Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research
Report
RR-02-02

Secure Mobile Multiagent Systems In Virtual Marketplaces

A Case Study on Comparison Shopping

Ina Schaefer

March 2002

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210
E-Mail: info@dfki.uni-kl.de

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341
E-Mail: info@dfki.de

WWW: http://www.dfki.de

Deutsches Forschungszentrum für Künstliche Intelligenz

DFKI GmbH

German Research Center for Artificial Intelligence

Founded in 1988, DFKI today is one of the largest nonprofit contract research institutes in the field of innovative software technology based on Artificial Intelligence (AI) methods. DFKI is focusing on the complete cycle of innovation — from world-class basic research and technology development through leading-edge demonstrators and prototypes to product functions and commercialization.

Based in Kaiserslautern and Saarbrücken, the German Research Center for Artificial Intelligence ranks among the important "Centers of Excellence" worldwide.

An important element of DFKI's mission is to move innovations as quickly as possible from the lab into the marketplace. Only by maintaining research projects at the forefront of science can DFKI have the strength to meet its technology transfer goals.

DFKI has about 165 full-time employees, including 141 research scientists with advanced degrees. There are also around 95 part-time research assistants.

Revenues for DFKI were about 30 million DM in 2000, half from government contract work and half from commercial clients. The annual increase in contracts from commercial clients was greater than 20% during the last three years.

At DFKI, all work is organized in the form of clearly focused research or development projects with planned deliverables, various milestones, and a duration from several months up to three years.

DFKI benefits from interaction with the faculty of the Universities of Saarbrücken and Kaiserslautern and in turn provides opportunities for research and Ph.D. thesis supervision to students from these universities, which have an outstanding reputation in Computer Science.

The key directors of DFKI are Prof. Wolfgang Wahlster (CEO) and Dr. Walter Olthoff (CFO).

DFKI's five research departments are directed by internationally recognized research scientists:

❏ Knowledge Management (Director: Prof. A. Dengel)
❏ Intelligent Visualization and Simulation Systems (Director: Prof. H. Hagen)
❏ Deduction and Multiagent Systems (Director: Prof. J. Siekmann)
❏ Language Technology (Director: Prof. H. Uszkoreit)
❏ Intelligent User Interfaces (Director: Prof. W. Wahlster)

In this series, DFKI publishes research reports, technical memos, documents (eg. workshop proceedings), and final project reports. The aim is to make new results, ideas, and software available as quickly as possible.

Prof. Wolfgang Wahlster
Director

Secure Mobile Multiagent Systems In Virtual Marketplaces

A Case Study on Comparison Shopping

Ina Schaefer

DFKI-RR-02-02

This work has been supported by a grant from The Federal Ministry of Education, Science, Research, and Technology (FKZ ITW-01 IWA 01).

ISSN 0946-008X

Secure Mobile Multiagent Systems
In Virtual Marketplaces

A Case Study on Comparison Shopping

Ina Schaefer

Abstract

The growth of the Internet has deeply influenced our daily lives as well as our commercial structures. Agents and multiagent systems will play a major role in the further development of Internet-based applications like virtual marketplaces. However, there is an increasing awareness of the security problems involved. These systems will not be successful until their problems are solved. This report examines comparison shopping, a virtual marketplace scenario and an application domain for a mobile multiagent system, with respect to its security issues. The interests of the participants in the scenario, merchants and clients, are investigated. Potential security threats are identified and security objectives counteracting those threats are established. These objectives are refined into building blocks a secure multiagent system should provide. The building blocks are transformed into features of agents and executing platforms. Originating from this analysis, solutions for the actual implementation of these building blocks are suggested. It is pointed out under which assumptions it is possible to achieve the security goals, if at all.

Contents

1 Introduction

The success of the Internet and the World Wide Web has deeply influenced our every day lives as well as our commercial structures. Agent technologies and multiagent systems will play a major part in the further development of WWW-based applications: virtual marketplaces with customer and seller agents, chat rooms and avatars, personal assistant agents as well as non benevolent agents designed to attack a site, are just some of many applications. While there is still a considerable hype concerning agent technologies, there is also an increasing awareness of the problems involved. The growth of Internet-based commerce is tempered by legitimate concerns on the security of such systems. In particular, these applications will not be successful unless security issues can be adequately handled. One of the major concerns for both customers and merchants participating in eCommerce is the potential loss of assets and privacy due to the breaches in the security of corporate computer systems. Although there is a large body of work on cryptographic techniques that provide basic building blocks to solve specific security problems, relatively little work has been done in investigating security in a multiagent system context. The introduction of mobile software agents significantly increases the risks involved in Internet and Web-based applications.

Mobile agents have several advantages in a system like the Internet. Mobile agents travel to a platform to be executed and go where the required data is stored. So the overall communication traffic over low-bandwidth, high-latency and high-cost access networks is reduced. Also if the connection to the agent owner is interrupted, the agent can still go on working. It returns the results when the connection is re-established. The owner does not have to be online all the time for his agent to perform his task. This is particularly useful in case the connection is made via mobile phone. Therefore, the trade-off between performance and security issues has to be considered.

The research presented in this report was done as a part of the SEMAS (Security in Mobile Mulitagent Systems) project funded by the German Ministry for Education and Research. It investigates the fundamental security threats in the design of mobile multiagent systems within virtual marketplaces. These threats can be classified according to whether they are inherent to the application scenario to be implemented, inherent to the multiagent system level design, a consequence of the design of the individual agent or a result of using mobile computing. SEMAS therefore investigates into how the design of the application, the design of the agent society and the selection of the computational paradigm influences the characteristics of the security threats and how security measures can be combined to an all-embracing security infrastructure. Accordingly, the SEMAS methodology and also the research work is organised into three layers: firstly the application layer, secondly the system architecture and thirdly the computational architecture. The aim of the SEMAS project is to come up with a methodology for the design and implementation of secure mobile multiagent systems, particularly for virtual marketplaces. Since SEMAS covers the application oriented design phase as well, there is a need to focus on a family of scenarios. Guided by the economical importance and scientific significance, SEMAS explores concrete instances of virtual marketplaces based on auctions and free negotiation. The cases considered in SEMAS are auctions and comparison shopping as important applications for mobile agents in virtual marketplaces. They are also important instances of negotiation on markets from an economic perspective.

This report focuses on the comparison shopping scenario, one of the SEMAS cases on the application layer. It investigates security requirements and possible solutions for this concrete scenario. In [DEW96], the comparison shopping problem is described as follows: given are a domain description with useful attributes to differentiate between different products, a set of URLs for the homepages of possible vendors, an attribute A by which the user wants to compare the vendors (e.g. the price) and finally a specification of the desired product in terms of desired values for the product's attributes. The task of a comparison shopping agent is to determine the set of stores where the desired product is available sorted by the attribute A.

In this report, a detailed model for the comparison shopping scenario will be established. With respect to its different phases and instances, it will be explored which interests and expectations the participants have. The interests and possibilities of an attacker and the resulting security threats for the application will be considered. From that analysis, the overall security objectives counteracting those threats are identified. The security objectives specify the requirements the system has to satisfy for considering it as secure. Having sketched a potential mobile multiagent system to realise the scenario, the objectives are broken down into more detailed features of the system to be constructed, i.e. building blocks or interfaces the system architecture has to provide at the application level. The building blocks are further refined into features of single agents roaming in the system and of executing platforms. Finally, concrete technical means are proposed to implement the building blocks on the level of the system architecture.

Furthermore, this report gives an overview of research on comparison shopping from different points of view, i.e. the construction of shopbots, virtual marketplaces, economic impact and security issues. Additionally, an overview of existing security mechanisms for mobile agents and platforms is presented. It is shown which of those are applicable in this case study.

The remainder of this report is structured as follows: In section 2, we discuss related work with respect to comparison shopping and security of mobile agents. In section 3, a detailed model of the comparison shopping scenario is established and its different phases and instances are analysed. In section 4, we move towards a secure system and show which building blocks are needed to construct a secure mobile multiagent system for this applicaion scenario and how they can be realised technically. Section 5 finishes the report with a brief summary of the main results and an outlook to future work.

2 Related work

2.1 Related work – Comparison Shopping

Research on comparison shopping can be divided into different areas according to its focus. The first main area of research is concerned with the functionality and construction of comparison shopping agents or so-called shopbots. It is investigated how a comparison shopping agent has to work, how wrappers for the retrieved information are constructed and how the findings will be ranked. A second focus are virtual marketplaces, most of which contain a comparison shopping phase. A third area of research is the economic perspective on comparison shopping. Researchers investigate which impact shopbots have to the economy and develop methods to analyse economies with comparison shopping agents. Finally, comparison shopping is often used as example in literature considering security of mobile agents. Many authors use comparison shopping to illustrate the security issues linked to mobile agents. In the following, we have a closer look at these four areas of comparison shopping research.

2.1.1 Construction and Working Principles of Comparison Shopping Agents

The first area of comparison shopping research is concerned with the construction of comparison shopping agents that are sent out to find the best match for a given product description.

Andresen Consulting's BargainFinder [Kru96] is the first ever model of a merchant brokering shopping agent or comparison shopping agent. Given a specific music CD name BargainFinder requests its price (including delivery) from each of nine different online music catalogs using the same requests as a web browser. It presents its results to the consumer that makes the final decision where to buy from. Several merchants decided not to participate or blocked BargainFinder. BargainFinder works in a hard-wired way and is hand-coded for the specific product domain. It employs manual rule extraction and does not construct wrappers itself. This means that it is explicitly encoded in the BargainFinder agent how the information from a specific visited website is extracted. Exite's Jango was another merchant brokering shopping assistant similar to BargainFinder, but with more product features and shopping categories to search across.

Shopbot [DEW96] is comparable to BargainFinder and Jango. It is inspired by BargainFinder's feasibility demonstration and popularity. However, Shopbot is product independent and takes a description of a product domain as an input. All information it needs about a shop is its URL. Shopbot learns how to extract information from the store and relies on AI techniques like heuristic search, pattern matching, or inductive learning in contrast to the hand-coded BargainFinder. Shopbot suggests an automatic rule extraction technique by analysing and learning in shopping malls. In order to integrate specific product information, Shopbot removes irrelevant information such as advertisments by using inductive learning mechanisms and then extracts necessary product information. However, Shopbot uses strong assumptions about the structure of HTML files and the display format of products for learning. More about the technical details can be found in [PDEW95].

[JCK$^+$00] proposes a more scalable comparison shopping agent as an improvement to Shopbot. They present a robust and automatic shopping mall learning algorithm and an ontology generation method. The main idea of the proposed algorithm is to determine the position of a product description unit from the HTML source of a search result page by recognizing a repeated pattern of logical line information. The positional information is converted into an extraction rule that becomes the main part of the wrapper. This algorithm is simple, but robust because no strong biases are assumed. Consequently, the success rate is higher for constructing a correct wrapper. Furthermore, a mechanism is suggested that generates the ontology from the well-structured outputs. The existing ontology is automatically extended by applying it to unstructured search results. More details on the construction of these wrappers can be found in [YLC00].

In [BG99], Brody et al. introduce the PocketBargainFinder device. A customer enters a bookshop and finds an interesting book. He takes the PocketBargainFinder and scans the book's barcode. PocketBargainFinder connects to the Internet and evaluates the book's price at different online retailers. The customer sees whether he could order the book on the Internet for better conditions taking delivery costs and delivery time into account. The used hardware is a PDA and a barcode reader as well as wireless communication. PocketBargainFinder is proposed for use in augmented commerce, i.e. commerce in the real world enhanced with electronic commerce components.

[GM98] stresses the necessity of including multiple attributes in the product ranking done by agents during comparison shopping. An online-merchant would, as in the physical world, prefer his customers only to shop at his site because cross-merchant comparison is seen as a threat to his own profitability. However, consumers want to compare product offerings across merchants. Cross-merchant comparison is a characteristic of retail marketplaces. Thus, merchants enhance their products with product-added values like extended warranties, superior customer service and so on to distinguish themselves from other merchants. Cross-merchant comparison is much easier and less costly if it is done by comparison shopping agents. The first generation of comparison shopping agents makes their recommendations only on the price of the product ignoring other product-added values. That results in inappropriately competative markets. That may mislead customers since the cheapest product is not always the best to buy. Comparison shopping agents have to be improved in so far as they should employ integrative negotiation techniques, i.e. they try to resolve a conflict over multiple, but not mutually exclusive goals [GM98]. This decision process involving multiple attributes can be described and analysed using multi-attribute decision theory.

2.1.2 Virtual Marketplace Systems

Many of the existing virtual marketplace systems implement a stage similar to comparison shopping. Kashbah [CM96] is a web-based multi-agent classified ad system where users create buying and selling agents helping to transact goods. These agents automate comparison shopping and negotiation between buyers and sellers. A user wanting to buy or sell a good creates an agent and sends it to a centralised marketplace. An agent's goal is to complete an acceptable deal satisfying its owner's preferences. However, there are other more sophisticated markets which implement more market mechanisms and more advanced negotiation.

MAGMA [TMGW97] is such a more sophisticated virtual marketplace system which comprises all stages from the product brokering to the actual purchase. MAGMA, as a real virtual marketplace, comprises banking, communication infrastructure, mechanisms for transportation and storage of goods, facilities for advertising, economic mechanisms and transaction protocols. MAGMA also contains a comparison shopping stage. Another virtual marketplace system of this kind including comparison shopping called Tete-a-Tete was developed at the MIT.

In [GMM98] a survey of existing virtual and agent-based marketplace systems is given. The classification of such virtual marketplaces is made according to which stages of the Consumer Buying Behaviour (CBB) model are implemented. The CBB model divides a purchase process into different phases. In the product brokering stage, a customer decides what he wants to buy.

In the following merchant brokering or comparison shopping stage the customer evaluates the offers for this product of different merchants to find out whom to buy from. This includes the evaluation of merchant alternatives, based on customer provided criteria (e.g. price, warranty, availability, delivery time, repudiation). After the merchant brokering stage, the negotiation phase follows. The process ends with purchase and delivery of a product. In this survey, it can be seen which existing systems implement a comparison shopping stage and which do not.

2.1.3 The Economic Perspective

Kephart and Greenwald in [KG99, GK99] explore the potential impact of shopbots on market dynamics by proposing, analysing and simulating a model of shopbot economics which incorporates software agent representations of buyers and sellers. They state that the reduction of economic friction due to the decreased search costs could dramatically alter market behaviour in the future as shopbots become more frequently used. Their main objective is to understand the dynamics of the future information economy in which software agents, rather than humans, play the key role and to design utility maximisation algorithms for economically motivated software-agents. In the latter paper, they also examine the impact of pricebots, i.e. software agents that set prices according to supply and demand.

In [MU01], the authors focus on the impact of software agent-based shopbots and pricebots on electronic markets. Shopbots and pricebots change the capabilities available to buyers and sellers on the market. A shopbot is attached to a single buyer and able to query several sellers about a desired product. In this sense, shopbots are similar to comparison shopping agents. A pricebot is attached to a single seller and has the ability to change the price of a service dynamically to maximize the seller's profit. The paper proposes a model in which different situations, e.g. no price and no shopbot, only shopbots or both of them are analysed. One main result of this investigation is that sellers are always better off colluding with shopbots by fixing prices and permitting them to evaluate those. A second result is that the use of pricebots may result in a price-war which in the long run leads to profit decline.

2.1.4 Comparson Shopping in the Security Literature

Also in the security-related literature comparison shopping is widely spread as a motivating example. [Yee97] proposes means to protect the computation results of free-roaming mobile agents. This is motivated by the following example of comparison shopping. A software agent is sent out to find the least expensive fare for a flight from San Diego to Washington D.C. taking into account various trip timing, seat preference and routing constraints. One of the queried airlines, Fly-By-Night.com, runs a web server www.flybynight.com, where the agent's code is automatically recognized and brainwashed. The agent's memory about collected offers of other airlines is modified such that it ends up recommending a flight by Fly-By-Night airlines although a less expensive daytime flight has been offered by another airline. This example is also quoted by other authors, e.g. [FGS96b], [Mea97], [KAG98].

In [CMS01], a framework for a secure marketplace on the Internet is proposed. A comparison shopping agent, dispatched to find the most convenient offer for a flight ticket among several air travel agencies, is facing the following security risks: the shopping agent could try to access privileged information, reduce resource availability of the current hosting site or perform a coordinate attack with other agents. The other way round, a malicious host could disclose agent's private information, tamper with the agent's code or modify or delete previously collected prices, thereby gaining economic advantage.

[Hoh97] uses a comparison shopping example as illustration of the code mess up mechanism proposed to protect agents from direct manipulation of their code. The code of the comparison shopping agent is altered such that the semantic of the agent cannot be found out easily.

In [Vig98], Vigna proposes the concept of cryptographic traces where execution traces of the mobile agents are used to check whether agents have been executed correctly. At the end of his paper he illustrates his concept at a comparison shopping scenario. He shows that using his approach it is possible to find out that previously collected offers where modified.

More details about the proposed mechanisms can be found in the next section.

2.2 Related work – Security Mechanisms for Mobile Agents

Research on the security of mobile agents is divided into two different categories, firstly the protection of hosts from malicious agents, the easier part, and secondly the protection of agents from malicious hosts which is much harder. Some approaches, however, have components which can be used for protection in both directions. In the following, we will illustrate some techniques which we may use later in our system.

2.2.1 Protection of Hosts from Malicious Agents

In this section, we focus on the protection of hosts from malicious operations performed by agents. We order the techniques according to increasing strictness. The final approach in this part concentrates on resource control at hosts.

Signed Code The main idea of signing the code digitally is to create an unforgeable link between the author and his code. The author or the dispatcher of a mobile object signs it with his secret key and certifies that this is his object. The signature can be verified with the signer's public key assuming a PKI exists. If there exists a trust model the trust in the author can so be transferred to the mobile object that works on his behalf. A platform that trusts the author of the code assumes that the code is not malicious and executes it. This approach is portable to almost any system, where a public key infrastructure exists. This however restricts the openness of the system since participants have to register their keys with a central authority. A drawback could be that an author can also sign malicious code and harm someone that trusts him.

Safe Interpreters [**Moo98**] Running already compiled executables is a severe security risk. It can be addressed by shifting to the interpretation of some intermediary code on a virtual machine. The security problem is reduced to the security policy implemented by the interpreter. Examples for this approach are Safe-Tcl and Java1.

1. **Safe-Tcl**

   In Safe-Tcl, the agent is executed inside a padded cell, which operates in a different name space. The control over the environment belongs to a master interpreter which prevents the call of unsafe functions. The problem is that it has to be determined whether a function is unsafe or not. So functions that are essential for the agent may not be executed. In addition to that, an access control list is maintained for the system resources. This uses cryptographic authentication, configurable security policies and the intersection of access rights to get the least common access.

2. **Java1**

   In Java1, the Java Virtual Machine has several components to ensure security. The security manager approves the access to unsafe operations. The Byte Code Verifier checks the Java Byte Code for violations in the name space restrictions, for stack-over or under-flow and for illegal typecasts. The Class Loader keeps separate name spaces for local trusted classes and for downloaded, untrusted classes. A problem is that the Security Manager and the Class Loader can be cheated. Additionally, there is only one Security Manager per browser which disables to have different rights for applets in the same browser.

Fault Isolation / Sandboxing [**Moo98**] Sandboxing is another mechanism to monitor the execution of agents and to restrict safety critical operations. The untrusted code runs in a separate domain or sandbox, the so-called fault domain. Each load, store or jump command is only permitted inside the fault domain. This is implemented by conditional address checks or overwriting upper address bits such that each address falls into the fault domain. Sandboxing has a better performance than interpreters and is cheaper in terms of code overhead. However, the downloaded code is no longer platform-independent, because the addresses have to be mapped into the fault-domain.

Code Verification / Proof Carrying Code (PCC) [Moo98, Nec97] In this approach, the author of the code compiles a proof that his code satisfies a security policy given in some logical framework by the host. This proof is sent with the agent. At the arrival of the agent, the host verifies the proof to guarantee that the code has indeed the desired properties. However, the question remains in which logical framework the security properties should be formulated to have the necessary expressiveness. Furthermore, the code is no longer platform independent and porting is not straight forward.

Market-based Resource Control [BKR98] This approach is concerned with the restriction of resources an agent can allocate at a host. If agents use too many resources for a too long time they can prevent the server from being available to other users. The main idea is that agents have a restricted amount of e-cash to pay a resource manager for the allocation of resources. Because of the restricted amount of e-cash, agents can only allocate a limited number of resources at a time. This enables agents to use the server's resources in an equal proportion. Also it prevents denial of service attacks caused by a small number of agents blocking all available resources. Additionally, the price for resources can be set dynamically depending on the demand for resources to reduce bottlenecks. However, agents can try to cheat during payment, e.g. acquire resources without paying for. This could be prevented by introducing an arbiter agent where a deposit is left that is lost if an agent misbehaves.

2.2.2 Protection of Agents against Malicious Hosts

Protection in this direction is more difficult since the host or platform certainly needs access to the agent's code and control state in order to execute it. Therefore, it can read and alter the agent's data in plain text. Important questions here are how sensitive data can be kept secret and how the honest execution of the agent can be guaranteed. The following two approaches focus on the protection of data the agents collects or computes on his way, whereas the last three techniques concentrate on ensuring a correct execution. The approaches are ordered according to their strictness.

Detection Objects [Mea97] Detection objects are a way to detect intensional modifications of the data an agent carries with itself. Therefore, detection objects, which are dummy data items not used by the agent, are added. These detection objects will not be modified during a correct execution of the agent. But if the agent comes back to its owner and the detection objects are modified, it is clear that the agent has been tampered with. For instance, an incredible low offer for a product is added as a detection object if the agent is looking for cheap offers for this product. If the agent comes to a malicious merchant, who changes all offers the agent collected before to make his offer look the best, also the detection object will be modified. However, detection objects are only applicable for detection and do not offer protection against tampering. They have to be chosen application specific and are not usable in all scenarios. Another problem in constructing fictional data for the detection objects is that it has to be plausible enough to fool hosts, but may not influence the final results. Furthermore, it might be necessary to modify the detection objects from time to time such that it is not possible for a host to discover them by comparing several agents.

Partial Result Authentication Codes (PRAC) [Yee97] Partial result authentication as proposed by Yee in [Yee97] is a method that tries to protect the privacy and integrity of an agent's computation results. This is done by authenticating the agent's partial results before it is sent to a next host. The results are authenticated with digital signatures created with a key from a sequence of public keys the agent carries. A used key is destroyed to avoid that a host is able to change the result later. An alternative to a sequence of keys is to compute a new public key from an old one using a one-way function. Additionally, [Yee97] proposes a mechanism to publicly verify the correctness of the partial results on the agent's journey by providing it with verification predicates. However, it is not made explicit how these predicates are constructed. A drawback of this approach is that the number of hosts that will be visited has to be known beforehand to provide the correct number of keys. This problem is addressed in [KAG98] where the ideas of Yee are extended and improved. In [KAG98] the partial results and the identities of the hosts are linked together by a hash chain which prevents that results can later be modified or exchanged. This method does not need a sequence of keys anymore, but assumes the existence of a PKI. However, only the state after the agent execution can be checked and verified with these approaches. Tampering in the interaction with the agent while still on the host can not be detected or prevented.

Code Mess Up and Limited Lifetime [Hoh97] To protect agents against manipulation of code, data or control flow and to ensure the correct execution of an agent, [Hoh97] proposes the method of Code Mess Up. The agent's code is translated into an unreadable and hardly analysable format, such that it takes the host an unproportional amount of time to find out what the code is supposed to do. The lifetime of the code is restricted by an expiry time such that it is impossible to be analysed before the code expires. This mechanism does not try to detect modifications, but tries to prevent them. However, undirected modifications are always possible just by randomly altering certain bits. Another problem is to determine a reasonable expiry time for the code, i.e. the time in which it is possible to figure out the meaning of the code. Additionally, rules for the code mess up have to be fixed. Code Mess Up offers no protection against black-box-tests, sabotage or denial of execution.

Cryptographic Traces [Vig98] Since mobile agents cannot be entirely protected from damage done to them, mechanisms have to developed which detect potential tampering. One of those mechanisms is execution tracing as proposed by Vigna in [Vig98]. The executing host produces an execution protocol or an execution trace for the agent. The trace consists of pairs (n,s) where n is the identifier of a code statement and s is the input from outside. If there is no input, s is empty. After the execution, a hash of this trace and a hash of the agent's state is created. These hashs are signed by the host and transmitted with the agent. The trace is stored at the host in case the agent owner doubts the correct execution of his agent. Then he requests the trace from the host to compare it with the hash. If necessary, the trace is re-executed and so a cheating host can be identified. If the initial state of an agent is signed before it is sent to some host, it can be prevented that hosts lie about the initial state of a received agent. However, this method has some serious drawbacks. It cannot be detected if a host lies about input from the outside. Also the applicability might be restricted because of the hugh overhead produced by the storage of traces. A general problem of detection is that it is only possible a posteriori. Participants have to be made liable after the detection of cheating.

Encrypted Functions [ST98] Encrypted functions are the only mechanism that hides the semantics of the agent. The host executes the agent and computes some function. But it does not know about the semantics of the program because both the function and its result are encrypted. The mechanism works like this: firstly the agent owner encrypts the function f to E(f) and creates a program P(E(f)). Then the agent is sent to a host dispatched with P(E(f)). At the host P((E(f))(x) executed and E(f)(x) is computed. Back home, the owner decrypts E(f)(x) and obtains the result f(x). The evaluation of the function f(x) is completely secret and does not reveal anything about its semantics. Since the host does not know about the semantics of the computation, it cannot directly modify its result. This mechanism tries to prevent intensional attacks to the functionality of agents. However, not all functions can be expressed as encrypted functions. [ST98] shows that polynomials are expressible as encrypted functions. In [ACCK01], results are presented that extend this to logarithmic and polynomial size circuits. But research has not gone so far yet that encrypted functions can be used in a broad range of applications. This method can not be used if interaction with the host is dependant on the computed results since the host will not understand those. Indirected attacks, like randomly altering certain bits, are still possible and undetectable.

2.2.3 Protection in Both Directions

The approaches to be presented in this section protect agents and hosts likewise. The first method presented makes use of fault-tolerance techniques, while the second checks the state of the agent to detect modifications and to protect the host.

Fault-Tolerance Approaches Approaches used to ensure the availablility of a system can be transfered to the area of mobile code security. For instance, server replication, a fault-tolerance method, can be combined with cryptography to enhance the confidence in computed results. The servers or hosts in the sytem are replicated. An agent visits some of these replicated servers and uses voting and secret sharing or resplitting to find out what the most likely result of a correct execution is. It simply compares the results it got from all servers and decides to accept the result that has been computed in most cases. However, this approach relies on the assumption that servers fail or cheat independently. But this is contradicted by the fact that they are all under the same control.

Another approach works with agent replication. Agents are replicated and sent along different paths with the aim to detect malicious hosts. Supposing two agents are sent on the same path, but in reverse order. A modification by a malicious host can be detected if only one hosts cheats by comparing the results of those two agents. However, [Yee97] only shows for a special case that this approach is a solution of the malicious host problem.

Authentication and State Appraisal [FGS96a] [FGS96a] proposes a technique which checks agents arriving at the host before starting the execution to protect hosts from executing malicious agents and to detect modifications of agents. This can also be used to prevent agents from gaining dangerous access to the hosts's data and resources. At the arrival of an agent at a host, a state appraisal function determines the permits that the agent requests from the host, i.e. the resources it will need, after successfull authentication. An authorisation mechanism establishes which permissions will be granted. The state appraisal function depends on the agent's current state which allows to check this state at arrival, e.g. for some invariant conditions. Assuming that a host would only accept agents whose states satisfy certain conditions, malicious, modified or corrupted agents can be refused at this point. So misuse of agents can be prevented. However, not all state alternations, and not even all dangerous modifications, can be detected since detection depends on the checked conditions.

3 Comparison Shopping – A Case Study

In this section, we present the security analysis of the comparison shopping scenario which is done in the following way. Firstly, the concrete scenario to be considered is clarified. Secondly, the acting entities are identified and their interests and expectations in the single phases of the scenario are analysed. Thirdly, it is investigated which possibilities and incentives an attacker would have.

3.1 The Scenario

The electronic marketplace or virtual mall considered for comparison shopping consists of a set of merchants that offer their products, a set of matchmakers that provide a directory service about the merchants at the portal of the mall and a set of customers that are willing to shop at the merchant that matches their preferences best. Customers send their agents to a matchmaker and then to merchants in order to collect the required information. Afterwards, they decide where to buy from. Customers, matchmakers and merchants are connected via a network in which the agents roam.

The comparison shopping problem consists of the following parts as described in [DEW96]:

- A domain description, including information about product attributes useful for discriminating between different products and between variants of the same product (e.g. name, manufacturers, price...)
- A set of addresses of potential merchants
- An attribute A by which the user wants to compare the vendors
- A specification of the desired product in terms of values of selected attributes **Determine:** The set of vendors where the desired product is available sorted by the given attribute A.

Suppose we like to find the cheapest price for a specific software program or to find a certain book with the shortest time of delivery. This problem can be solved with a mobile customer agent in the following way:

1. The customer dispatches an agent with a description of the desired product and the attributes to compare different offers.
2. The agent visits a matchmaker to obtain information about merchants in the virtual mall. The matchmaker is situated at the portal of the virtual mall and simplifies the search for relevant merchants.
3. The customer agent visits all merchants advertised by the matchmaker and enquires about the desired product. The merchant submits an offer, specifying price, delivery costs, delivery time etc.
4. After having visited all relevant merchants, the agent returns to its owner and reports his findings ranked according to its owner's preferences.

The comparison shopping scenario can be refined into different phases in order to get a deeper understanding for evolving security requirements. This refinement is done with respect to existing consumer buying behaviour models in the literature. There are many different models that try to characterize the process in which a consumer is buying something from the first recognition that he might need something to the final purchase or even beyond. The Nicosia model (Francesco Nicosia, 1966), the Howard-Shet-model (1969), the Engel-Kolat-Blackwell (EKB) model or the Consumer Decision Process Model (CDP) by Blackwell, Minard and Engel (2001) are models of consumer buying behaviour, to name only a few.
The Consumer Decision Process Model (CDP) [Sch01] splits the consumer buying process into seven fundamental stages. It starts with the need recognition phase, where the consumer realises that he has got some need or problem. In phase 2, search for information, the consumer starts to look for information how he can satisfy the unmet need. Phase 3 is called pre-purchase evaluation of alternatives where the customer knows how he wants to satisfy his unmet need and investigates options where to buy. In Phase 4, the purchase phase, the customer finalises his choice what to buy and where to buy. The phase is subdivided into two subphases, where firstly the choice for the product is made and secondly the in-store choices are finalised. Phase 5 is called the consumption phase, in which the customer has got the product in his possession. In phase 6, the customer evaluates the experiences he has had with the product. The last phase is the disvestment phase, in which the customer decides whether to dispose, sell or recycle the product.

Overview of the Consumer Decision Process (CDP) Model:

1. Need Recognition
2. Search for Information
3. Pre-Purchase Evaluation of Alternatives
4. Purchase
    (a) Customer finalises choice of retailer from options investigated.
    (b) In-Store Choices (specific salesperson, payment method)
5. Consumption
6. Post-Consumption Evaluation Behaviour
7. Disvestment

The second model that was considered in order to identify the phases for the comparison shopping scenario is the Consumber Buying Behaviour Model [GM98]. The CBB model comprises six fundamental stages of many other buying behaviour models.

Its first phase is the problem recognition where the customer finds out that he might need something. Then he starts to investigate which alternatives might satisfy his need in the information search or product brokering stage. After that, he evaluates these alternatives by looking around shops and tries to decide where to buy. The fourth stage comprises the actual buying decision. Purchase, including payment, and post-purchase evaluation are the last phases in the model.

Overview of the Consumer Buying Behviour (CBB) model:

1. Problem Recognition
2. Information Search
3. Evaluation of Alternatives
4. Purchase Decision
5. Purchase
6. Post-Purchase Evaluation

Based on the models of consumer behaviour, the comparison shopping scenario can be divided into four different phases:

- Phase 1 - Information Search/Product Brokering

  Phase 1 covers comparison shopping without the customer's intention to buy anything. The customer just walks around the mall and tries to find out what products are on offer and what he might like to buy. His interest is to get to know what a possible price range for a product might be like. He evaluates the attributes for his preferences without wanting to buy something. He does not want to enter any liabilities and does not need any provably true information.

- Phase 2 - The 'real' Comparison Shopping

  This phase is the actual comparison shopping stage. The consumer compares what he knows about the different products and brands with what he considers important before deciding what to buy. He monitors the different attributes of the product and the features of the store visited. For many customers, it is essential to the buying decision to trust in a merchant. A prerequiste for this stage is that the consumer knows the need or the problem he has. In this phase, it is definitely the customers intension to buy something, but he has yet not decided where to buy. Therefore, his requirements for security, here particularly regarding the trustworthiness of the merchant, are higher than in the preceding phase.

  The phases 1 and 2 correspond to phase 3 in the consumer buying behaviour models described above. In both, the customer dispatches his agent with a product description and his preferences. The agent contacts the matchmaker at the portal of the mall to find out about appropriate merchants. It visits the advertised merchants and evaluates the values for attributes of the desired product. The products are ranked according to a given attribute, e.g. the price. Finally, the agent returns to its owner and reports its findings.

- Phase 3 - Commitment/Purchase Decision

  In phase 3, the customer finalises his decision. The choice among the possible alternatives is based on the 4 Ps, namely Product, Price, Place and Promotion [TMP+97]. The consumer confirms with the merchant what he wants to buy and for which conditions. Then he orders the product by making a legally liable contract. After that, the conditions of purchase are fixed and cannot be changed without mutual agreement. The customer remains no longer remain anonymous since he has to enter liabilities. Therefore, it is essential that his identity is known undeniably and verifiably although the content of the contract can be kept secret. In general, there are two ways of how the decision to buy something somewhere can be made. Either the agent himself makes the decision based on his findings in phase 2 or the agent makes the decision in interaction with its owner. In our approach, the second possibility is adopted. This purchase or commitment phase corresponds to phase 4 in the CDP and CBB model.

- Phase 4 - Purchase and Payment

  The fourth and last phase considered is the payment phase. Note that the physical delivery is not modelled since this would involve threats that are not computer specific and caused by transport companies and alike. This phase is similar to parts of phase 5 in both models. According to the contract made in phase 3, the customer pays the desired product in this stage. In general, there are different ways available to pay in eCommerce which have all their strengths and weaknesses. Possibilities are payment by bill, bank draft or credit card, to mention the more conventional ways. Other possibilities are Paybox [Pay] or other forms of eCash.

3.2 Security Analysis

In the following, the comparison shopping scenario is analysed focussing on the interests and expectations of its participants regarding security. The potential actions of an attacker threatening the system are considered. In addition to the phases, different instances of a comparison shopping scenario are investigated using the example of high price and low price goods.
The participants in the scenario are customers, merchants situated inside the virtual mall and matchmakers at the portal of the mall. Matchmakers provide customers with information about the merchants inside the mall. Furthermore, the network owner is considered in order to analyse the security requirements with respect to the network. In this analysis, it is omitted that agents are able to contact other customer agents inside the mall to obtain information about merchants. That would introduce new security aspects, for instance, whether an agent can trust such information or not.

3.2.1 Roles and their Interests

- Interests of Customers

  In a first information search phase, the customer wants to find out what a merchant has on offer for which price. He expects to be informed about all interesting products and the attached conditions. He does not want to enter any liabilities just by looking around and does not want to be forced or required to buy anything. It is his main objective to get the desired product for the best possible conditions. In the second stage, where the customer actually intends to buy something, he wants to get exhaustive information about products and their attributes matching his preferences. He requires this information to be correct which he wants to base his commitment on.

  When the customer wants to commit himself, he wants to make a legally binding contract with the merchant that also holds as legal evidence in case of litigation. The product has to be available and has to be delivered for the conditions the customer was told. The content of the contract can be kept confidential if both parties agree on that. The customer does not want to be deceived by the merchant. He wants to be sure that the merchant he is contacting is exactly the one he thinks he is negotiating with. He wants to provide his personal data only for agreed purposes and wants to prevent that the merchant misuses his data for unintended purposes such as profiling or advertisment. When it comes to paying, the customer wants to use a secure, but convenient method of payment. He does not want to be deceived by the merchant by billing more than it was actually agreed on. Additionally, he wants his payment information to be protected against misuse, e.g. the merchant should not forward his credit card number to any other merchant. He wants the merchant to behave trustworthily, for instance not to sell products he cannot supply or to deliver the product after payment. Furthermore, a customer expects that the merchant sticks to the conditions fixed in the contract.

  Regarding the matchmaker the customer wants to get all relevant information about appropriate merchants. The list provided by the matchmaker should be exhaustive and contain no irrelevant information. With respect to other customers, he expects them to behave in a competative, but fair manner.

The customer wants the merchant and the matchmaker to be available and provide a service of sufficient quality and also that they behave reliably and trustworthily. It is important for him that his data (like partial results) and his code are not manipulated by some external attacker or platform. Furthermore, he wants to stay anonymous and maintain his privacy. The customer expects that his agent is executed as it was programmed and that it can migrate as intended.

- Interests of Merchants

  It is the main interest of the merchant that customers buy at his store in order to make the best possible profit. A merchant wants to attract a customer's attention for instance by offering good products and prices, granting attractive conditions of purchase and having a good reputation. Furthermore, the merchant wants his store to be available such that customers can visit it. Additionally, the integrity of his data and working principles should be guaranteed. Possibly, the merchant wants to issue some confidential offers which should indeed be kept private by the customer. Phase 1 and 2 do not make any difference for the merchant since he cannot distinguish whether a customer intends to buy something or not.

  When a customer commits himself, the merchant wants to make a legally binding contract with him. The contract should hold as evidence in court in order to prevent that the customer refuses to pay for a delivered product, for instance. The merchant wants the customer to provide him with correct information about his person to make a correct contract. This contract can be kept secret by both parties. Additionally, he wants the customer to authenticate himself such that he can be sure whom he is communicating with.

  At the payment stage, the merchant's main interest is to get the agreed amount of money from the customer as fixed in the contract in a convenient manner. The merchant expects the customer to be reliable and trustworthy in that he gives correct information, sticks to the contract and fulfils his obligations. This includes the payment of the product.

  Regarding his fellow merchants, a merchant expects them to behave competatively, but fairly. They should not perform any illegal actions. The matchmaker, in the merchant's view, should inform the customers about himself and his products, be available and trustworthy.

- Interests of Matchmakers and Network Owner

  The network owner wants his network to be reliable and secure in all phases in order to attract users and to maintain the infrastructure. Furthermore, he wants to keep out criminal actions like sabotage or manipulation. The users of the network expect it to be reliable and secure. They want their communication over the network to be confidential, i.e. that communication cannot be disclosed, monitored or manipulated.

  The matchmaker is more a mean to an end and not an end in himself. Therefore, he is not assumed to have any interests on his own. He simply offers a service to all enitities that contact him. However, his clients expect him to provide a sufficient quality of service, i.e. that he provides exhaustive and relevant information, is available and non-manipulated.

- Interests of an Attacker

  In this scenario, an attacker an either come as a malicious merchant, matchmaker or customer, as a malicious platform or as someone unknown from the outside. The attacker's interest is to perform legal as well as illegal actions to maximize his utility. An attacker can use legal working principles of the system for unintended purposes, such as denial of service attacks by making too many requests. A major interest of an attacker is to remain undiscovered.

  One objective of the attacker can be to gain useful information for himself. He can try to compromise customer privacy and anonymity to find out what products the customer looks for. He can achieve information gain by pretending to be a platform, merchant or matchmaker which the agent trusts in. Furthermore, he can try to disclose secret offers and contracts. Another way to obtain information is by disclosing the network communication.

An attacker can sabotage platforms and restrict their availability in order to have more customers visiting his site and to pretend to be a better choice for customers. Manipulation of data or working principles, sabotage or denial of service attacks can restrict the availability, reliability and quality of service of merchants, matchmakers and platforms. So the competition of the market can be influenced.

A malicious merchant can provide the wrong conditions of purchase. He can misuse the information he got from the customer for unwanted purposes such as profiling, reselling or advertising. He can cash more than he was actually entitled to, or he can refuse to deliver the product after payment. A malicious matchmaker can distribute incomplete, irrelevant or incorrect information about merchants favouring particular merchants. A malicious customer can provide false personal information or refuse to pay a received product. A malicious hosts can refuse to execute a customer agent as it was programmed. Also, he can refuse to send an agent where it wants to go to.

In the first and second phase, an attacker can manipulate the customer's already collected offers. The reason for that can be that the attacker wants have the best offer himself or that he collaborates with other merchants which he wants to look best. In the payment stage, the incentive for attacks is even greater because real money can be gained. So payment information, e.g. the credit card number of a customer, can be obtained to get money of the customer's account or to resell it.

3.2.2 Different Instances of the Scenario

The analysis of different instances of comparison shopping gives an impression how security requirements evolve. One example for different instances is the purchase of high price goods in contrast to low price goods. High price goods are, for instance, cars, houses or something which is not usually bought every day or every month. Low price goods, however, are things that are bought more often, like CDs, books or alike. It seems natural that the interests of customers and merchants differ in these cases since the risks increase with the higher price of the product. Consequently, there are differences in the security requirements people have both instances.

With low price goods, it seems to be less serious for the customer if something goes wrong because the financial damage is smaller. In the high price case, fraud, deception and other attacks are more severe since the amount of money involved is higher. Additionally, fraud and deception seem more likely since the expected gain is higher, if the manipulation remains undetected. Because of the higher risks with high price goods, people require greater reliability and trustworthiness of the system.

Looking at the phases, we have identified previously, differences between the high and the low price case can be observed. In phase 2, the comparison shopping phase with the intention to buy, the customer wanting to buying something more expensive definitely requires correct information about the product, because false information can lead to serious financial harm. In some cases, it is not easy to determine the actual value of a product. In case of a car or a house, a trusted third party or a censor is needed to estimate the actual value of the object. For phase 3, the contract, that is eventually made, has to be indeed legally binding, since in case of litigation this contract has to be valid evidence in court. Also the payment method used in phase 3 must be more secure for high price goods because of the higher financial risks.

To sum up, the difference between high and low price goods is that the security requirements for high price goods are higher. Whereas the technical threats remain more or less the same, the application-oriented threats, i.e. the opportunities for fraud, increase. In order to counter fraud, the trust a customer has in a retailer before commitment should be higher.

3.3 Overall Security Threats and Security Objectives

In the previous analysis, we illustrated occurring security problems and the requirements of users to a secure system. From that, we set up an overall view of the threats to the mobile multiagent system in the virtual marketplace. We will identify security objectives to counter those threats and to satisfy the security requirements of the system users. The threats will be grouped into different threat scenarios.

Threat Scenario 1 – Data Security
The first threat scenario comprises all threats that are concerned with the misuse of data, or more precisely, the unauthorised disclosure, copying or modification of data. All data that occur in this scenario can be used in an unintended manner if they are unprotected. The data of an agent comprises its code and the data it carries, like collected offers, identity information, contracts made with merchants, or payment information. This data can be copied, disclosed or modified. An interesting instance is the case in which an agent has collected several offers from other merchants and visits another merchant. This merchant can modify all other previously collected offers such that his offer seems to be the best. Another critical point with respect to confidential data is the leak of data without permission of the owner. In addition to that, the inter-agent communication can be disclosed and modified by a malicious platform. Malicious agents and other attackers can try to disclose, copy or modify the data that is stored at the platform and also the platform's code and working principles. For instance, a Trojan horse can be inserted into the platform's code such that someone else gains control over the platform.

T1 Unauthorised Disclosure, Copying and Modification of Data or Code T1.1 Disclosure of identity
T1.2 Disclosure of secret offers
T1.3 Disclosure or manipulation of contracts
T1.4 Modification of already collected offers
T1.5 Disclosure and modification of payment information
T1.6 Modification of agent's code
T1.7 Modification of agent's data
T1.8 Modification of host's code
T1.9 Modification of host's data
T1.10 Disclosure of submitted messages between agents
T.1.11 Modification of inter-agent communication
T1.12 Unauthorised passing on of confidential information

Security Objective 1 – Protection of Data
Resulting security objectives are that the agents and platforms can protect their and their data and code from unauthorised copying, disclosure and modification. It should be possible to detect and to prevent that confidential information is passed without permission. Additionally, the customers should be able to stay anonymous as long as possible before eventual commitment.

SO1 No Unauthorised Disclosure, Copying or Modification of Data
SO1.1 Only authorised access to agent's data and code
SO1.2 Only authorised access to host's data and code
SO1.3 No unwanted disclosure of identity
SO1.4 Only authorised access to special offers
SO1.5 Only authorised access to contract information
SO1.6 Only authorised access to payment information, no unauthorised modification of payment information
SO1.7 Confidential and integer inter-agent communication
SO1.8 Detection and prevention of the unauthorised passing on of confidential information

Threat Scenario 2 – Interception of Network Communication
This threat scenario deals with the security of the network communication. Here, the network that connects the platforms with each other is considered. Some malicious attacker from the

outside can try to disclose the messages transferred between platforms in order to read or modify them. Furthermore, he can analyse the traffic on the network and extract knowledge about who communicates with whom. Furthermore, he can block messages from being received, or he can remove them from the network.

T2 Interception of Network Communication

T2.1 Disclosure of network communication

T2.2 Modification of network communication

Security Objective 2 – Protection of Network Communication

The resulting security objectives are the confidentiality, privacy, integrity and reliability of the network communication.

SO2 Secure Network Communication

SO2.1 No disclosure of transmitted messages

SO2.2 No modification of transmitted messages

SO2.3 Reliable network communication, i.e. sent messages are received by the intended recipient

Threat Scenario 3 – Restrictions to Availability of Services

In our scenario, restrictions to the availability of services can happen to agents, platforms or to the network. The agents are mainly threatened by malicious platforms they are on. They can refuse to execute an agent, or they can not execute it as it was programmed by its owner. Furthermore, malicious platforms can refuse to transmit agents to other platforms or transmit them to an unwanted platform. Moreover, they can kill an agent such that it is neither executed nor transmitted any further. Additionally, platforms can control the communication between two agents being on the same platform, i.e. they can prevent, alter and forge messages sent between agents.

Similarly, the platforms and the network are threatened by malicious agents or other attackers from the outside. Potential attacks are denial of service attacks where too many queries are submitted such that a service breaks down. Alternatively, resources can be corrupted or blocked such that a satisfying service is no longer possible.

T3 Sabotage, Restrictions to Availability of Services

T3.1 Incorrect or non-execution of agents

T3.2 Sabotage of hosts (e.g. by denial of service)

T3.3 Sabotage of network

T3.4 Incorrect transmission of agents from platform to platform

T3.5 Non-working inter-agent communication on a platform

Security Objective 3 – Availability and Quality of Services

The security objectives concerning availability and quality of services are the correct and complete execution of agents, the reliable communication between agents and the correct transmission of agents between platforms. Correctness and completeness of agent execution means that agents are executed as programmed by their owner. Reliable communication between agents describes that sent messages are received by the intended recipient. If agents are transmitted as described in their code we say that they are transmitted correctly. The objective concerning platforms and network is that the service is available at the quality that meets the user's expectations.

SO3 Reliability and Availability of Services

SO3.1 Correct and complete execution of agents

SO3.2 Availability of hosts

SO3.3 Availability of the network

SO3.4 Correct transmission of agents

SO3.5 Reliable inter-agent communication on a platform

Threat Scenario 4 – Masquerading vs. Authentication

This threat scenario deals with the identity of the participants, like agents and platforms. An participating agent can try to masquerade itself to harm an opposite party. Similarly, platforms can masquerade as a trusted platform to extract secret information of an agent.

T4 Masquerading

T4.1 Merchants masquerade to fool customers.

T4.2 Customers masquerade to harm other participants.

T4.3 Platforms masquerade as trusted platforms.

T4.4 Matchmaker masquerades.

Security Objective 4 – Authentication

The security objective with respect to authentication is to create a unforgeable link between an agent or a platform and its owner. Therefore, each agent needs to have a unique identity which a communication partner can verify. Platforms need the capability to authenticate themselves and prove their identity to an agent. In contrast, a customer agent should be able to stay anonymous.

SO4 Authentication and Verification of Identity

SO4.1 Each entity in the system, agent or platform, has got a unique, verifiable identity.

Threat Scenario 5 – Fraud vs. Trustworthy Behaviour

This scenario comprises all threats that cannot be classified as classical security issues and are concerned with fraud and legal problems. With respect to legal behaviour, we can observe the following threats mainly connected to contracts and payment information: Contracts may be faked, e.g. by using masquerading. Entities involved in a contract can deny their commitment. The payment information a merchant obtains can be used to cash more than agreed on in a contract. A customer can give false payment information to deceive the merchant. Merchants can provide customers false information on products or issue false offers which becomes a legal problem if offers are considered legally binding.

Regarding trust and quality of services, we see that the given information from the matchmaker, i.e. the list of matching retailers, can be incomplete or irrelevant leading to a disadvantage of certain retailers. Furthermore, matchmakers and retailers can try to misuse or resell the information they have got about their clients for profiling or advertisments. Other threats, which are out of the scope of this paper, are related to the delivery of the product after payment.

T5 Fraud

- Trust Related Issues / Quality and Policy of Services

T5.1 Insufficient quality of service, e.g. incorrect or incomplete list of merchants given by matchmaker

T5.2 Misuse of information (reselling, profiling, advertising...)

- Legal Liability

T5.3 Faking of a contract (e.g. by masquerading)

T5.4 Denial of contracts

T5.5 Give incorrect payment information

T5.6 Misuse of payment information, cash more than being entitled to

T5.7 Distribution of false information, particularly offers

- Not in the scope of this paper, Payment and Delivery

T5.8 Non-delivery of product

T5.9 Refuse to pay product after delivery

Security Objectives 5 and 6 – Legal Liability and Trusted Services

There are two security objectives to counter these security threats. All threats concerned with legal behaviour can be countered by introducing legal liability, for instance making contracts and offers legally binding such that fraud can be sued later. The issues linked to trust and quality of services can be counteracted by introducing the objective 'Trusted Services'. This objective can be achieved by giving services the possibility to prove that they work according to a quality of service policy.

SO5 Legal Liability
SO5.1 Merchants can issue legally binding offers.
SO5.2 Customers and merchants can make legally binding contracts.

SO6 Trusted Services
SO6.1 Matchmakers can prove that they operate a quality of service policy.
SO6.2 Merchants can prove to the customer that they operate a quality of service policy.

3.4 Remarks on the Security Threats and Objectives

The identified security threats can be split into technical threats and application-oriented threats. The technical threats are independent of the concrete scenario in which the agent-system is applied and are countered by technical means, whereas the application-oriented threats can depend on the concrete scenario in which the system is used or and on the phase of an agent's activity. Roughly speaking, the application-oriented threats can be summed up with the term 'fraud'. It is hard to come up with means to counter them.

Fraud also happens in the real-world and a priori less can be done to prevent it. Only after fraud was detected, the responsible person can be sued and punished for it. Before fraud occurs, the only way prevent deception or to minimize its probability is to interact with someone one trusts in. In the comparison shopping scenario, it is the same problem with fraud. It cannot be countered with technical means alone. To reduce the possiblity that fraud occurs, we introduce a trust model such that agents can take appropriate measures before contacting an untrusted entity. Additionally, legal liability can decrease the possibility of fraud and deception.

In our scenario, the technical threats are the security threats 'T1 Unauthorised Disclosure, Copying or Modification of Data or Code', 'T2 Interception of Network Communication', 'T3 Sabotage, Restrictions of Availability of Services' and ' T4 Masquerading'. 'T5 Fraud' comprises the application-oriented threats. It can be partially countered by the introduction of legally binding contracts and offers in SO5, but it also needs the establishment of a trust infrastructure as in SO6.

Looking at the individual phases of the comparison shopping scenario, the following changes in the security threats can be observed. During all phases the technical threats T1 – T4 remain more or less the same. Only in T1, the issue changes over the different phases. In phase 1 and 2, an agent's identity has to be protected. Additionally, its collected offers should not be manipulated. Also, merchant's offers should remain secret if necessary. However, from phase 3 on the customer can no longer stay anonymous since he commits himself. In this phase, the contract between customer and merchant has to be protected. In phase 4, payment information is the main concern. Furthermore, in all phases the data and code of agents and hosts should not be disclosed or manipulated.

With respect to the application-oriented threats, one notes more severe changes over the single phases. Generally, the more commitment a participant makes the more security is required. From phase 1 to phase 2, the correctness of product information becomes important, since in phase 2 the customer wants to base a decision on it. In phase 1 and 2 the customer's activity is by no means legally binding, whereas in phase 3 he commits himself, and he enters legal liabilities. Therefore, in phase 3 we need more trust into merchants than before. Also, with low price goods fraud is not such a serious issue, but it increases as the price increases.

4 Towards a Secure System

Using the results from the previous analysis, we develop a setting of a secure mobile multiagent system for comparison shopping. We start with assumptions about the underlying technical model. Then, we transform the security objectives into building blocks which the underlying system architecture has to provide in order to accomplish those objectives. The building blocks are further refined into features of agents and platforms. Finally, ideas concerning the technical realisation of the building blocks are presented.

4.1 A Technical Realisation of the Multiagent System

Generally, any platform is able to read and alter non-encrypted data of agents. Furthermore, a platform is able to manipulate agents and to influence their behaviour. For the establishment of security, the notion of a trusted platform is necessary. Therefore, we have a look on the notion of trusted platforms, before we discuss a concrete model of our electronic marketplace.
A platform is said to be trusted with respect to an functionality if an agent on that platform can be sure that the platform implements this functionality as expected. If a platform is trusted with respect to a functionality it does not exhibit malicious behaviour connected to that functionality. It is not reasonable that a platform is completely trusted meaning that no malicious behaviour occurs. But one can trust in a platform with respect to a subset of operations, i.e. the agent is safe on the platform if only operations from this subset are carried out. The following box gives examples with respect to which functionalities a platform can be trusted.

A platform may be trusted

- to execute agents as they were programmed by their owner.
- in transmitting agents between platforms as intended in the agent's code.
- to maintain the confidentiality and integrity of inter-agent communication.
- to preserve secret keys from disclosure.
- to keep secret data private.
- to preserve an agent's data from modification and disclosure.
- not to misuse the information and the knowledge (data/code) it gets from the agent.
- to provides correct authentication.
- not to compromise an agent's privacy and anonymity.
- to protect agents from each other such that agents cannot manipulate each other with respect to data or functionality.

The last point above refers to the problem that not all agents on a trusted platform are benevolent as well. It still has to be assumed that they can be malicious and try to attack the platform or other agents. Therefore, any platform should protect itself from malicious agents and protect the agents from each other, such that malicious agents cannot harm 'good' agents on the same platform.
In case of an untrusted or malicious platform, a correct or benevolent behaviour cannot be predicted or assumed. A malicious platform may behave as it wants and nothing can be said a priori about it. It can behave benevolently, but it may as well corrupt agents and be malicious in various aspects as the manipulation of the agent execution, wrong transmission to somewhere where the agent does not want to go, provision of incorrect or incomplete data and so on. While a platform can be trusted by some agent, it can be untrusted for other agents. For instance, an agent sitting on its own platform will certainly trust in it, while a visiting agent might consider the same platform as untrusted.
In order to think of technical means to counteract the threat scenarios, one has to be precise about the underlying technical model of the agent system. In general, there are two configurations how an electronic marketplace using mobile agents with these types of participants can be set up.
Firstly, one can assume that customers, retailers and matchmakers are implemented as mobile agents that meet on a common platform, the virtual mall. In this approach all agents play

the same role. However, the databases of matchmaker and retailer agents are not necessarily connected to the platform the agents are currently on. In this case, the data the agents need must be transferred over the network to that platform. This leads to a high amount of traffic on the network and to bandwidth reductions. Thus, the advantage of mobile agents to travel to the data and to reduce network traffic would not hold.

Secondly, customers, retailers and matchmakers are modelled as mobile agents, but matchmaker and retailer agents operate on their own platform. Their database is directly connected to this platform and so network traffic is reduced. Only customer agents hop from one merchant and retailer platform to another. An advantage of this alternative is that matchmaker and retailer agents are situated on their own platforms that they trust in which reduces their security risks. However, customer agents are still threatened by matchmaker or retailer platforms.

In this report, the ideas on the technical realisation of the building blocks are based on the second alternative configuration. We will assume that merchants and matchmaker agents operate on their own platform and are visited by mobile customer agents.

4.2 Building Blocks

The following building blocks are set up to counter the previously established security threats and to implement the desired security objectives.

BB1 Inter-Agent Communication
Two agents must be able to communicate with each other confidentially.
The integrity of submitted messages must be guaranteed.
It has to be guaranteed that a submitted message is received.

This first building block 'BB1 Inter-Agent Communication' provides the system with the capability that agents can communicate confidentially on the same platform, i.e. noone else is able to listen to their communication. It ensures the integrity of the communication and that the messages are indeed received. This building block counters the threats of unreliable inter-agent communication T3.5 and of unconfidential and modified messages T.1.10 and T.1.11. The building block implements the security objectives S01.7 'Confidential and Integer Inter-Agent Communication' and SO3.5 'Reliable Inter-Agent Communication'.

BB2 Authentication
An agent can authenticate itself if it is required
The customer agent can stay anonymous before making a commitment.
A platform can authenticate itself.

The second building block is needed to counter all threats that are connected to masquerading, i.e. T4 and its subthreats. It implements the security objective SO4 'Authentication' and ensures that every participant in this system has got a verifiable and undeniable identity, including platforms. It also protects the identity of the customer countering threat T1.1 'Disclosure of customer's identity' and implementing objective SO1.3.

BB3 Data Protection
An agent is able to keep his collected data secret and to protect it from unauthorised modification. The data can only be accessed and modified with his permission.
Personal information, e.g. identity, payment information can be prevented from unauthorised disclosure, copying and modification.

The protection of its data is essential for a comparison shopping agent on whose findings a user wants to base his decisions. Therefore, this building block counters the threat scenarios concerned with data security, namely T1.2–T1.7 and T1.12, including the protection of offers, contracts and payment information or generally protection of data and code. It implements the security objectives SO1.1, SO1.3–1.6 and SO1.8.

BB4 Agent Execution
It is guaranteed that an agent is executed as it was programmed by its owner.
It is guaranteed that the code and the control state of an agent are not modified.
It is guaranteed that an agent is transmitted as intended between two platforms. The integrity and the confidentiality during transmission is ensured. It is guaranteed that the agent is received.

As we have seen in the analysis, the correct and complete execution of the agent is a severe security threat. This building block provides means to ensure that the agent is executed as it was programmed by its owner. That includes the correct, confidential and integer transmission of the agent. This counters the security threats T3.1 and T3.4 and implements the corresponding objectives SO3.1 and SO3.4.

BB5 Protection of Platforms and of the Connecting Network
The access of agents to data and resources of platforms is restricted and protected.
The availability of platforms is protected. Denial of service attacks can be prevented.
The availability of the network is protected. Denial of service attacks to the network can be prevented. The confidentiality and integrity of the network communication is protected.

This building block counters the threats concerning the availability and reliability of network and platforms, identified in T3.2 and T3.3. The means provided by the building blocks implement the security objectives SO3.2 and SO3.3. SO2 'Secure Network Communication' is also implemented by this building block countering T2.

BB6 Legal Liability
A customer and a provider agent are able to make a legally binding contract. The contract cannot be manipulated after it was made and cannot be denied by a signing party. If it is required, the contract can be kept confidential.
Provider agents can issue binding offers to customer agents. These offers cannot be manipulated and can be kept confidential, if it is necessary.

This building block counters the threats linked to the legal liability of offers and contracts T5.3 – T5.7 and implements the security objective SO5 'Legal Liability'.

BB7 Trusted Services
Agents can prove that the operate according to a quality of service policy.
Platforms can provide a certificate of their trustworthiness.

This building block contains the functionality that relates to the trust an agent has in the service another agent or a platform provides him. Merchant or matchmaker agents can show their clients that they work according to some certified quality policy. That can be for instance that a matchmaker will always provide all relevant and only verified information or that a merchant will not use the client's data for profiling or advertisments. That aspect of this building block counters the threats T5.1 and T5.2 and implements the security objective SO6. Additionally, the building block provides means for an agent to determine with respect to which features he trusts a platform. That addresses all threats connected to unwanted behaviour of platforms, for instance disclosure and modification of data, incorrect execution and so on. The threats are yet not directly countered. But an agent is free to decide whether it wants to visit such a platform or what countermeasures it uses before a visit to protect itself, e.g. encryption of sensitive data.

Figure 1 shows the relationship of security threats, security objectives and building blocks. The columns denote the main security threats and the rows the main security objectives. For reasons of clarity the subthreats and subobjectives are omitted. The building block in an intersection of a row and a column counters the threat in the column and implements the objective in the row. Building blocks can be mentioned several times if they contain means aiming at several threats and objectives. From the table, we can see that all threats are countered and that all objectives are implemented by at least one building block.

	T1	T2	T3	T4	T5
SO1	BB1, BB2, BB3				
SO2		BB5			
SO3			BB1, BB4, BB5		
SO4				BB2	
SO5					BB6
SO6					BB7

Figure 1: Relationship between Security Threats, Security Objectives and Building Blocks

4.3 Features of the Agents and Platforms

Features of Agents In the comparison shopping scenario, we distinguish three types of agents: customers, matchmakers and retailers. All of those agents possess general features. In addition, they are specialised with respect to their role in the system. The building blocks can be split into features of the single agent types in order to realise those blocks in the agent society. The technical realisation of the features is deduced from the technical realisation of the building blocks.

The essential features of an agent are the following: An agent is able to authenticate itself if it is required as postulated in BB2, e.g. when making a contract with a merchant or when issuing an offer. It is able to keep his internal data secret and to protect its integrity as in BB3. The communication with other agents can be performed confidentially and integrally as in BB1. Agents possess means to guarantee the intended execution. Furthermore, they can ensure the integrity of their code and control flow or are at least able to detect alternations, as required in BB4. A confidential, integer and working transmission of the agent from a platform to another platform can also be guaranteed, as in BB4.

Additional features of a customer agent are the ability to hide his identity and stay anonymous if he wants to as in BB2, the capability to make legally binding contracts with merchants, as in BB6, and the ability to provide payment information only to the entitled entities, as in BB3. A feature of the merchant agent in addition to the general features is the ability to make legally binding contracts with customers, as in BB6. The matchmaker and the merchant agent are able to prove their quality of service policy to their clients, as in BB7.

Features of Platforms The requirements that were established as building blocks can also be formulated as features of platforms. Some features are concerned with the self-protection of platforms, whereas others affect their trustworthiness. Even a malicious platform has an interest in protecting itself. Thus, it can be assumed that any platform possesses the features concerned with self-protection. But only a trusted platform can be assumed to have benevolent features that go beyond self-protection.

A platform should employ access and resource control mechanisms to protect its data and resources against unauthorised access and modifications. Furthermore, mechanisms to protect the availability and quality of the provided service should be available for a platform. Mechanisms to react to denial of service attacks are desirable, although there are yet no serious and reasonable solutions. The above features are all mentioned in building block BB5. Apart from these aspects in which the platform has an interest in itself, other features are required in order to establish trust in a platform. A trusted platform can apply technical means to guarantee the execution of an agent as it was programmed. Furthermore, it can guarantee the correct transmission of agents by a transmission protocol. The confidentiality and integrity of an agent's data can be protected by cryptographic means. A platform can isolate agents on it from each other. Furthermore, it can provide certificates for authentication. These features are derived from the building blocks BB4 comprising agent execution and BB2 comprising authentication. BB7 also suggests that a platform can provide a certificate of its trustworthiness.

4.4 Towards a Technical Realisation

4.4.1 General Remarks on the Achievability of Security Objectives

If one starts to think of a technical realisation of the proposed building blocks, firstly some remarks have to be made about principle restrictions of mobile agents with respect to security goals based on a paper by Farmer, Guttman and Swarup [FGS96b].

It is unachievable to find out whether a platform or interpreter has been tampered. Since this question only arises with untrusted hosts mainly run by competitors, noone will allow to inspect the platform's code. A program running on this platform to check tampering might as well get tampered. Testing will not always detect that the platform has been modified since any modification is designed such that it will not be detected at first sight. Similarly, it cannot be known a priori whether an interpreter or platform will execute an agent as it was programmed or whether it will run the agent to completion. Since the platform has total control over the agent and the agent is essentially passive itself, there is not way to prevent incorrect execution. Equally, the platform might decide to stop the execution of an agent. Neither can be known whether a platform will transmit an agent as requested. Similar to non-execution, a platform can deliberately decide not to submit an agent. Because the platform has access to all non-encrypted data of an agent, only encrypted data and encrypted code not needed on that host can be kept private. If certain data should not be known to a platform, it has to be encrypted before going there.

However, there are also some easily achievable security goals. For example, the author or the sender of an agent can be authenticated by signing their code. The integrity of code can be checked by verifying an attached signature. Privacy during transmission of an agent can be ensured by encrypting it before sending it. Interpreters can protect themselves against malicious agents by employing access control mechanisms checking the agent's author, its program and its state.

Encryption techniques cannot be used to the same extend in systems with mobile agents as in other settings. Digital signatures and asymmetric encryption require the existence of secret and public key pairs that are provably assigned to specific entities. Therefore, a public key infrastructure (PKI) is necessary for applying those techniques. This infrastructure sets up mechanisms for the distribution of public keys and for allocation of those keys to entities. Assuming a PKI in a system, public keys are uniquely attached to each entity. A public key of an entity can for instance be obtained by getting it from a central key server.

Asymmetric encryption techniques require public keys for encryption and secret keys for decryption. Each generation of a digital signature involves the use of a secret key. Agents visiting a possibly malicious platform cannot carry a secret key in plain text because the platform would find out about it. Therefore, an agent can not use a secret key on such a platform.

Taking that as a prerequisite, privacy of data is nevertheless easily achievable. For encryption of data in the asymmetric case only public keys are necessary. Public keys uniquely attached to an entity are available everywhere by means of the PKI. If an merchant wants to sent an confidential offer to a customer, it can ask its platform to get the customer's public key and to encrypt the offer with it. Only the customer that possesses the corresponding secret key can read that message. But decryption of the offer is only possible on a platform that the customer trusts with respect to the preservation of secret keys. Elsewhere the secret key would not be available.

Authentication and integrity can be achieved with digital signatures. A digital signature is the hash of a message encrypted with the sender's secret key attached to the message. The receiver can verify the signature by generating the hash of the message, encrypting it with the sender's public key for the PKI and comparing the received signature with the self-generated signature. If those are the same, the message has not been altered. This approach relies on the assumption that a slight modification of the message would yield a completely different hash. Digital signatures add accountability to the communication by attaching the sender's secret key to the content of a message.

Authentication of merchants or matchmakers sitting on their own platform is also easy since they can sign an offer or a matching with their secret key. Merchants and matchmakers always trust their own platform with respect to the preservation of their secret key. Authentication of agents, however, is much harder. Agents do not carry their secret key to an untrusted platform. On their travel through the mall, customer agents leave their secret key on a trusted platform or have it encrypted before visiting an untrusted platform. They have to migrate back to a trusted platform either to fetch their key or to have it decrypted in order to sign something.

For the execution of an agent, the platform needs to have total control over the agent. All code and data has either to be in plain text or encrypted with a key that the platform owns to be useable there. So the platform will always be able to find out about the program semantics and can potentially manipulate data or code. Only encrypted functions [ST98] offer complete protection against the platform finding out what an agent is actually doing. In this approach, the function and its result are encrypted such that it cannot be understood by the platform. However, there are still no encrypted functions available for practical applications. Research results prove that encrypted functions exists for instance for polynomials. Thus, the security mechanisms for our system can not make use of this technique. But if there were such functions, security of mobile code would be much easier to achieve.

Several trusted platforms can be combined to so called *trusted island*. These island form a compound structure in which no precautions against malicious platforms have to be taken. Standard encryption and authentication are applicable on these trusted island since secret keys are available anywhere. Transmission between platforms inside a trusted island can be made secure by encryption techniques. The same holds for the agent execution, the inter-agent communication and so on. Before leaving a trusted island, agents can leave sensitive data as well as secret keys inside the island. However, one has to think of mechanisms how to establish such islands, how to certify them and how to authenticate them, such that an agent can be sure that it is on such an island. On the other hand, trusted island have to be cautious which agents they allow to come in to protect themselves. This requires also authentication of the agents.

4.4.2 Aspects of Technical Realisations for the Proposed Building Blocks

BB1 Inter-Agent Communication
Two agents must be able to communicate with each other confidentially.
The integrity of submitted messages must be guaranteed.
It has to be guaranteed that a submitted message is received.

Aspects of a Technical Realisation:

On a trusted platform, the confidentiality and the integrity of the inter-agent communication is guaranteed by the definition of a trusted platform. Such a platform will of course learn about the content of the communication and will be able to alter it. But because it is trusted with respect to inter- agent communication, it will not disclose and modify messages. The reliability of communication can either be guaranteed by the trusted platform or by an handshaking protocol. In a handshaking protocol, every received message is acknowledged or sent again after some delay if no acknowledgement was received.

On a malicious platform confidential, communication between two agents is impossible. In open systems, two parties that want to communicate confidentially with each would use standard encryption techniques. The approach is safe in a system with static agents communicating over a malicious network. However, this cannot be transferred to the communication of mobile agents on the same platform. It would be possible for an agent to encrypt a message with the recipient's public key, but the recipient would not be able to decrypt the message on the platform that it does not trust with respect to its secret key. Two agents can only communicate with each other confidentially if both are on a platform which they trust with respect to their public key, see section 4.4.1.

The integrity and authentication of submitted messages is also a problem. In a non-agent world, one would use digital signatures to detect unauthorised modifications. But with mobile agents, this technique is not applicable on any platform, since it requires the sender to have an accessible secret key. Mobile agents can only use a secret key on a trusted platform, compare section 4.4.1.

On a malicious platform, it is almost impossible to guarantee that a message sent by an agent to another agent on the same platform is received. In the client-server model, one would use handshake protocols where each message is acknowledged. However, in this case you will not know whether the message or just the acknowledgement has been lost or if the platform generated an acknowledgement and just threw the message away.

To sum up this reflection up, it has to be pointed out that secure communication requires a public key infrastructure. As the agent does not possess secret keys, data to be sent confidentially has to be already encrypted before the agent arrives at an untrusted platform or the agent has to trust the plattform with respect to the plaintext of the message. Messages to be received by the agent have to be encrypted with the platform's public key or – in case of an untrusted platform – encrypted with the public key of another (trusted) platform which the agent has to visit in order to decrypt the message. For integrity, this means that for each message if it is not sent on a trusted platform the digital signature has to be created on a trusted platform, before the agent migrates somewhere else to sent the message. The verification of a digital signature, however, can be done everywhere, since only private keys are needed.

BB2 Authentication
An agent can authenticate itself if it is required
The customer agent can stay anonymous before making a commitment.
A platform can authenticate itself.

Aspects of a Technical Realisation:

For the purpose of authentication agents can carry a passport stating their identity, their service description, their dispatcher's identity and whatever information is useful apart from that. The passports have to be protected from unauthorised modifications using digital signatures. Furthermore, they have to be certified by some trusted certification authority. The passports have to be closely linked to the agents such that it is impossible to take a passport of some agent and to stick it to another agent. A possibility would be to attach a signed hash of the agents code to the passport using digital signatures. Alternatively, authentication can be done using a challenge response protocol where questions are asked which only a particular agent can answer. However, a problem is that some malicious platform can extract this challege respone information from an agent camouflaging this agent by having the same knowledge. To identify the dispatcher of an agent, it would be sufficient to add a digital signature to the agent composed of a hash of the agent's code and encrypted with the dispatcher's secret key. This however requires the existence of a PKI, compare section 4.4.1. A way to prevent agents from being copied unauthorizedly would be the use of watermarking-techniques.

To realise the anonymity of customer agents, one can allow customer agents not to carry a passport with them. In case they have to authenticate before committing to something, they can either go back home to fetch their passport or collect it from a trusted platform where they left it beforehand. These passports can be encrypted such that they can only be read by an authorised entity. Additionally, they can be protected by digital signatures against modifications. Nevertheless, for absolute anonymity, it is not sufficient that agents do not carry passports. Usually, agents carry the address and ID of their dispatcher such that they know where they have to go back to at the end of their task. From this information, it is possible to trace the agents back to where they come from.

This drawback can be countered by the introduction of a trusted pseudonym server. After an agent is dispatched, it is directly sent to this pseudonym server. There its original identity and its home address is deleted and replaced by a pseudonym. Malicious platforms can only identify the pseudonym, but not the real identity. Since the pseudonym server is trusted, it will protect the agent's identity. At the end of an agent's travel, the agent revisits the pseudonym server which recovers its home address and sends it back. Generally, anonymity and accountability are contrary issues. If we allow agents to stay anonymous, we will not be able to trace malicious actions back to specific agents. However with the pseudonym server, it is possible to integrate both issue. With the pseudonym, the original identity of agents is hidden. But if something illegal happens in a system caused by an agent, it can be traced back to the pseudonym. In case of justified evidence, the pseudonym server can recover the original identity which introduces accountability.

Platforms have to be able to authenticate themselves. It would be very harmful for an agent if it assumes to be on a trusted platform, but the platform it is on is actually not the platform it thinks it is on. A way to provide authentication would be via a public key infrastructure for platforms and a mechanism similar to digital signatures and certificates.

BB3 Data Protection
An agent is able to keep his collected data secret and to protect it from unauthorised modification. The data can only be accessed and modified with his permission.
Personal information, e.g. identity, payment information can be prevented from unauthorised disclosure, copying and modification.

Aspects of a Technical Realisation:
For general remarks on the protection of confidentiality and integrity of data in a mobile multiagent system see section 4.4.1. Assuming the technical model that customer agents roam on untrusted platforms, while matchmaker and merchant agents stay on their own trusted platform, only the customer agent needs special attention in this context. Problems to be addressed are confidentiality and integrity of its data. The data to be protected are offers an agent collects during its journey, the payment information, its identity potentially in form of an agent passport, the requests it wants to issue, contracts it has made and its code and control state.
Sensitive data can be encrypted on the dispatcher platform with the public key of the platform to which the agent is supposed to go. The data is safe during the transmission and on all platforms that the agent visits until it reaches the desired platform. For protecting data on the way back to the dispatcher the data can be encrypted with the dispatcher's public key as a last step before transmitting the agent to a next platform. As the data of the agent is encrypted, it is impossible for another platform read it unless it has a key for decryption. But a drawback is that the data cannot be used in this case. For integrity, the data can be signed digitally. This can be done with the platform's secret key which the agent is currently on to ensure that the data has not been modified on the way back. Similarly, the agent's data can be signed digitally by the dispatcher to detect modifications of data the agent carries from the beginning.
Another approach to protect data or at least to detect modifications are detection objects proposed by Meadows in [Mea97]. However, detection objects are always created for a particular application and cannot be used in all cases. An application for detection objects in comparison shopping, however, is to add a very low price for a product to the agent's memory before dispatching it. If the agent comes back with this price altered, one would know that the agent has been tampered with.
Yee [Yee97] proposes a protocol how to protect the partial results of agents after they leave a server and move to the next one. This is applicable to the protection of previously collected offers on malicious platforms. The results of Yee are extended and improved in [KAG98]. PRACs (partial result authentication codes) provide perfect forward integrity which means that all results that are collected before visiting a malicious server cannot be forged. An agent is dispatched with a sequence of keys, one per server visited. Before travelling to a next server, the partial results to be preserved are encrypted with one of these keys. After usage, the key is destroyed. So the result cannot be read, before the agent gets back home. An enhancement to this is a one-way-function that computes a new key from the last one. So the agent has an unlimited set of keys at his disposal. The number of servers that can be visited has not to be determined a priori. A further improvement are publicly verifiable PRACs. An agent is supplied with a list of secret keys and a corresponding list of verification predicates. Again it is also possible to defer the key and verification predicate generation to a one-way function. The verification predicates are publicly available, whereas the signature keys are secret and destroyed after use. So it can be verified whether a partial result is correct for computations that depend on these partial results. However, [Yee97] does not present techniques for the actual construction of the functions and predicates.
[KAG98] extends the mechanisms proposed by [Yee97] by a component to ensure the integrity of the collected offers. This approach assumes the existence of a public key infrastructure or at least the existence of a publicly available key of the agent's originator. Each shop signs its offer with its secret key such that the offer is unrepudiable and unforgeable. Then the offer may be encrypted with the agent owner's public key for confidentiality. [KAG98] proposes a

hash chain that links the offer of the previous shop with the identity of the next shop. This disables a shop from modifying its own offer later. The agent originator computes a hash of a dummy offer and the identity of the first shop to be visited as the anchor for the hash chain using a nounce. Afterwards, he encrypts it with his own public key and dispatches the agent. When a shop k is visited on the journey, the next offer is added probabilistically encrypted to the agents memory. A new hash is created from the previous offer of shop $k-1$ connected with the identity of the next shop $k+1$, encrypted and signed. This links the previous with the current offer such that the current offer cannot be modified without modifying the previous ones. The inclusion of the next shop's identity ensures that only this shop can add the next offer. This protocol can be further enhanced by encrypting the offer and the hash such that forward privacy is achieved. To avoid that a shop is able to exchange its offer at a later point in time, a nounce is included in this protocol which serves as the input of the computation of the next hash. The idea of publicly verifiability can additionally be applied in this approach.

When an agent comes to the stage where it commits itself and has to pay, it has to submit payment information. The requirement is that only the retailer that needs the information can obtain it. The best would be that the retailer gets only the information that the customer can pay the product, but not the concrete payment details. In that approach, as in the SET protocol (Secure Electronic Transaction), a trusted bank gets the payment details and notifies the retailer that the customer is able to pay. However, if this is not possible, there are alternatives to solve the problem. Firstly, the customer agent goes back home and fetches the payment information which is encrypted with the retailer's public key. Secondly, the customer agent jumps back to a trusted platform where it has left the payment information beforehand. Thirdly, the customer agent could carry the payment information with it in encrypted form. If it wants to submit the payment information to a retailer, it fetches the decryption key from a trusted platform and gives it to the retailer. If we include a bank or certification authority, the payment information could be certified by one of those organisations as being correct. Integrity can be protected by a digital signature of the agent owner or a central authority.

The program (its hash-value, respectively) of an agent can be digitally signed which allows to reveal unauthorised modifications of the program. However, the code cannot be encrypted if the platform has to work on it. Static parts of an agent's code that are not needed for a computation can be encrypted for confidentiality, before dispatching the agent. Additionally, they can be digitally signed for protection of integrity.

Any computation on private data has to be done on a trusted platform with respect to these data. For instance, in case of the shopping scenario the merchant platform may be trusted with respect to the offer of its merchant, but not with respect to other offers.

Platforms should protect agents from each other. Even on a trusted platform, there can be malicious agents that try to corrupt other 'good' agents on the same platform. Therefore, the platform has to guarantee that the agents can only communicate via messages and can not access the data or code of other agents directly. Furthermore, the communication between two agents should be impossible to monitor, disclose or alter by some other agent on the same trusted platform. Technically, this can be achieved my separating the name and address spaces for the single agents on the platform or by executing only one agent in a separate sandbox.

Although difficult, the passing on of confidential information can be prevented by technical means. Passing on of confidential information means that someone communicates information without permission of the owner of this information. In order to make that detectable, confidential data can be marked with the original sender and receiver using watermarking techniques. A watermark is linked to the content of the document such that it can not be separated from the document. Moreover, it can not be noticed at first sight, but there are specialised mechanisms for detection of a watermark. A watermark is robust to common modifications of a document and also to attacks. A violation of confidentiality is detected if someone receives watermarked information with a watermark that does not certify that he/she is the legal receiver. For preventing the illegal passing on of watermarked information a coordination medium can be used. The coordination medium detects watermarked data and checks whether these data is sent to the legal receiver. If there are inconsistencies, it blocks the message from going through the network. A coordination medium is a programmable communication device whose behaviour can be defined by means of suitable programming languages according to global system needs. There are several models for the implementation of coordination media for the Internet available such as Linda [ML94] or ReSpecT [DNO98]. An overview of the current research on watermarking techniques can be found in [Pfi00].

BB4 Agent Execution
It is guaranteed that an agent is executed as it was programmed by its owner. It is guaranteed that the code and the control state of an agent are not modified. It is guaranteed that an agent is transmitted as intended between two platforms. The integrity and the confidentiality during transmission is ensured. It is guaranteed that the agent is received.

Aspects of a Technical Realisation:
On a trusted platform the agent is executed as it was programmed by definition. However, on an untrusted platform it is very difficult to guarantee correct execution. There are several approaches in the literature that try to ensure that an agent is executed as it was programmed. The Code Mess Up technique [Hoh97] aims at the prevention of intended and directed modification of the agent's code. The basic idea in this approach is to modify the code such that it is impossible to find out the semantics of the code within a defined period of time. However, it is hard to determine what time is needed to solve the code mess up. Undirected modifications and denial of service attacks to the agent are still possible.

In [Vig98], Cryptographic traces are proposed to detect whether an agent was not executed as it was programmed. The execution of an agent on a remote host is logged by that host. At the end of the execution a hash of this execution trace is sent with the agent such that the agent owner can verify a correct execution or at least knows where the tampering must have happened. An assumption in this approach is that a malicious host nevertheless provides a correct trace of the execution which seems paradox. The data overhead produced enormous which restricts the applicability of this method.

Encrypted functions are the only approach to hide the program semantics from the platform. Encrypted functions are programs that are encrypted and yield encrypted results. These cannot be understood without having a decryption key. Therefore, directed modifications of the agent can be prevented with this method. A serious drawback however is that only certain kinds of functions can be expressed as encrypted functions [ST98].

Correct, confidential and integer transmission of agents over the network can only be achieved between trusted platforms. Digital signatures and encryption can enhance confidentiality and integrity. Transmission protocols, like handshaking, can ensure that an agents is transmitted correctly and that it is received. However, a malicious platform can refuse to transmit an agent or transmit it to the wrong place.

As we have seen, only trusted platforms completely ensure the correct execution of agents. The mechanisms for guaranteeing correct execution presented above have drawbacks with respect to their applicability. In addition to that, they only partially achieve the security goals. Therefore, it is interesting to ask whether it will be noticed by other agents or platforms if an agent is not executed or transmitted correctly. If other agents and platforms are able to discover that another platform is malicious, it is possible to put a kind of social pressure on platforms to execute agents correctly. In a society of agents, it could be common knowledge that platform X is malicious. Thus, noone will ever again visit it without taking the necessary precautions beforehand. A common reputation service could be introduced where such violations are reported to. A black list of platforms or agents can be maintained. A way to detect incorrect execution or transmission could be time stamps or expiry times for the code. If an agent does not get back after a defined time, it is assumed to have been killed. In order to identify where this might have happened, the route of the agent should be known to its owner. Alternatively, the agent could sent a message to its owner at any time it is leaving a host telling the owner where it is supposed to go to. If the agent does not notify its owner after some time, the owner knows that on the host the agent wanted to go to something malicious happened.

BB5 Protection of Platforms and of the Connecting Network
The access of agents to data and resources of platforms is restricted and protected.
The availability of platforms is protected. Denial of service attacks can be prevented.
The availability of the network is protected. Denial of service attacks to the network can be prevented. The confidentiality and integrity of the network communication is protected.

Aspects of a Technical Realisation:
There are several approaches that try to protect hosts from damage done by malicious agents. The data stored on hosts can be protected by sandboxing or safe interpreters [Moo98]. The agent is executed inside a padded cell or a sandbox and does not have unprotected access to the host. Each agent has its own address space. Other accesses to the outside are checked separately by a security manager. A link between the author and his agent can be established by the author singing it. Supposing a trust model, the trust in the author can be transferred to the code he has written. Proof-carrying code ensures that the agent works according to some security policy. The creator of the agent compiles a proof for the compliance to this policy which is verified at the agent's arrival [Nec97].
In order to protect the host's or platform's resources one can use access control mechanisms. For restricting the risk of denial of service attacks, one can use market-based resource control mechanisms that do not allow one agent to allocate infinitely many resources [BKR98]. However, currently there are no serious means available to prevent denial of service attacks. In general, it is very hard, if not impossible, to determine that a denial of service attack happens. A reason therefore it that it is hard to distinguish between an ordinary and a faked customer in a virtual mall.
The availability of the network can be protected using fault-tolerance mechanisms such that another path is chosen if some router breaks down. Alternatively, a second server can be used as a backup device. A possibility to reduce traffic at certain network nodes, possibly caused by a denial of service attack, is the use of load balancing techniques. The confidentiality and integrity of the network communication can be protected by encryption techniques and digital signatures, assuming that a PKI between the platforms exists, comparable to agent transmission in building block BB4. In order to make the network communication untraceable, mixes and anonymizer can be used.

BB6 Legal Liability
A customer and a provider agent are able to make a legally binding contract. The contract cannot be manipulated after it was made and cannot be denied by a signing party. If it is required, the contract can be kept confidential.
Provider agents can issue binding offers to customer agents. These offers cannot be manipulated and can be kept confidential, if it is necessary.

Aspects of a Technical Realisation:
Digital signatures in a public key infrastructure can be used to authenticate messages and to make a sender of a message legally liable. For details about digital signatures in a mobile multiagent system see section 4.4.1.
Legal liability of contracts can also be achieved by digital signatures. In our model, merchants are always on their own trusted platform. Therefore, they can sign the contract with the secret key at his disposal. The client, however, will only be able to sign the contract on a trusted platform where he can use his secret key. If the merchant's platform is not trusted by the customer, he has to move to a trusted platform, sign the contract and or submit it to the merchant. The digital signature including a hash of the contract also provides protection of integrity. The confidentiality of a contract can be established by encryption with public keys.
An offer issued by some merchant can be made legally binding by digitally signing it by the merchant. Since merchants only roam on their own trusted platforms, they can generate a hash of their offer and to sign it. The customer is given the offer with the attached signature.

The merchant's signature connects the merchants identity to the offer by means of the PKI. Additionally, the signature guarantees the integrity of the offer. Confidentiality can be achieved by encrypting the offer with the recipient's public key.

Another issue is the correctness of information that the customer agent is given by retailers. Here, we can distinguish between two phases. In the first, the customer only wants to set up an information collection and has no requirements with respect to correctness. In this case, nothing has to be done about the offers given by retailers. In the second phase, a customer wants to rely on the information and base his commitment based on it. Therefore, provider agents should issue binding offers to customers to vouch for the correctness of issued information.

BB7 Trusted Services
Agents can prove that the operate according to a quality of service policy.
Platforms can provide a certificate of their trustworthiness.

Aspects of a Technical Realisation:
Matchmaker and retailer agents can possess certificates that assert that they operate to a quality policy. The quality policy can for instance include that data is not resold or that offers are correct. The certificates are signed by some certification authority (CA). Matchmaker and retail agents can provide these certificates as a part of their authentication. So a customer contacting such an agent immediately knows by the provided certificate who the other agent is and to which quality of service policy it complies. The trust an agent will put into such a certificate always depends on the trust the agent has in the CA. If a merchant or a matchmaker is certified by an authority in which the client highly trusts he will also put high trust in the certified provider. If he does not trust the authority, he will be cautious what he tells the provider about himself and what he thinks of the provided service. Additionally, the client can take precautions before further interaction. The certificates need not necessarily be issued by some central CA. It is also possible to set up a web of trust as in PGP. Then a customer trusts a certificate if he trusts the certifier. By means of certificates, a trusted shop infrastructure or a web of certified matchmakers can be created.

Also platforms can be certified by some central certification authority to be benevolent, to execute agents correctly, and not to copy, modify or distribute the agent's data or code. Again, it will depend on the trust an agent has in the certification authority if it trusts the platform. If the agent accepts the certificate, the platform is trusted and the agent just goes there. But if the agent does not accept the certificate, it can take precautions before going to that platform. Precautions can include the encryption of sensitive data or the notification of its owner about what it is going to do such that the owner can detect a potentially malicious platform.

5 Conclusion and Future work

Agents and multiagent systems will play a major role in the further development of Internet-based applications like virtual marketplaces. However, these systems will not be successful until their specific security problems are solved. In this report, we examined comparison shopping as a case study for a virtual marketplace scenario and as an application domain for a mobile multiagent system with respect to its security issues. We established a detailed model of the scenario and analysed it regarding the interests of its participants and the possibilities of an attacker. From that analysis, we identified the overall security threats in this scenario and security objectives to counteract them. The security objectives were refined into building blocks, which the underlying multiagent system should provide, and further into features of agents and executing platforms. We discussed solutions for the implementation of these building blocks and pointed out under which assumptions it is possible to achieve the security goals.

Agent mobility increases the difficulty for the establishment of security in comparison to a system with static agents. While there are well-founded mechanisms to establish security in such systems, like encryption or digital signatures, those methods do not work in a mobile agent scenario. In some cases, we simply have to admit that a satisfactory solution is not possible because of technical and conceptual constraints. In this area, more work is necessary to come up with solutions for security issues connected to agent mobility. Currently, it seems that some problems cannot be solved at all. Consequently, the trade-off between advantages of mobile agents and their security issues has to be considered before choosing the agent paradigm.

References

[ACCK01] J. Algesheimer, C. Cachin, J. Camenisch, and G. Karjoth. Cryptographic security for mobile code. In *IEEE Symposium on Security and Privacy 2001*, pages 2–11. IEEE Computer Society, 2001.

[BG99] A.B. Brody and E.J. Gottsman. PocketBargainFinder: A handheld device for augmented commerce. In *Proc. of the International Symposiom on Handheld and Ubiquitous Computing (HUC'99)*, 1999.

[BKR98] J. Bredin, D. Kotz, and D. Rus. Market-based resource control for mobile agents. In *Proceedings of Autonomous Agents'98*, May 1998.

[CM96] A. Chavez and P. Maes. Kashbah: An agent marketplace for buying and selling goods. In *Proceedings of the First International Conference on the Practical Appication of Intelligent Agents and Multi-Agent Technology*, London, UK, April 1996.

[CMS01] A. Corradi, R. Montanari, and C. Stefanelli. Security of mobile agents on the internet. *Internet Research: Electronic Networking Applications and Policy*, 11(1):84–95, 2001.

[DEW96] R. B. Doorenbos, O. Etizioni, and D.S. Weld. A scalable comparison-shopping agent for the world-wide-web. Technical Report UW-CSE-96-03-01, Department of Computer Science and Engineering, University of Washington, 1996.

[DNO98] E. Denti, A. Natali, and A. Omicini. On the expressive power of a language for programmable coordination media. In *Proceedings of 1998 ACM Symposium on Applied Computing*, 1998.

[FGS96a] W. M. Farmer, J. D. Guttman, and V. Swarup. Security for mobile agents: Authentication and state appraisal. In *Proceedings of the 4th European Symposium on Research in Computer Security*, pages 118–130, Rome, Italy, 1996.

[FGS96b] W.M. Farmer, J.D. Guttman, and V. Swarup. Security for mobile agents: Issues and requirements. In *Proceedings of the National Systems Security Conference*, pages 591–597, 1996.

[GK99] A. R. Greenwald and J.O. Kephart. Shopbots and pricebots. In *Proc. of the International Joint Conference on Artificial Intelligence '99 (IJCAI'99)*, 1999.

[GM98] R. H. Guttmann and P. Maes. Agent-mediated integrative negotiation for retail electronic commerce. In *Lecture Notes in Artificial Intelligence*, volume 1571, pages 70–90, 1998.

[GMM98] R.H. Guttmann, A.G. Moukas, and P. Maes. Agent-mediated electronic commerce: A survey. *Knowledge Engineering Review Journal*, June 1998.

[Hoh97] F. Hohl. An approach to solve the problem of malicious hosts. Technical Report 1997/03, Universität Stuttgart, Germany, 1997.

[JCK$^+$00] J.Yang, J. Choi, J. Kim, H. Ham, and K. Lee. A more scalable comparison-shopping agent. In *Proceedings of Engineering of Intelligent Systems (EIS2000)*, pages 766–772, 2000.

[KAG98] G. Karjoth, N. Asokan, and C. Gülcü. Protecting of the computation results of free-roaming agents. In *Mobile Agents, Second International Workshop, MA'98, Stuttgart, Germany, Proceedings*, volume 1477 of *Lecture Notes in Computer Science*, September 1998.

[KG99] J. O. Kephart and A. R. Greenwald. Shopbot economics. In *Autonomous Agents '99*, 1999.

[Kru96] B. Krulwich. The BargainFinder agent: Comparison price shopping on the internet. In *Agents, Bots, and other Internet Beasties*, pages pp.257–263. SAMS.NET publishing (Division of Macmillan publishing), 1996.

[Mea97] C. Meadows. Detecting attacks on mobile agents. In *Proceedings of the 1997 Foundations for Secure Mobile Code Workshop*, pages 64–65, Monterey, CA, March 1997.

[ML94] N. Minsky and J. Leichter. Law-governed linda as a coordination model. In *Object-Based Models and Languages*, volume 924 of *Lecture Notes in Computer Science*, pages 125–145. Springer-Verlag, 1994.

[Moo98] J. T. Moore. Mobile code security techniques. Technical Report MS-CIS-98-28, University of Pennsylvania, 1998.

[MU01] P. M. Markopoulos and L. H. Ungar. Shopbots and pricebots in electronic service markets. In *Game Theory and Decision Theory in Agent-Based Systems.* Kluwer Academic Publishers, 2001. to appear.

[Nec97] George C. Necula. Proof-carrying code. In *Conference Record of POPL '97: The 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 106–119, Paris, France, January 1997.

[Pay] URL http://www.paybox.de.

[PDEW95] M. Perkowitz, R.B. Doorenbos, O. Etizioni, and D.S. Weld. Learning to unterstand information on the internet - an example-based approach. In *Proceedings of International Joint Conference on Artificial Intelligence '95 (IJCAI'95)*, 1995.

[Pfi00] A. Pfitzmann, editor. *Information Hiding '99*, volume 1768 of *Lecture Notes in Computer Science.* Springer-Verlag, 2000.

[Sch01] R. Scheepers. Supporting the online consumer decision process: Electronic commerce in a small australian retailer. In *Proceedings of The Twelfth Australasian Conference on Information Systems, Coffs Harbour, NSW, Australia*, December 5-7 2001.

[ST98] T. Sander and C Tschudin. Towards mobile cryptography. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, 1998. IEEE Computer Society Press.

[TMGW97] M. Tsvetovatyy, B. Mobasher, M. Gini, and Z. Wieckowski. Magma: An agent based virtual market place for electronic commerce. *Applied Artificial Intelligence*, 1997.

[TMP+97] I. Terpsidis, A. Moukas, B. Pergioudakis, G. Doukidis, and P. Maes. The potential of electronic commerce in re-engineering consumer-retail relationships through intelligent agents. In J.-Y. Roger, B. Stanford-Smith, and P. Kidd, editors, *Advances in Information Technologies: The Business Challenge.* IOS Press, 1997.

[Vig98] G. Vigna. Cryptographic traces for mobile agents. In G. Vigna, editor, *Mobile Agents and Security*, pages 137–153. Springer-Verlag, Heidelberg, Germany, 1998.

[Yee97] B.S. Yee. A sanctuary for mobile agents. In *Proceedings of the DARPA Workshop on foundations for secure mobile code, Monterey, USA*, March 1997.

[YLC00] J. Yang, E. Lee, and J. Choi. A shopping agent that automatically constructs wrappers for semi-structured online-vendors. *Lecture Notes in Computer Science*, 1983:368–373, 2000.

Secure Mobile Multiagent Systems In Virtual Marketplaces

A Case Study on Comparison Shopping

RR-02-02

Research Report

Ina Schaefer