, \pi) \middle| \begin{matrix} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, S(\lambda)) \\ (x, y) \leftarrow \mathcal{A}_\lambda(\mathsf{pp}) \end{matrix}\right] \leq 2^{-E} + \mathsf{negl}(\lambda)$$

Computational Uniqueness : $\mathsf{VSHF} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ is computationally unique if for all PPT adversaries $\mathcal{A}$ there exists a negligible function negl s.t. for all $\lambda \in \mathbb{N}$ and $x \in \mathbb{X}$:

$$\Pr\left[\mathsf{Verify}(\mathsf{pp}, x, y^*, \pi^*) = 1 \wedge y^* \neq y \middle| \begin{matrix} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, S(\lambda)) \\ (x, y^*, \pi^*) \leftarrow \mathcal{A}_\lambda(\mathsf{pp}) \\ (y, \pi) \leftarrow \mathsf{Eval}(\mathsf{pp}, x) \end{matrix}\right] \leq \mathsf{negl}(\lambda)$$

Efficiency : There exists a polynomial poly such that for all $\lambda, S \in \mathbb{N}$, $\pi \in \Pi$, and $x \in \mathbb{X}$ the runtime (and therefore space usage) of $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, S)$ and $\mathsf{Verify}(\mathsf{pp}, x, y, \pi)$ is $\leq \mathsf{poly}(\lambda, \log S)$.

We now show how to construct a verifiable space-hard function. We follow the same basic idea as the weak verifiable delay function of [BBBF18] but we require more care because we do not rely on permutation polynomials.

We present the function as an interactive proof, but the proof can be made non-interactive using the Fiat-Shamir heuristic.

Construction 6.4.2 (Verifiable Space-Hard Function)**.** Let $(\mathsf{Setup}', \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify}')$ be an extractable polynomial commitment scheme.

$\mathsf{Setup}(S, 1^\lambda)$:

- Output $\mathsf{pp} = \mathsf{Setup}'(1^\lambda, d(S))$, where d is the degree of polynomials required to achieve space-hardness S.

$\mathsf{Eval}(S, x)$:

- Let $f(X) = \mathsf{H}(x)$.
- Let the degree of $f(X)$ be d.
- Let $(p_i)_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$ be the binary decomposition of p.
- Let $m_0(X) = X$ and $v_0(X) = X^{p_0}$.
- For $i \in [\lfloor\log p\rfloor]$:
  - Compute $m_i(X) = m_{i-1}(X) \cdot m_{i-1}(X) \mod f(X)$.
  - Compute $e_i(X) = (m_{i-1}(X) \cdot m_{i-1}(X) - m_i(X))/f(X)$
  - Compute $v_i(X) = v_{i-1}(X) \cdot m_i^{p_i}(X) \mod f(X)$.
  - Compute $w_i(X) = (v_{i-1}(X) \cdot m_i^{p_i}(X))/f(X)$
- Compute $g(X)$, the gcd of $v_{\lfloor\log p\rfloor}(X) - X$ and $f(X)$ together with their Bézout coefficients $a(X), b(X)$.
- Commit to
  - the degree $d-1$ polynomials

    - $(m_i(X))_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$,
    - $(v_i(X))_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$,
    - and $a(X)$,
  - the degree $d-2$ polynomials
    - $(e_i(X))_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$,
    - $(w_i(X))_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$,
    - and $b(X)$

  with Commit to get polynomial commitments

  - $(\hat{m}_i(X))_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$,
  - $(\hat{v}_i(X))_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$,
  - $\hat{a}(X)$,
  - $(\hat{e}_i(X))_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$,
  - $(\hat{w}_i(X))_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$,
  - and $\hat{b}(X)$.

  Further, send these commitments to the verifier.
- The verifier responds with a uniformly random field element r.
- Evaluate the polynomials at r and compute corresponding openings $(o_{m_i(r)})_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$, $(o_{v_i(r)})_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$, $o_{a(r)}$, $(o_{e_i(r)})_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$, $(o_{w_i(r)})_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$, and $o_{b(r)}$ via Open.
- Return function output $y = \frac{g(X)}{\text{leading coefficient of } g(X)}$ and final round of the proof system $(m_i(r))_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$, $(v_i(r))_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$, $a(r)$, $(e_i(r))_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$, $(w_i(r))_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$, $b(r)$, $(o_{m_i(r)})_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$, $(o_{v_i(r)})_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$, $o_{a(r)}$, $(o_{e_i(r)})_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$, $(o_{w_i(r)})_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$, and $o_{b(r)}$.

$\mathsf{Verify}(x, y = g(X))$:

- Let $f(X) = \mathsf{H}(x)$.
- If $g(X)$ is not monic return 0.
- Run V to verify that $c(X)$ is at a constant relative distance from a polynomial of degree d.
- Let $(p_i)_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$ be the binary decomposition of p.
- Sample a random set $R \subset \mathcal{L}$ of size λ.
- For $r \in R$:
  - Read $m_0(r)$ and $v_0(r)$ from the prover string.
  - If $m_0(r) \neq r$ return 0.
  - If $v_0(r) \neq r^{p_0}$ return 0.
- For $i \in [\lfloor\log p\rfloor]$:
  - For $r \in R$:
    - Read $(m_{i-1}, m_i(r), e_i(r), v_{i-1}(r), v_i(r), w_i(r))$ from the prover string.
    - If $m_{i-1}(r)^2 \neq m_i(r) + e_i(r) \cdot f(r)$ return 0.
    - If $v_{i-1}(r) \cdot m_i(r)^{p_i} \neq v_i(r) + w_i(r) \cdot f(r)$ return 0.
- For $r \in R$:

  - If $a(r) \cdot (v_{\lfloor \log p \rfloor}(r) - r) + b(r) \cdot f(r) \neq g(r)$ return 0.
- If $f(X) \mod g(X) \neq 0$ or $X^p - X \mod g(X) \neq 0$ return 0.
- Return 1.

Remark 6.4.3. Note, the above function does not have high output entropy because $f(X)$ does not have any roots with $\leq 1/2$ probability. This is required by many applications and can easily be fixed by repetition.

Theorem 6.4.4. *Construction 6.4.2 is a verifiable space-hard function.*

Proof. Follows from Lemmas 6.4.5 to 6.4.7 and Corollary 6.4.8. □

Lemma 6.4.5. *[Correctness] Construction 6.4.2 is correct.*

Proof. Because the prover divides $g(X)$ by its leading coefficient it outputs a monic polynomial. All the checks made by V pass, which follows from the correctness of (P, V). For $i \in \{0\} \cup [\lfloor \log p \rfloor]$ it holds that $m_i(X) = X^{2^i} \mod f(X)$. We also have for $i \in [\lfloor \log p \rfloor]$ it holds $m_{i-1}(X)^2 = m_i(X) + f(X)e_i(X)$. Similarly, for binary decomposition $(p_i)_{i \in \{0\} \cup [\lfloor \log p \rfloor]}$ of p we have

$$v_i(X) = \prod_{i \in \{\} \cup [\lfloor \log p \rfloor]} X^{2^i p_i} \mod f(X)$$

and

$$w_i(X) = \prod_{i \in \{\} \cup [\lfloor \log p \rfloor]} X^{2^i p_i} / v_i(X).$$

Therefore, evaluating all these polynomial at the point r still makes these equations hold. □

By definition of a common divisor $g(X)$ divides $v_{\lfloor \log p \rfloor}(X) - X$ and $f(X)$. Because $v_{\lfloor \log p \rfloor}(X) - X = X^p - X \mod f(X)$ we get if $g(X)$ divides $v_{\lfloor \log p \rfloor}(X) - X$ and $f(X)$ it also divides $X^p - X$. The Bézout coefficients have the property that $a(X) \cdot (v_{\lfloor \log p \rfloor}(X) - X) + b(X) \cdot f(X) = g(X)$. Therefore, each check passes if π is honestly generated.

Lemma 6.4.6. *[Efficiency] Construction 6.4.2 is efficient.*

Proof. The prover's operations are all in $\mathsf{poly}(d, \lambda)$. The rest of the proof regards itself with the efficiency of the verfiier.

We require from $\mathcal{D}_{\lambda,S}$ that with all but negligible probability $f(X)$ has a polynomial number of roots. For polynomials where the λ low-order monomials are dense (see Definition 6.2.10) this follows from Lemma 6.2.11. Therefore, the degree of $g(X)$ is polynomial in λ with overwhelming probability. The verifier has to check polynomially many opening and $\mathbb{F}_p$ equations. Therefore, the verifier is polynomial in $\log(d)$ and λ if checking an opening is polynomial in $\log(d)$. □

Lemma 6.4.7. *[Computational Uniqueness] The Construction 6.4.2 is computationally unique.*

Proof. First, we extract the polynomials

- $(m_i(X))_{i \in \{0\} \cup [\lfloor \log p \rfloor]}$,
- $(v_i(X))_{i \in \{0\} \cup [\lfloor \log p \rfloor]}$,
- $a(X)$,

- $(e_i(X))_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$,
- $(w_i(X))_{i\in\{0\}\cup[\lfloor\log p\rfloor]}$,
- and $b(X)$

from their respective commitments and openings at r.

Then we check each polynomial equation the polynomial identity lemma [2.2.8]. Each polynomial equation we check is at most of degree $2d$. Therefore, we get if a polynomial equation does not hold then evaluating the polynomial at a random point on in $\mathbb{F}_p$ and then catches this with probability $1 - 2d/|\mathbb{F}_p|$, which is overwhelming.

From these equations follows that

$$v_{\lfloor\log p\rfloor}(X) - X = X^p - X \mod f(X).$$

Therefore, $gcd(v_{\lfloor\log p\rfloor}(X) - X, f(X)) = gcd(X^p - X, f(X))$. By Bézout's identity for all $a'(X)$, $b'(X)$ we have

$$a'(X) \cdot (\hat{v}_{\lfloor\log p\rfloor}(X) - X) + b'(X) \cdot f(X)$$

is a multiple of $gcd(\hat{v}_{\lfloor\log p\rfloor}(X), f(X))$. So, the check $a(X) \cdot (\hat{v}_{\lfloor\log p\rfloor}(X) - X) + b(X) \cdot f(X) = g(X)$ verifies that $g(X)$ is a multiple of $gcd(X^p - X, f(X))$. The checks $f(X) \mod g(X) = 0$ and $X^p - X \mod g(X) = 0$ verify that $g(X)$ is a divisor of $f(X)$ and $X^p - X$. Therefore, $g(X)$ is a greatest common divisor of $f(X)$ and $X^p - X$. The gcd is unique up to multiplication by a field element, which is why we require $g(X)$ to be monic. □

Corollary 6.4.8 (Space Hardness)**.** *Construction 6.4.2 is space-hard.*

Proof. Under Assumption 6.2.2 for polynomials that are dense in the low degrees Definition 6.2.10 space-hardness with entropy 1 follows directly from computational uniqueness and the fact that the polynomial has a root with probability $> 1/2$ (Lemma 6.2.8). □

Chapter 7

Final Remarks

7.1 Conclusion

This thesis explored how space and time, two fundamental computational resources, interact in cryptographic protocols. In contrast to standard perspective in which resources are qualitatively, we adopted a quantitative perspective throughout, highlighting how simultaneous constraints on space and time give rise to new limitations, constructions, and security notions.

We began by studying communication-computation trade-offs in secure two-party computation. Focusing on the fundamental primitive of oblivious transfer, we showed that reducing communication in OT protocols often necessitates increasing computational cost. In Chapter 3, we formalized this intuition by proving a lower bound on the number of expensive public-key operations required by communication-efficient OT protocols, thereby uncovering an inherent space-time trade-off in the OT and PIR domains.

In Chapter 4, we examined proof systems and developed designated-verifier SNARGs that minimize proof size. We achieved proofs significantly shorter than existing SNARGs by using techniques from trapdoor hashing. This results in a clear trade-off: reduced communication (space) at the cost of increased verification time. Our constructions demonstrate that, even within proof systems, the balance between time and space plays a critical role in practical protocol design.

Chapter 5 investigated the security limits of encryption in adversarial models where the attacker eventually learns the secret key. We constructed incompressible encryption schemes that retain meaningful security by assuming the adversary cannot store the entire ciphertext until the key is revealed. These schemes fundamentally rely on space-time asymmetry: without bounding both resources, security collapses.

Finally, in Chapter 6, we introduced new primitives in the domain of space-hard cryptography, analogous to well-known time-hard constructions such as time-lock puzzles and verifiable delay functions. We conjectured the space-hardness of root-finding in sparse polynomials and used it to construct the first verifiable space-hard functions and space-lock puzzles. These results further support the thesis that space, like time, can serve as a meaningful and verifiable resource in cryptographic design.

7.2 Outlook

Taken together, these results enrich our understanding of how space and time interact in cryptographic settings. They suggest a broader research agenda: to study cryptographic security under fine-grained resource models, and to ask not just whether something is possible, but at what resource cost and under which trade-offs.

There remain many exciting open questions. For instance:

- Can we expand our techniques for lower bounds to other setting and use them to learn more about different cryptographic protocols?
- Are our dv-SNARGs the limit? Is it possible to build even smaller dv-SNARGs or drastically improve practicality of our constructions?
- Is it possible to build incompressible symmetric-key encryption just form symmetric-key cryptography?
- Can we find exciting applications to our enhanced space-hard cryptographic primitives and increase their functionality?

This thesis contributes to a growing line of work that asks the question: What remains secure when adversaries and honest parties are bounded not just in computation, but in space and time?

Bibliography

[AB16] Joël Alwen and Jeremiah Blocki. Efficiently computing data-independent memory-hard functions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 241–271, Santa Barbara, CA, USA, August 14–18, 2016. Springer Berlin Heidelberg, Germany.

[ABB22] Mohammad Hassan Ameri, Alexander R. Block, and Jeremiah Blocki. Memory-hard puzzles in the standard model with applications to memory-hard functions and resource-bounded locally decodable codes. In Clemente Galdi and Stanislaw Jarecki, editors, *SCN 22*, volume 13409 of *LNCS*, pages 45–68, Amalfi, Italy, September 12–14, 2022. Springer, Cham, Switzerland.

[ABBK16] Frederik Armknecht, Ludovic Barman, Jens-Matthias Bohli, and Ghassan O. Karame. Mirror: Enabling proofs of data replication and retrievability in the cloud. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 1051–1068, Austin, TX, USA, August 10–12, 2016. USENIX Association.

[ABCH19] Per Austrin, Jonah Brown-Cohen, and Johan Håstad. Optimal inapproximability with universal factor graphs. *ACM Transactions on Algorithms*, 2019.

[ABFG14] Giuseppe Ateniese, Ilario Bonacina, Antonio Faonio, and Nicola Galesi. Proofs of space: When space is of the essence. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 538–557, Amalfi, Italy, September 3–5, 2014. Springer, Cham, Switzerland.

[ABP15] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Disjunctions for hash proof systems: New constructions and applications. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 69–100, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany.

[ABP18] Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Sustained space complexity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 99–130, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

[ACFY24a] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. STIR: Reed-solomon proximity testing with fewer queries. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 380–413, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[ACFY24b] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. WHIR: reed-solomon proximity testing with super-fast verification. *IACR Cryptol. ePrint Arch.*, page 1586, 2024.

[ACP+17] Joël Alwen, Binyi Chen, Krzysztof Pietrzak, Leonid Reyzin, and Stefano Tessaro. Scrypt is maximally memory-hard. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 33–62, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland.

[ACY23] Gal Arnon, Alessandro Chiesa, and Eylon Yogev. IOPs with inverse polynomial soundness error. In *64th FOCS*, pages 752–761, Santa Cruz, CA, USA, November 6–9, 2023. IEEE Computer Society Press.

[ADD+22] Divesh Aggarwal, Nico Döttling, Jesko Dujmovic, Mohammad Hajiabadi, Giulio Malavolta, and Maciej Obremski. Algebraic restriction codes and their applications. In Mark Braverman, editor, *ITCS 2022*, volume 215, pages 2:1–2:15, Berkeley, CA, USA, January 31 – February 3, 2022. LIPIcs.

[ADI25] Gal Arnon, Jesko Dujmovic, and Yuval Ishai. Designated-verifier snargs with one group element. *CRYPTO 2025*, 2025.

[ADMP20] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 411–439, Daejeon, South Korea, December 7–11, 2020. Springer, Cham, Switzerland.

[ADN+10] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 113–134, French Riviera, May 30 – June 3, 2010. Springer Berlin Heidelberg, Germany.

[AFLN24] Martin R. Albrecht, Giacomo Fenzi, Oleksandra Lapiha, and Ngoc Khanh Nguyen. SLAP: Succinct lattice-based polynomial commitments from standard assumptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 90–119, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[AGP24] Benedikt Auerbach, Christoph U. Günther, and Krzysztof Pietrzak. Trapdoor memory-hard functions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part III*, volume 14653 of *LNCS*, pages 315–344, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[AHRS01] Yonatan Aumann, Johan Håstad, Michael O. Rabin, and Madhu Sudan. Linear-consistency testing. *J. Comput. Syst. Sci.*, 62(4):589–607, 2001.

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *33rd FOCS*, pages 14–23, Pittsburgh, PA, USA, October 24–27, 1992. IEEE Computer Society Press.

[AR99] Yonatan Aumann and Michael O. Rabin. Information theoretically secure communication in the limited storage space model. In Michael J. Wiener, editor,

CRYPTO'99, volume 1666 of *LNCS*, pages 65–79, Santa Barbara, CA, USA, August 15–19, 1999. Springer Berlin Heidelberg, Germany.

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; A new characterization of NP. In *33rd FOCS*, pages 2–13, Pittsburgh, PA, USA, October 24–27, 1992. IEEE Computer Society Press.

[AS15a] Joël Alwen and Vladimir Serbinenko. High parallel complexity graphs and memory-hard functions. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 595–603, Portland, OR, USA, June 14–17, 2015. ACM Press.

[AS15b] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 191–209, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 315–334. IEEE Computer Society, 2018.

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 757–788, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.

[BBCE25] Joseph Bonneau, Benedikt Bünz, Miranda Christ, and Yuval Efron. Good things come to those who wait - dishonest-majority coin-flipping requires delay functions. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part VII*, volume 15607 of *Lecture Notes in Computer Science*, pages 225–253. Springer, 2025.

[BBD+20] Zvika Brakerski, Pedro Branco, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Constant ciphertext-rate non-committing encryption from standard assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 58–87, Durham, NC, USA, November 16–19, 2020. Springer, Cham, Switzerland.

[BBDP22] Zvika Brakerski, Pedro Branco, Nico Döttling, and Sihang Pu. Batch-OT with optimal rate. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 157–186, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17, Prague, Czech Republic, July 9–13, 2018. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 171–188, Interlaken, Switzerland, May 2–6, 2004. Springer Berlin Heidelberg, Germany.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357, Vienna, Austria, May 8–12, 2016. Springer Berlin Heidelberg, Germany.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 111–120, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333, Tokyo, Japan, March 3–6, 2013. Springer Berlin Heidelberg, Germany.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60, Beijing, China, October 31 – November 3, 2016. Springer Berlin Heidelberg, Germany.

[BDD22] Pedro Branco, Nico Döttling, and Jesko Dujmovic. Rate-1 incompressible encryption from standard assumptions. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 33–69, Chicago, IL, USA, November 7–10, 2022. Springer, Cham, Switzerland.

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 407–437, Nuremberg, Germany, December 1–5, 2019. Springer, Cham, Switzerland.

[BDS23] Pedro Branco, Nico Döttling, and Akshayaram Srinivasan. A framework for statistically sender private OT with optimal rate. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 548–576, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th ACM STOC*, pages 479–488, Philadephia, PA, USA, May 22–24, 1996. ACM Press.

[Ber70] Elwyn R Berlekamp. Factoring polynomials over large finite fields. *Mathematics of computation*, 24(111):713–735, 1970.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *23rd ACM STOC*, pages 21–31, New Orleans, LA, USA, May 6–8, 1991. ACM Press.

[BFM15] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Random-oracle uninstantiability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 428–455, Warsaw, Poland, March 23–25, 2015. Springer Berlin Heidelberg, Germany.

[BG25] Nir Bitansky and Rachit Garg. Succinct randomized encodings from laconic function evaluation, faster and simpler. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 406–436. Springer, 2025.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer Berlin Heidelberg, Germany.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539, Santa Barbara, CA, USA, August 14–18, 2016. Springer Berlin Heidelberg, Germany.

[BGI17] Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 163–193, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland.

[BGJ+16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In Madhu Sudan, editor, *ITCS 2016*, pages 345–356, Cambridge, MA, USA, January 14–16, 2016. ACM.

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 439–448, Portland, OR, USA, June 14–17, 2015. ACM Press.

[BHI+24] Nir Bitansky, Prahladh Harsha, Yuval Ishai, Ron D. Rothblum, and David J. Wu. Dot-product proofs and their applications. In *65th FOCS*, pages 806–825, Chicago, IL, USA, October 27–30, 2024. IEEE Computer Society Press.

[BIM00] Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers computation in private information retrieval: PIR with preprocessing. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 55–73, Santa Barbara, CA, USA, August 20–24, 2000. Springer Berlin Heidelberg, Germany.

[BIOW20] Ohad Barta, Yuval Ishai, Rafail Ostrovsky, and David J. Wu. On succinct arguments and witness encryption from groups. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*,

pages 776–806, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 662–693, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 247–277, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland.

[BISW18] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal SNARGs via linear multi-prover interactive proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 222–255, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

[BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 738–767, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.

[BL95] Dan Boneh and Richard J. Lipton. Quantum cryptanalysis of hidden linear functions (extended abstract). In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 424–437, Santa Barbara, CA, USA, August 27–31, 1995. Springer Berlin Heidelberg, Germany.

[Bla06] John Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In Matthew J. B. Robshaw, editor, *FSE 2006*, volume 4047 of *LNCS*, pages 328–340, Graz, Austria, March 15–17, 2006. Springer Berlin Heidelberg, Germany.

[BLR90] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. In *22nd ACM STOC*, pages 73–83, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

[BN00] Dan Boneh and Moni Naor. Timed commitments. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 236–254, Santa Barbara, CA, USA, August 20–24, 2000. Springer Berlin Heidelberg, Germany.

[Bon36] Carlo Bonferroni. Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, 8:3–62, 1936.

[BP17] Alex Biryukov and Léo Perrin. Symmetrically and asymmetrically hard cryptography. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 417–445, Hong Kong, China, December 3–7, 2017. Springer, Cham, Switzerland.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[BS23] Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 518–548, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[CFPZ09] Céline Chevalier, Pierre-Alain Fouque, David Pointcheval, and Sébastien Zimmer. Optimal randomness extraction from a Diffie-Hellman element. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 572–589, Cologne, Germany, April 26–30, 2009. Springer Berlin Heidelberg, Germany.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, jul 2004.

[CGH+21] Melissa Chase, Sanjam Garg, Mohammad Hajiabadi, Jialin Li, and Peihan Miao. Amortizing rate-1 OT and applications to PIR and PSI. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 126–156, Raleigh, NC, USA, November 8–11, 2021. Springer, Cham, Switzerland.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th FOCS*, pages 41–50, Milwaukee, Wisconsin, October 23–25, 1995. IEEE Computer Society Press.

[CGN98] Benny Chor, Niv Gilboa, and Moni Naor. Private information retrieval by keywords. Cryptology ePrint Archive, Report 1998/003, 1998.

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA.

[CHK22] Henry Corrigan-Gibbs, Alexandra Henzinger, and Dmitry Kogan. Single-server private information retrieval with sublinear amortized time. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 3–33, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

[CHR17] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 694–726, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.

[CK20] Henry Corrigan-Gibbs and Dmitry Kogan. Private information retrieval with sublinear online time. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 44–75, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *18th ACM STOC*, pages 364–369, Berkeley, CA, USA, May 28–30, 1986. ACM Press.

[CLSY93] J-Y Cai, Richard J Lipton, Robert Sedgewick, and AC-C Yao. Towards uncheatable benchmarks. In *[1993] Proceedings of the Eigth Annual Structure in Complexity Theory Conference*, pages 2–11. IEEE, 1993.

[CM97] Christian Cachin and Ueli M. Maurer. Unconditional security against memory-bounded adversaries. In Burton S. Kaliski, Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 292–306, Santa Barbara, CA, USA, August 17–21, 1997. Springer Berlin Heidelberg, Germany.

[CM16] Jing Chen and Silvio Micali. Algorand. *arXiv preprint arXiv:1607.01341*, 2016.

[CM20] Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 34–63, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.

[CMNW24] Valerio Cini, Giulio Malavolta, Ngoc Khanh Nguyen, and Hoeteck Wee. Polynomial commitments from lattices: Post-quantum security, fast verification and transparent setup. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 207–242, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 402–414, Prague, Czech Republic, May 2–6, 1999. Springer Berlin Heidelberg, Germany.

[CP18] Bram Cohen and Krzysztof Pietrzak. Simple proofs of sequential work. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 451–467, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25, Santa Barbara, CA, USA, August 23–27, 1998. Springer Berlin Heidelberg, Germany.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer Berlin Heidelberg, Germany.

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[CY24] Alessandro Chiesa and Eylon Yogev. Building cryptographic proofs from hash functions. *URL: https://github. com/hash-based-snargs-book*, 2024.

[CZ81] David G Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36(154):587–592, 1981.

[DD22] Nico Döttling and Jesko Dujmovic. Maliciously circuit-private FHE from information-theoretic principles. In Dana Dachman-Soled, editor, *ITC 2022*, volume 230 of *LIPIcs*, pages 4:1–4:21, Cambridge, MA, USA, July 5–7, 2022. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

[DDJ24] Nico Döttling, Jesko Dujmovic, and Antoine Joux. Space-lock puzzles and verifiable space-hard functions from root-finding in sparse polynomials. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024, Part III*, volume 15366 of *LNCS*, pages 431–459, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.

[DDLO25] Nico Döttling, Jesko Dujmovic, Julian Loss, and Maciej Obremski. Minicrypt pir for big batches. *Cryptology ePrint Archive*, 2025.

[Den02] Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 100–109, Queenstown, New Zealand, December 1–5, 2002. Springer Berlin Heidelberg, Germany.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer Berlin Heidelberg, Germany.

[DFKP15] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 585–605, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany.

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 3–32, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

[DGM24] Jesko Dujmovic, Rachit Garg, and Giulio Malavolta. Time-lock puzzles with efficient batch solving. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 311–341, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[DGO19] Ivan Damgård, Chaya Ganesh, and Claudio Orlandi. Proofs of replicated storage without timing assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 355–380, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

[DGP25] Jesko Dujmovic, Christoph U. Günther, and Krzysztof Pietrzak. Space-deniable proofs. *Under Submission*, 2025.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DH24] Jesko Dujmovic and Mohammad Hajiabadi. Lower-bounds on public-key operations in PIR. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 65–87, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 119–136, Cheju Island, South Korea, February 13–15, 2001. Springer Berlin Heidelberg, Germany.

[DKK18] Itai Dinur, Nathan Keller, and Ohad Klein. An optimal distributed discrete log protocol with applications to homomorphic secret sharing. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 213–242, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.

[DL78] Richard A. DeMillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7(4):193–195, 1978.

[DLM19] Nico Döttling, Russell W. F. Lai, and Giulio Malavolta. Incremental proofs of sequential work. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 292–323, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland.

[DLW06] Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 225–244, New York, NY, USA, March 4–7, 2006. Springer Berlin Heidelberg, Germany.

[DM04] Stefan Dziembowski and Ueli M. Maurer. On generating the initial key in the bounded-storage model. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 126–137, Interlaken, Switzerland, May 2–6, 2004. Springer Berlin Heidelberg, Germany.

[DMO00] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. Single database private information retrieval implies oblivious transfer. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 122–138, Bruges, Belgium, May 14–18, 2000. Springer Berlin Heidelberg, Germany.

[DMPS19] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable delay functions from supersingular isogenies and pairings. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 248–277, Kobe, Japan, December 8–12, 2019. Springer, Cham, Switzerland.

[DMQ24] Jesko Dujmovic, Giulio Malavolta, and Wei Qi. Registration-based encryption in the plain model. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025*, LNCS. Springer, Cham, Switzerland, May 10–13, 2024.

[DMS24] Michel Dellepere, Pratyush Mishra, and Alireza Shirzad. Garuda and pari: Faster and smaller SNARKs via equifficient polynomial commitments. Cryptology ePrint Archive, Report 2024/1245, 2024.

[DN93] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 139–147, Santa Barbara, CA, USA, August 16–20, 1993. Springer Berlin Heidelberg, Germany.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.

[DQW21] Yevgeniy Dodis, Willy Quach, and Daniel Wichs. Speak much, remember little: Cryptography in the bounded storage model, revisited. Cryptology ePrint Archive, Report 2021/1270, 2021.

[Dzi06a] Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 207–224, New York, NY, USA, March 4–7, 2006. Springer Berlin Heidelberg, Germany.

[Dzi06b] Stefan Dziembowski. On forward-secure storage (extended abstract). In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 251–270, Santa Barbara, CA, USA, August 20–24, 2006. Springer Berlin Heidelberg, Germany.

[EG24] Martin Ekerå and Joel Gärtner. Extending regev's factoring algorithm to compute discrete logarithms. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*, pages 211–242, Oxford, UK, June 12–14, 2024. Springer, Cham, Switzerland.

[FGL+96] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM (JACM)*, 43(2):268–292, 1996.

[FGM+89] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. *Adv. Comput. Res.*, 5:429–442, 1989.

[FJ12] Uriel Feige and Shlomo Jozeph. Universal factor graphs. In *Automata, Languages, and Programming: 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I 39*, pages 339–350. Springer, 2012.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999. Springer Berlin Heidelberg, Germany.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer Berlin Heidelberg, Germany.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press.

[GGK03] Rosario Gennaro, Yael Gertner, and Jonathan Katz. Lower bounds on the efficiency of encryption and digital signature schemes. In *35th ACM STOC*, pages 417–425, San Diego, CA, USA, June 9–11, 2003. ACM Press.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645, Athens, Greece, May 26–30, 2013. Springer Berlin Heidelberg, Germany.

[GHMM18] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ameer Mohammed. Limits on the power of garbling techniques for public-key encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 335–364, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.

[GHO20] Sanjam Garg, Mohammad Hajiabadi, and Rafail Ostrovsky. Efficient range-trapdoor functions and applications: Rate-1 OT and more. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 88–116, Durham, NC, USA, November 16–19, 2020. Springer, Cham, Switzerland.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pages 102–115, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press.

[GKM+00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st FOCS*, pages 325–335, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th FOCS*, pages 612–621, Berkeley, CA, USA, October 15–17, 2017. IEEE Computer Society Press.

[GLW20] Rachit Garg, George Lu, and Brent Waters. New techniques in replica encodings with client setup. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 550–583, Durham, NC, USA, November 16–19, 2020. Springer, Cham, Switzerland.

[GM97] Robert M Guralnick and Peter Müller. Exceptional polynomials of affine type. *Journal of Algebra*, 194(2):429–454, 1997.

[GMM17] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. Lower bounds on obfuscation from all-or-nothing encryption primitives. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 661–695, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland.

[GMMM18] Sanjam Garg, Mohammad Mahmoody, Daniel Masny, and Izaak Meckler. On the round complexity of OT extension. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 545–574, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GNU16] Zeyu Guo, Anand Kumar Narayanan, and Chris Umans. Algebraic problems equivalent to beating exponent 3/2 for polynomial factorization over finite fields. *arXiv preprint arXiv:1606.04592*, 2016.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340, Singapore, December 5–9, 2010. Springer Berlin Heidelberg, Germany.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326, Vienna, Austria, May 8–12, 2016. Springer Berlin Heidelberg, Germany.

[GT00] Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st FOCS*, pages 305–313, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179, Santa Barbara, CA, USA, August 19–23, 2012. Springer Berlin Heidelberg, Germany.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108, San Jose, CA, USA, June 6–8, 2011. ACM Press.

[GWZ22] Jiaxin Guan, Daniel Wichs, and Mark Zhandry. Incompressible cryptography. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 700–730, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

[GWZ23] Jiaxin Guan, Daniel Wichs, and Mark Zhandry. Multi-instance randomness extraction and security against bounded-storage mass surveillance. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part III*, volume 14371 of *LNCS*, pages 93–122, Taipei, Taiwan, November 29 – December 2, 2023. Springer, Cham, Switzerland.

[GZ19] Jiaxin Guan and Mark Zhandry. Simple schemes in the bounded storage model. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 500–524, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland.

[GZ21] Jiaxin Guan and Mark Zhandry. Disappearing cryptography in the bounded storage model. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 365–396, Raleigh, NC, USA, November 8–11, 2021. Springer, Cham, Switzerland.

[Hås01] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

[HHC+23] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023*, pages 3889–3905, Anaheim, CA, USA, August 9–11, 2023. USENIX Association.

[HHK+22] Charlotte Hoffmann, Pavel Hubácek, Chethan Kamath, Karen Klein, and Krzysztof Pietrzak. Practical statistically-sound proofs of exponentiation in any group. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 370–399, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland.

[HHKK23] Charlotte Hoffmann, Pavel Hubácek, Chethan Kamath, and Tomás Krnák. (Verifiable) delay functions from lucas sequences. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 336–362, Taipei, Taiwan, November 29 – December 2, 2023. Springer, Cham, Switzerland.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HK05] Johan Håstad and Subhash Khot. Query efficient pcps with perfect completeness. *Theory Comput.*, 1(1):119–148, 2005.

[HKT11] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 89–98, San Jose, CA, USA, June 6–8, 2011. ACM Press.

[HLWW13] Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-resilient cryptography from minimal assumptions. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 160–176, Athens, Greece, May 26–30, 2013. Springer Berlin Heidelberg, Germany.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161, Santa Barbara, CA, USA, August 17–21, 2003. Springer Berlin Heidelberg, Germany.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short pcps. In *22nd Annual IEEE Conference on Computational Complexity (CCC 2007), 13-16 June 2007, San Diego, California, USA*, pages 278–291. IEEE Computer Society, 2007.

[IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 575–594, Amsterdam, The Netherlands, February 21–24, 2007. Springer Berlin Heidelberg, Germany.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61, Seattle, WA, USA, May 15–17, 1989. ACM Press.

[JLLW23] Aayush Jain, Huijia Lin, Ji Luo, and Daniel Wichs. The pseudorandom oracle model and ideal obfuscation. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 233–262, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, pages 60–73, Virtual Event, Italy, June 21–25, 2021. ACM Press.

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 78–95, Aarhus, Denmark, May 22–26, 2005. Springer Berlin Heidelberg, Germany.

[KC21] Dmitry Kogan and Henry Corrigan-Gibbs. Private blocklist lookups with checklist. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 875–892. USENIX Association, August 11–13, 2021.

[Kel16] Zander Kelley. Roots of sparse polynomials over a finite field. *LMS Journal of Computation and Mathematics*, 19(A):196–204, 2016.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732, Victoria, BC, Canada, May 4–6, 1992. ACM Press.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th FOCS*, pages 364–373, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press.

[KO00] Eyal Kushilevitz and Rafail Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 104–121, Bruges, Belgium, May 14–18, 2000. Springer Berlin Heidelberg, Germany.

[KS95] Erich Kaltofen and Victor Shoup. Subquadratic-time factoring of polynomials over finite fields. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 398–406, 1995.

[KU11] Kiran S Kedlaya and Christopher Umans. Fast polynomial factorization and modular composition. *SIAM Journal on Computing*, 40(6):1767–1802, 2011.

[KWJ24] Alireza Kavousi, Zhipeng Wang, and Philipp Jovanovic. SoK: Public randomness. In *2024 IEEE European Symposium on Security and Privacy*, pages 216–234, Vienna, Austria, July 8–12, 2024. IEEE Computer Society Press.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194, Singapore, December 5–9, 2010. Springer Berlin Heidelberg, Germany.

[Lip13] Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60, Bengalore, India, December 1–5, 2013. Springer Berlin Heidelberg, Germany.

[Lip24] Helger Lipmaa. Polymath: Groth16 is not the limit. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 170–206, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[LM19] Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 530–560, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

[LMQW22] Alex Lombardi, Ethan Mook, Willy Quach, and Daniel Wichs. Post-quantum insecurity from LWE. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 3–32, Chicago, IL, USA, November 7–10, 2022. Springer, Cham, Switzerland.

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In Barna Saha and Rocco A. Servedio, editors, *55th ACM STOC*, pages 595–608, Orlando, FL, USA, June 20–23, 2023. ACM Press.

[LMW25] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Black box crypto is useless for doubly efficient pir. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 65–93. Springer, 2025.

[LW15] Arjen K. Lenstra and Benjamin Wesolowski. A random zoo: sloth, unicorn, and trx. Cryptology ePrint Archive, Report 2015/366, 2015.

[LW17] Arjen K Lenstra and Benjamin Wesolowski. Trustworthy public randomness with sloth, unicorn, and trx. *International Journal of Applied Cryptography*, 3(4):330–343, 2017.

[Mau92] Ueli M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, January 1992.

[Mau93] Ueli M. Maurer. Protocols for secret key agreement by public discussion based on common information. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 461–470, Santa Barbara, CA, USA, August 16–20, 1993. Springer Berlin Heidelberg, Germany.

[May93] Timothy C. May. Time-release crypto. `https://mailing-list-archive.cryptoanarchy.wiki/archive/1993/02/a421c6fc805dfb4ae4197521e8a9e91dd456e3deab855f12af31a4b1ccccf6cb/`, 1993. (Accessed 2025-03-26).

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press.

[MMV13] Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Publicly verifiable proofs of sequential work. In Robert D. Kleinberg, editor, *ITCS 2013*, pages 373–388, Berkeley, CA, USA, January 9–12, 2013. ACM.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer Berlin Heidelberg, Germany.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39, Cambridge, MA, USA, February 19–21, 2004. Springer Berlin Heidelberg, Germany.

[MS24] Daniele Micciancio and Mark Schultz-Wu. Bit security: Optimal adversaries, equivalence results, and a toolbox for computational-statistical security analysis. In Elette Boyle and Mohammad Mahmoody, editors, *Theory of Cryptography - 22nd International Conference, TCC 2024, Milan, Italy, December 2-6, 2024, Proceedings, Part II*, volume 15365 of *Lecture Notes in Computer Science*, pages 224–254. Springer, 2024.

[MW20] Tal Moran and Daniel Wichs. Incompressible encodings. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 494–523, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109, Santa Barbara, CA, USA, August 17–21, 2003. Springer Berlin Heidelberg, Germany.

[Nec94] Vassiliy Ilyich Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, February 1994.

[OS07] Rafail Ostrovsky and William E. Skeith, III. A survey of single-database private information retrieval: Techniques and applications (invited talk). In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 393–411, Beijing, China, April 16–20, 2007. Springer Berlin Heidelberg, Germany.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer Berlin Heidelberg, Germany.

[Per09] Colin Percival. Stronger key derivation via sequential memory-hard functions, 2009.

[Pie19] Krzysztof Pietrzak. Simple verifiable delay functions. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 60:1–60:15, San Diego, CA, USA, January 10–12, 2019. LIPIcs.

[PY22] Giuseppe Persiano and Kevin Yeo. Limits of preprocessing for single-server PIR. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *33rd SODA*, pages 2522–2548, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022. ACM-SIAM.

[Raz17] Ran Raz. A time-space lower bound for a large class of learning problems. In Chris Umans, editor, *58th FOCS*, pages 732–742, Berkeley, CA, USA, October 15–17, 2017. IEEE Computer Society Press.

[RD16] Ling Ren and Srinivas Devadas. Proof of space from stacked expanders. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 262–285, Beijing, China, October 31 – November 3, 2016. Springer Berlin Heidelberg, Germany.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[RRR16] Omer Reingold, Ron Rothblum, and Guy Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th ACM Symposium on the Theory of Computing*, STOC '16, pages 49–62, 2016.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, February 1978.

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506, Tallinn, Estonia, May 15–19, 2011. Springer Berlin Heidelberg, Germany.

[RSW96] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. *Technical Report*, 1996.

[SACM21] Elaine Shi, Waqar Aqeel, Balakrishnan Chandrasekaran, and Bruce M. Maggs. Puncturable pseudorandom sets and private information retrieval with near-optimal online bandwidth and time. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 641–669, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.

[Sch80] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.

[Sha90] Adi Shamir. IP=PSPACE. In *31st FOCS*, pages 11–15, St. Louis, MO, USA, October 22–24, 1990. IEEE Computer Society Press.

[Sho93] Victor Shoup. Factoring polynomials over finite fields: asymptotic complexity vs. reality. In *Proc. IMACS Symposium, Lille, France*. Citeseer, 1993.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer Berlin Heidelberg, Germany.

[SLM+23] Shravan Srinivasan, Julian Loss, Giulio Malavolta, Kartik Nayak, Charalampos Papamanthou, and Sri Aravinda Krishnan Thyagarajan. Transparent batchable time-lock puzzles and applications to byzantine consensus. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 554–584, Atlanta, GA, USA, May 7–10, 2023. Springer, Cham, Switzerland.

[SSE+24] Ron Steinfeld, Amin Sakzad, Muhammed F. Esgin, Veronika Kuchta, Mert Yassi, and Raymond K. Zhao. LUNA: Quasi-optimally succinct designated-verifier zero-knowledge arguments from lattices. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 3167–3181, Salt Lake City, UT, USA, October 14–18, 2024. ACM Press.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

[SW21] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *SIAM J. Comput.*, 50(3):857–908, 2021.

[TCLM21] Sri Aravinda Krishnan Thyagarajan, Guilhem Castagnos, Fabien Laguillaumie, and Giulio Malavolta. Efficient CCA timed commitments in class groups. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2663–2684, Virtual Event, Republic of Korea, November 15–19, 2021. ACM Press.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 1–18, San Francisco, CA, USA, March 19–21, 2008. Springer Berlin Heidelberg, Germany.

[VZGS92] Joachim Von Zur Gathen and Victor Shoup. Computing frobenius maps and factoring polynomials. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 97–105, 1992.

[WC81] Mark N Wegman and J Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences*, 22(3):265–279, 1981.

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 379–407, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland.

[WW06] Stefan Wolf and Jürg Wullschleger. Oblivious transfer is symmetric. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 222–232, St. Petersburg, Russia, May 28 – June 1, 2006. Springer Berlin Heidelberg, Germany.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164, Chicago, Illinois, November 3–5, 1982. IEEE Computer Society Press.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

[Yeo23] Kevin Yeo. Lower bounds for (batch) PIR with private preprocessing. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part I*, volume 14004 of *LNCS*, pages 518–550, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

[Zha22] Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 66–96, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *International symposium on symbolic and algebraic manipulation*, pages 216–226. Springer, 1979.

[ZLTS23] Mingxun Zhou, Wei-Kai Lin, Yiannis Tselekounis, and Elaine Shi. Optimal single-server private information retrieval. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part I*, volume 14004 of *LNCS*, pages 395–425, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

[ZPZS24] Mingxun Zhou, Andrew Park, Wenting Zheng, and Elaine Shi. Piano: Extremely simple, single-server PIR with sublinear server computation. In *2024 IEEE Symposium on Security and Privacy*, pages 4296–4314, San Francisco, CA, USA, May 19–23, 2024. IEEE Computer Society Press.

Appendix A

Appendix: Incompressible Encryption

A.1 Programmable HPS from wPR-EGA

We show that the hash proof system based on weak pseudorandom effective group actions of [ADMP20] is programmable. Weak pseudorandom effective group actions can be instantiated from isogeny based assumptions.

A.1.1 Construction

Construction A.1.1. Using weak pseudorandom effective group actions and a randomness extractor $\mathsf{Ext} : \mathcal{X}^\lambda \times \{0,1\}^\lambda \to \{0,1\}$ we get a programmable hash proof system with an encapsulated key size of 1 bit.

$\mathsf{Gen}(1^\lambda)$:

- Sample $\overline{x}_0 \xleftarrow{\$} \mathcal{X}$ uniformly at random
- Sample $[z] \xleftarrow{\$} \mathbb{G}$ uniformly at random
- Let $\overline{x}_1 \leftarrow [z] \star \overline{x}_0$
- Return $\mathsf{pp} = (\overline{x}_0, \overline{x}_1)$ and $\mathsf{td}_{\mathcal{L}} = [z]$

$\mathsf{samp}\mathcal{L}(\mathsf{pp})$:

- Parse $\mathsf{pp} = (\overline{x}_0, \overline{x}_1)$
- Sample $[g] \xleftarrow{\$} \mathbb{G}$ uniformly at random
- Let $(x_0, x_1) \leftarrow ([g] \star \overline{x}_0, [g] \star \overline{x}_1)$
- Return $x = (x_0, x_1)$ and witness $w = ([g])$

$\mathsf{samp}Y(\mathsf{pp})$:

- Parse $\mathsf{pp} = (\overline{x}_0, \overline{x}_1)$
- Sample $[g_0] \xleftarrow{\$} \mathbb{G}$ uniformly at random
- Sample $[g_1] \xleftarrow{\$} \mathbb{G} \setminus \{[g_0]\}$ uniformly at random
- Let $(x_0, x_1) \leftarrow ([g_0] \star \overline{x}_0, [g_1] \star \overline{x}_1)$
- Return $x = (x_0, x_1)$ and trapdoor $\mathsf{td}_x = ([g_0], [g_1])$

$\mathsf{KeyGen}(\mathsf{pp})$:

- Sample $[\mathbf{h}] \xleftarrow{\$} \mathbb{G}^\lambda$ uniformly at random
- Sample $\mathbf{b} \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random
- Sample $\mathsf{s} \xleftarrow{\$} \{0,1\}^\lambda$
- Let $\mathsf{sk} \leftarrow ([\mathbf{h}], \mathbf{b})$
- Let $\mathsf{pk} \leftarrow ([h_1] \star \overline{x}_{b_1}, \ldots, [h_\lambda] \star \overline{x}_{b_\lambda}, \mathsf{s})$
- Return pk and sk

Encap $(\mathsf{pk}, x = ([x_0], [x_1]), w = [g])$:

- Parse $\mathsf{pk} = (y_1, \ldots, y_l)$
- Let $\mathsf{k} \leftarrow \mathsf{Ext}(([g] \star y_1, \ldots, [g] \star y_\lambda), \mathsf{s})$
- Return k

Decap $(\mathsf{sk}, x = ([x_0], [x_1]))$:

- Parse $\mathsf{sk} = ([\mathbf{h}], \mathbf{b})$
- Let $\mathsf{k} \leftarrow \mathsf{Ext}(([h_1] \star x_{b_1}, \ldots, [h_\lambda] \star x_{b_\lambda}), \mathsf{s})$
- Return k.

$\mathsf{Program}(\mathsf{td}_\mathcal{L}, \mathsf{td}_x, \mathsf{sk}, x, \mathsf{k})$:

- Parse $\mathsf{td}_\mathcal{L} = [z]$, $\mathsf{td}_x = ([g_0], [g_1])$, and $\mathsf{sk} = ([\mathbf{h}], \mathbf{b})$
- Repeat
  - Sample $\mathbf{b}' \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random
  - Let $([h'_1], \ldots, [h'_\lambda]) \leftarrow ((b_1 \oplus b'_1)[z] + [h_1], \ldots, (b_\lambda \oplus b'_\lambda)[z] + [h_\lambda])$
  - Let $\mathsf{sk}' \leftarrow (\mathbf{b}', [\mathbf{h}'])$
- Until $\mathsf{k} = \mathsf{Decap}(\mathsf{sk}', x)$
- Return sk'

Correctness and Language Indistinguishability Correctness and language indistinguishability remain exactly the same as in [ADMP20]. We change nothing about the hash proof system but adding the programmability. Therefore, we refrain from restating the proof.

Programmability and Programmable Smoothness Because the $\mathsf{Program}$ is just rejection sampling a secret key in exactly the same way as the original sampling but under the condition that $\mathsf{Decap}(\mathsf{sk}', x) = \mathsf{k}$ this follows from correctness and smoothness of the original scheme. Rejection sampling is efficient because the key k only has size 1.

Beyond one bit To encapsulate a key of size m one can simply generate m public-key-secret-key pairs and then encapsulation, decapsulation, and programming is done bitwise. This modification makes public and secret keys m times as big while leaving the language elements unchanged.

A.2 Programmable HPS from LWE

We present a programmable hash proof system using lattice the trapdoors by [MP12] and rounding.[1]

[1] This construction was suggested to us by Daniel Wichs.

A.2.1 Construction

Construction A.2.1. We construct a programmable hash proof system that is secure assuming LWE with superpolynomial modulus-to-noise ratio.

$\mathsf{Gen}(1^\lambda)$:

- Sample matrix $\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix} \leftarrow \mathsf{TrapSamp}(1^{2n}, 1^m, q)$ with lattice trapdoor $\mathbf{T}$ and $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$.
- Return $\mathsf{pp} = \mathbf{A}$ and $\mathsf{td}_{\mathcal{L}} = (\mathbf{B}, \mathbf{T})$.

$\mathsf{samp}\mathcal{L}(\mathsf{pp})$:

- Parse $\mathsf{pp} = \mathbf{A}$.
- Sample $\mathbf{S} \xleftarrow{\$} \{0,1\}^{v\times n}$ uniformly from the binary matrices.
- Sample $\mathbf{E} \leftarrow \chi^{v\times m}$ with small gaussian entries.
- Return $x = \mathbf{SA} + \mathbf{E}$ and $w = (\mathbf{S}, \mathbf{E})$.

$\mathsf{samp}Y(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$:

- Parse $\mathsf{td}_{\mathcal{L}} = (\mathbf{B}, \mathbf{T})$.
- Return $x = \mathbf{B}$ and $\mathsf{td}_x = ()$.

$\mathsf{KeyGen}(\mathsf{pp})$:

- Parse $\mathsf{pp} = \mathbf{A}$.
- Sample $\mathbf{R} \xleftarrow{\$} \chi_\sigma^{m\times u}$ with small gaussian entries.
- Return $\mathsf{pk} = \mathbf{AR}$ and $\mathsf{sk} = \mathbf{R}$

$\mathsf{Encap}(\mathsf{pk}, x, w)$:

- Parse $\mathsf{pk} = \mathbf{AR}$ and $w = (\mathbf{S}, \mathbf{E})$.
- Let $\mathsf{k} \leftarrow \lceil \mathbf{SAR} \rfloor$.
- Return k.

$\mathsf{Decap}(\mathsf{sk}, x)$:

- Parse $\mathsf{sk} = \mathbf{R}$ and $x = \mathbf{SA} + \mathbf{E}$.
- Let $\mathsf{k} \leftarrow \lceil (\mathbf{SA} + \mathbf{E})\mathbf{R} \rfloor$.
- Return k.

$\mathsf{Program}(\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x, \mathsf{sk}, x, \mathsf{k})$:

- Parse $\mathsf{sk} = \mathbf{R}$.
- Sample $\mathbf{K} \in \mathbb{Z}_q^{v\times u}$ uniformly at random conditioned on $\lceil \mathbf{K} \rfloor = \mathsf{k}$.
- Sample short $\mathbf{R}^* \leftarrow \mathsf{SampleD}(\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}, \mathbf{T}, \begin{pmatrix}\mathbf{AR}\\ \mathbf{K}\end{pmatrix}, \sigma)$
- Return $\mathsf{sk}' = \mathbf{R}^*$.

Statistical Correctness With $\mathbf{A}$, $\mathbf{S}$, $\mathbf{E}$ and $\mathbf{R}$ as in the construction, we have that $\mathsf{Encap}(\mathsf{pk}, x, w) = \lceil \mathbf{SAR} \rfloor$ which is statistically close to $\lceil (\mathbf{SA} + \mathbf{E})\mathbf{R} \rfloor = \lceil \mathbf{SAR} + \mathbf{ER} \rfloor = \mathsf{Decap}(\mathsf{sk}, x)$ when using a superpolynomial modulus-to-noise ratio.

Language Indistinguishability By LWE assumption $\mathbf{SA} + \mathbf{E}$ is indistinguishable from uniformly random. The trapdoored matrix $\mathbf{B}$ is statistically close to uniform. Therefore, language indistinguishability follows from LWE.

Programmability Using the correctness of the lattice trapdoor we have that the algorithm $\mathsf{Program}(\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x, \mathsf{sk}, x, \mathsf{k})$ returns a key sk' s.t. $\lceil \mathbf{BR}^* \rfloor = \mathsf{k}$.

Programmable Smoothness $\mathsf{sk} = \mathbf{R}$ are sampled from $\chi_\sigma^{m \times u}$ with small gaussian entries and $\mathbf{R}^*$ is statistically close to being sampled from $\chi_\sigma^{m \times u}$ conditioned on $\lceil \mathbf{BR}^* \rfloor = \mathsf{k}$ where k is uniformly random. Therefore, sk' is statistically close to sk.

A.2.2 Incompressible Encryption

Since we do not have a 2-smooth hash proof system for the same language as the above LWE programmable HPS, we do not achieve CCA incompressible PKE but only CPA incompressible PKE. The transfromation stays almost the same as in the CCA case.

Below we give the transfromation for CPA incompressible PKE.

Construction A.2.2 (Incompressible PKE)**.** Given security parameter λ, space bound S, and message length n let (KeyGen', Encap', Decap', $\mathsf{Program}'$) be a Y-programmable hash proof system for a language $\mathcal{L} \subset X$ (where you can sample x with according witness from $\mathcal{L}$ and sample x with according trapdoor from Y) where the representation size of X is $p(\lambda, S, n)$ and encapsulated keys of size $k(\lambda, S_{\mathsf{sym}}, n)$, and ($\mathsf{Enc}_{\mathsf{sym}}$, $\mathsf{Dec}_{\mathsf{sym}}$) be an incompressible SKE with messages of size n, keys of size $k(\lambda, S_{\mathsf{sym}}, n)$ and ciphertexts of size $l(\lambda, S_{\mathsf{sym}}, n)$ with incompressible SKE adversary being allowed to leak a state of size $S_{\mathsf{sym}} = S + p(\lambda, S, n)$.

$\mathsf{KeyGen}(1^\lambda, 1^S)$**:**

- Generate language and corresponding trapdoor

$$(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^n).$$

- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.
- Return $\mathsf{pk} = \mathsf{pk}'$ and $\mathsf{sk} = \mathsf{sk}'$.

$\mathsf{Enc}(\mathsf{pk}, \mathsf{m})$**:**

- Parse $\mathsf{pk} = \mathsf{pk}'$
- Let $(x, w) \leftarrow \mathsf{samp}\mathcal{L}(\mathsf{pp})$.
- Let $\mathsf{k} \leftarrow \mathsf{Encap}'(\mathsf{pk}', x, w)$.
- Let $c_{\mathsf{sym}} \leftarrow \mathsf{Enc}_{\mathsf{sym}}(\mathsf{k}, \mathsf{m})$.
- Return $c = (x, c_{\mathsf{sym}})$.

$\mathsf{Dec}(\mathsf{sk}, c)$**:**

- Parse $\mathsf{sk} = \mathsf{sk}'$.
- Parse $c = (x, c_{\mathsf{sym}})$.
- Let $\mathsf{k} \leftarrow \mathsf{Decap}'(\mathsf{sk}', x)$
- Return $\mathsf{m} = \mathsf{Dec}_{\mathsf{sym}}(\mathsf{k}, c_{\mathsf{sym}})$.

Correctness Follows from the correctness of the hash proof system (KeyGen', Encap', Decap', $\mathsf{Program}'$) and the symmetric key encryption scheme ($\mathsf{Enc}_{\mathsf{sym}}$, $\mathsf{Dec}_{\mathsf{sym}}$).

The security proof is similar but simpler than the CCA case.

Theorem A.2.3 (Security)**.** *The PKE construction 5.5.2 has incompressible PKE security if (*KeyGen'*,* Encap'*,* Decap'*,* $\mathsf{Program}'$*) is a programmable hash proof system with the listed parameters and (*$\mathsf{Enc}_{\mathsf{sym}}$*,* $\mathsf{Dec}_{\mathsf{sym}}$*) is an incompressible secure SKE with the listed parameters.*

Proof. We prove security via hybrids. First we list the hybrids and then argue their indistinguishability. In each hybrid we highlight the changes compared to the previous one.

$H_0(\lambda, S)$:

- Run key generation algorithm $\mathsf{KeyGen}(1^\lambda, 1^S)$ to obtain $(\mathsf{pk}, \mathsf{sk})$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{pk})$ on public key pk to receive two messages m_0, m_1 and state st_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Run $c \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_b)$ to encrypt m_b.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk}, c, \mathsf{st}_1)$ to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

In H_1 we explicitly represent what happens in KeyGen and Enc.

$H_1(\lambda, S)$:

- Generate language and corresponding trapdoor $(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^n)$.
- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.
- Let $\mathsf{pk} = \mathsf{pk}'$ and $\mathsf{sk} = \mathsf{sk}'$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1$ on public key pk to receive two messages m_0, m_1 and state st_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Let $(x, w) \leftarrow \mathsf{samp}\mathcal{L}(\mathsf{pp})$.
- Let $\mathsf{k} \leftarrow \mathsf{Encap}'(\mathsf{pk}', x, w)$.
- Let $c_{\mathsf{sym}} \leftarrow \mathsf{Enc}_{\mathsf{sym}}(\mathsf{k}, \mathsf{m}_b)$.
- Let $c = (x, c_{\mathsf{sym}})$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{pk}, c, \mathsf{st}_1)$ to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

In H_2 we use the decapsulation mechanisms to encrypt the challenge message instead of encapsulation.

$H_2(\lambda, S)$:

- Generate language and corresponding trapdoor
$$(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^n).$$
- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.

- Let $\mathsf{pk} = \mathsf{pk}'$ and $\mathsf{sk} = \mathsf{sk}'$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{pk})$ on public key pk to receive two messages m_0, m_1 and state st_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Let $(x, w) \leftarrow \mathsf{samp}\mathcal{L}(\mathsf{pp})$.
- Let $\mathsf{k} \leftarrow \mathsf{Decap}'(\mathsf{sk}', x)$.
- Let $c_{\mathsf{sym}} \leftarrow \mathsf{Enc}_{\mathsf{sym}}(\mathsf{k}, \mathsf{m}_b)$.
- Let $c = (x, c_{\mathsf{sym}})$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2^{\mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk}, c, \mathsf{st}_1)$ to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

In H_3 we sample x from $Y \subset X \setminus \mathcal{L}$ instead of $\mathcal{L}$.

$H_3(\lambda, S)$:

- Generate language and corresponding trapdoor

$$(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^n).$$

- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.
- Let $\mathsf{pk} = \mathsf{pk}'$ and $\mathsf{sk} = \mathsf{sk}'$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{pk})$ on public key pk to receive two messages m_0, m_1 and state st_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Let $(x, \mathsf{td}_x) \leftarrow \mathsf{samp}Y(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$.
- Let $\mathsf{k} \leftarrow \mathsf{Decap}'(\mathsf{sk}', x)$.
- Let $c_{\mathsf{sym}} \leftarrow \mathsf{Enc}_{\mathsf{sym}}(\mathsf{k}, \mathsf{m}_b)$.
- Let $c = (x, c_{\mathsf{sym}})$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{pk}, c, \mathsf{st}_1)$ to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

In H_4 we program the secret key given to the adversary to decapsulate the ciphertext to the randomly chosen key k.

$H_4(\lambda, S)$:

- Generate language and corresponding trapdoor

$$(\mathsf{pp}, \mathsf{td}_{\mathcal{L}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^n).$$

- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.
- Let $\mathsf{pk} = \mathsf{pk}'$ and $\mathsf{sk} = \mathsf{sk}'$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1$ on public key pk to receive two messages m_0, m_1 and state st_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Let $(x, \mathsf{td}_x) \leftarrow \mathsf{samp}Y(\mathsf{pp}, \mathsf{td}_{\mathcal{L}})$.
- Sample $\mathsf{k} \xleftarrow{\$} \{0,1\}^{k(\lambda, S_{\mathsf{sym}}, n)}$.
- Let $c_{\mathsf{sym}} \leftarrow \mathsf{Enc}_{\mathsf{sym}}(\mathsf{k}, \mathsf{m})$.
- Let $c = (x, c_{\mathsf{sym}})$.

- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{pk}, c)$ to produce a state st_2 smaller than S.
- Let $\mathsf{sk}'_{prog} \leftarrow \mathsf{Program}(\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x, \mathsf{sk}', x, \mathsf{k})$.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk} = \mathsf{sk}'_{prog}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

$H_0 \approx H_1$:
The differences between H_0 and H_1 are purely syntactical. In H_1 we just show more detail of KeyGen and Enc.

$H_1 \approx H_2$:
In H_2 we merely change how the challenge ciphertext is calculated. By the correctness of the hash proof system these two hybrids look identical to the adversary.

$H_2 \approx_c H_3$:
In H_3 sample x from $Y \subset X \setminus \mathcal{L}$ instead of $\mathcal{L}$. These two hybrids are computationally indistinguishable by the language indistinguishability. Assume there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ that can distinguish the two hybrids H_2 and H_3 with a non-negligible advantage of ϵ. From this we construct a statistical adversary $\mathcal{A}'$ that can break language indistinguishability of the HPS with advantage ϵ.

$\mathcal{A}'(\mathsf{pp}, x)$:

- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.
- Let $\mathsf{pk} = \mathsf{pk}'$ and $\mathsf{sk} = \mathsf{sk}'$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{pk})$ on public key pk to receive two messages m_0, m_1 and state st_1.
- Sample bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
- Let $\mathsf{k} \leftarrow \mathsf{Decap}'(\mathsf{sk}', x)$.
- Let $c_{\mathsf{sym}} \leftarrow \mathsf{Enc}_{\mathsf{sym}}(\mathsf{k}, \mathsf{m}_b)$.
- Let $c = (x, c_{\mathsf{sym}})$.
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{pk}, c, \mathsf{st}_1)$ to produce a state st_2 smaller than S.
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{sk}, \mathsf{st}_1, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$.
- The adversary wins if $b = b'$.

If $\mathcal{A}$ can distinguish H_2 from H_3 with advantage ε then the advantage of $\mathcal{A}'$ of distinguishing a x sampled from $\mathcal{L}$ and sampling x from $Y \subset X \setminus \mathcal{L}$ is also ϵ as it perfectly simulates H_2 in the case that $x \in \mathcal{L}$ and perfectly simulates H_3 in the other case.

$H_3 \approx_s H_4$:
According to programmable smoothness of $(\mathsf{KeyGen}', \mathsf{Encap}', \mathsf{Decap}')$ if $x \notin \mathcal{L}$ then $(\mathsf{pk}', \mathsf{sk}', x)$ is statistically close to the previously used distribution $(\mathsf{pk}', \mathsf{Program}(\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x, \mathsf{sk}', x, \mathsf{k}), x)$ for uniformly random k. Because this is exactly what we switch we get that H_3 and H_4 are statistically close.

$H_4 \approx \mathsf{Dist}^{\mathsf{IncomSKE}}_{\mathcal{A}', \Pi_{\mathsf{sym}}}$:
Finally, given an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ that wins experiment in hybrid $H_4(\lambda, S)$ with probability ϵ we construct a multi-stage adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2, \mathcal{A}'_3)$ that wins the $\mathsf{Dist}^{\mathsf{IncomSKE}}_{\mathcal{A}', \Pi_{\mathsf{sym}}}(\lambda, S + p(\lambda))$ experiment with probability ϵ.

$\mathcal{A}'_1(1^\lambda, 1^S)$:

- Generate language and corresponding trapdoor

$$(\mathsf{pp}, \mathsf{td}_L) \leftarrow \mathsf{Gen}(1^\lambda, 1^n).$$

- Let $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}'(\mathsf{pp})$.
- Let $\mathsf{pk} = \mathsf{pk}'$ and $\mathsf{sk} = \mathsf{sk}'$.
- Run the adversary $\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{pk})$ on public key pk to receive two messages m_0, m_1 and state st_1.
- Let $\mathsf{st}_1' \leftarrow (\mathsf{pk}, \mathsf{sk}, \mathsf{td}_{\mathcal{L}}, \mathsf{st}_1)$.
- Return m_0, m_1, and st_1'

$\mathcal{A}_2'(\mathsf{st}_1', c_{\mathsf{sym}})$:

- Parse $\mathsf{st}_1' = (\mathsf{pk}, \mathsf{sk}, \mathsf{td}_{\mathcal{L}}, \mathsf{st}_1)$
- Let $(x, \mathsf{td}_x) \leftarrow \mathsf{samp}Y(\mathsf{pp}, \mathsf{td}_L)$
- Let $c \leftarrow (x, c_{\mathsf{sym}})$
- Run the adversary $\mathsf{st}_2 \leftarrow \mathcal{A}_2(\mathsf{pk}, c, \mathsf{st}_1)$ to produce a state st_2 smaller than S
- Return state $\mathsf{st}_2' \leftarrow (x, \mathsf{td}_x, \mathsf{st}_2)$ smaller than $S_{\mathsf{sym}} = S + p(\lambda)$

$\mathcal{A}_3'(\mathsf{k}, \mathsf{st}_1', \mathsf{st}_2', \mathsf{m}_0, \mathsf{m}_1)$:

- Parse $\mathsf{st}_1 = (\mathsf{pk}, \mathsf{sk} = \mathsf{sk}', \mathsf{td}_{\mathcal{L}}, \mathsf{st}_1')$
- Parse $\mathsf{st}_2 = (x, \mathsf{td}_x, \mathsf{st}_2')$
- Program $\mathsf{sk}'_{prog} \leftarrow \mathsf{Program}(\mathsf{td}_{\mathcal{L}}, \mathsf{td}_x, \mathsf{sk}', x, \mathsf{k})$
- Run the final adversary $b' \leftarrow \mathcal{A}_3(\mathsf{pk}, \mathsf{sk}'_{prog}, \mathsf{st}_2, \mathsf{m}_0, \mathsf{m}_1)$
- Return b'

$\mathcal{A}'$ wins $\mathsf{Dist}^{\mathsf{IncomSKE}}_{\mathcal{A}', \Pi_{\mathsf{sym}}}(\lambda, S + p(\lambda))$ iff $\mathcal{A}$ wins in $H_4(\lambda, S)$ because $\mathcal{A}'$ perfectly simulates H_4 from the perspective of $\mathcal{A}$.

□

Appendix B

Appendix: Designated-Verifier SNARGs

B.1 On Measuring Concrete Security

In this section we discuss in more detail our measures of *concrete* (as opposed to asymptotic) proof length, which follow the standards of previous related works, and discuss an important related distinction between public and designated verification.

Concrete proof length is only meaningful when specifying a concrete security level. While there are several principled approaches for defining the exact "bit security" of cryptographic primitives (see, e.g., [MS24] and references therein), the common practice in generic model constructions is to simply refer to the bit-length of the oracle instantiation. For instance, a garbled circuit construction in the ROM is referred to as having 128-bit security when the random oracle is instantiated using (say) AES, even if the security reduction involves some *multiplicative* polynomial loss (as opposed to quadratic loss required by collision resistance). Similarly, in GGM constructions that rely on the hardness of computing discrete logarithms, a 128-bit security level refers to an instantiation with suitable elliptic-curve groups whose order is a 256-bit prime. This accounts for the quadratic speedup of fast algorithms for computing discrete logarithm. We follow this convention here as well.

Finally, we would like to point out an important distinction between dv-SNARGs and publicly verifiable SNARGs in the context of measuring concrete security. In publicly verifiable proofs, the prover can test whether a proof they generate is accepted by the verifier. It is therefore natural to only measure the expected time it takes for a malicious prover to generate an accepted false proof. In contrast, in a designated verifier setting, where the prover does not know whether they will be caught cheating, there is a natural separation between the computational security level and the statistical soundness error.

For example, soundness as low as 2^{-64} is more than enough in most use-cases of dv-SNARGs, when a malicious prover cannot verify their own proofs without access to a verification oracle. Indeed, an access to a verification oracle is typically much more costly than just checking a publicly verifiable proof. Consider an extreme scenario in which it costs \$0.0001 to query a verification oracle, and if a prover manages to cheat, they gain the entire earth's GDP ($\approx \$10^{14}$ in 2022). With soundness error 2^{-64}, even in this extreme scenario a malicious prover who tries to cheat has a negative expected utility.

Given the above, our concrete measures of proof size refer to the arguably conservative

setting of 2^{-80} soundness error at a 128-bit security level, where the latter refers to using a 256-bit group and ignores the small loss in the security reduction.