On the Problem of Trust in Mobile Agent Systems*

Uwe G. Wilhelm and Sebastian Staamann
Laboratoire de Systèmes d'Exploitation

Levente Buttyán
Laboratoire de Telecommunications

Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland
e-mail: {Uwe.Wilhelm, Sebastian.Staamann, Levente.Buttyan}@epfl.ch

Abstract

Systems that support mobile agents are increasingly being used on the global Internet. Security concerns dealing with the protection of the execution environment from malicious agents are extensively being tackled. We concentrate on the reverse problem, namely how a mobile agent can be protected from malicious behaviour of the execution environment, which is largely ignored.

We will identify the problem of trust as the major issue in this context and describe a trusted and tamper-proof hardware that can be used to divide this problem among several principals, each of which has to be trusted with a special task. We show that the presented approach can be used to mitigate an important problem in the design of open systems.

1 Introduction

The notion of trust has long been recognized as being of paramount importance for the development of secure systems [8, 16, 28]. For instance, any conceivable system for authenticating users needs trusted functionality that holds the necessary authentication information (see e.g., [26, 22]). However, the meaning that is associated with trust is hardly ever clearly defined in these approaches and the reader is left with his intuition.

To illustrate the problem of trust, we will start with a description of the problems encountered when protecting the execution environment from malicious agents. Although we are primarily interested in the reverse problem of how a mobile agent can be protected from malicious behaviour of the execution environment, we prefer to illustrate the notion of trust with a problem that is more readily understood. Let us consider the example of signed code, such as Java applets or ActiveX controls. The issue of code signing has been identified as a major problem that is currently being tackled in both systems (Signed Java Archives [11] in Java and Authenticode [17] in ActiveX). The signature is intended to convey trust in the code. However, the only thing that it can convey is the trust in the identity of the signer of the code and the integrity of the received code (i.e., the assurance that the code was not tampered with after the signature has been applied). If the user also trusts the signer, which is an entirely different question, he can place some confidence in the code (e.g., if he has been satisfied with previous applications from this principal). If on the other hand the signer is an unknown company that tries to introduce a new application into the market, then the only thing that is gained by the signature is accountability. That is, if the code turns out to be malicious (or simply contains bugs that might damage the local system), the user knows who was responsible for the incurred damage – provided that the creator of the code did not successfully employ elaborate methods to evade this accountability.

In Java, there is another possibility to establish more trust in the mobile code prior to its execution by checking the bytecode for proper construction and by reducing the capabilities of mobile code through the definition of an explicit policy for access to local resources. The latter is enforced by the *Security Manager*, which is a trusted entity in the local system. Again, no concepts to protect the mobile agent from the execution environment exist. We will investigate, whether a similar mechanism for policy enforcement can be adapted to achieve the protection of mobile agents.

A more theoretical approach to better structure the notion of trust can be found in [2], where the authors identify *trust management* as an important problem to provide security in network services. Their approach is centered on local reasoning and requires the existence of trusted entities. A motivation why such entities are trusted, is not provided.

*Research supported by a grant from the EPFL ("Privacy" project).

In this paper we want to address the underlying question of how trust in a certain principal can be motivated based on social or technical reasoning and we will discuss our ideas in the context of mobile agent systems[1]. We believe that our findings can have important repercussions for the architecture of open systems for mobile agents, where everyone should be capable to enter as service provider. If, for instance, an unknown company designs a novel service that requires a user to surrender private information, the user might be quite reluctant to hand it over to a service provider he has never heard of. We will show how the trust that the user has in another principal can be transferred to the new service provider.

In the following Section 2, we will introduce our model for mobile agents and point out the problems related to trust within this model. Then, in Section 3, we will discuss the notion of trust and define its relation to policy, which will enable us to better assess the possible motivations for trust. In Section 4, we will introduce a piece of trusted hardware, the TPE, and a protocol, the CryPO protocol, that will allow us in Section 5 to show how these can be used to give a motivation for trust that is based on technical means. In Section 6 we discuss why this is a better way to approach the problem and what effects this has on the notion of open systems. Finally, Section 7 concludes the paper with a summary of the main contributions.

2 The mobile agent paradigm

The mobile agent paradigm (also referred to as mobile code, mobile computation, or program mobility) has been identified by many authors as a promising and innovative new approach to structure problems in distributed computing [4, 6, 12, 13, 24]. The paradigm is still under lively discussion and it has been shown in [14] that there is no single compelling reason to favour the mobile agent paradigm over classic client/server approaches. On the other hand, the same authors point out that the mobile agent paradigm provides interesting solutions to many real-life problems, for instance in the context of:

- mobile users, where agents are sent out from a mobile computer in order to accomplish a well-defined task on behalf of the user while he is disconnected from the communication network. Once the user reconnects, the agent returns and reports the result of the task or the problems it encountered.
- high-bandwidth interactions, where an agent is sent to a database server that holds a large amount of unstructured data to search for some specific information for the user.
- resident agents, which are stationary agents that take residence at some service provider and handle simple routine actions for their owner. An example for such routine actions is the communication management for a mobile user, where the agent decides how to handle an incoming communication request based on configured data (e.g., black and white lists), current information (e.g., time and date), and the user's situation (e.g., in a meeting or on holiday).

We are here neither interested in the underlying technology that is used to implement the mobile agent paradigm nor are we concerned with more high-level questions such as those identified in [4]. We only require a rather simple model for our discussion; thus, we identify the following major abstraction that we associate with mobile agents. A mobile agent consists of code and data (it may or may not contain the explicit execution state of the agent), which can be marshaled by the agent owner (in the following also referred to as service user) in a transportable form and subsequently sent to the agent executor (also referred to as service provider).

The agent can be confidentiality and integrity protected during transit to protect it against outside attackers through the use of cryptographic mechanisms. These mechanisms can also provide data origin authentication for the marshaled agent.

The agent executor will then eventually unmarshal the agent and instantiate it on its local *agent execution environment*. Here, the mobile agent can interact with other agents and try to accomplish the task it was given by its owner. Depending on the employed technology, the agent may continue to move to other service providers in order to accomplish its task and ultimately return to its owner or simply return the result of its task to the user in a message. We do not consider agents that visit several service providers as a special case, since in our solution this is only several instances of the same problem.

Our main interest lies in the fact that the relocation of a mobile agent raises the problem of how to protect the agent (especially the data contained in it) from undue manipulation by or undesired disclosure to the

[1]The term mobile agent has been charged with many different meanings. We use it here in the context of distributed systems (as opposed to its use in artificial intelligence) to refer to an object that is capable to move between different agent execution environments in order to accomplish some well-defined task on behalf of its owner.

agent executor, which is mainly a question of trust in the agent executor.

2.1 The problem with trust

There are many examples where an agent might need confidential information that should not be disclosed to the service provider, even though the agent needs the information to accomplish its task:

- an agent for electronic-commerce might hold data that could give a bargaining advantage to the service provider if it were known to him (e.g., a maximum price that the service user is willing to pay or the lowest QoS that he is willing to accept before inquiring at another service provider).

- a shopping agent (which we conceive as a special form of an agent for electronic-commerce that also integrates mechanisms for on-line payment) might hold information for several payment methods (e.g., different credit cards) of which the service provider should obtain at most one information in case the agent buys from it, but never more than one (in order to not disclose the payment information the service provider does not need). We will return to this example in Section 5.1.

- an agent in a personalized information system might contain private data from the customization by the user that is needed to find the right information; for instance, an agent searching for movies that are likely to interest its owner, might contain some very personal information about the user's special interests, which can not necessarily be inferred from simply observing a final choice. If the agent purchases a ticket for some movie, the agent executor should not be able to determine for what reason a particular movie was chosen.

- finally, an agent that merely searches for some particular financial information (such as stock quotes) might, depending on the owner of the agent, convey some very sensitive information (the mere request already conveys the interest in the information).

In a conventional agent system, when the agent owner (service user) sends a mobile agent to an agent executor (service provider) in order to use some service, the agent owner loses all control over the code and data of the agent. The agent executor can:

- reverse engineer the agent's code,
- analyze the agent's data,
- arbitrarily change the agent's code and data, or
- experiment with the agent (e.g., by feeding it with arbitrary data and resetting it to its initial state, in order to observe the agents reactions – using a combination of both, a provider might obtain the complete payment information from an agent even though it might not be able to analyze the code and data directly and it does not really want to provide the service to the agent).

This constellation puts the service provider in a much stronger position than the service user. The user simply has to trust the service provider not to use the methods described above to illicitly obtain confidential information from the agent, which it has to carry in order to use the service. There is no way for the user to control or even know about the behaviour of the service provider.

The reason for the imbalance between service provider and service user in the mobile agent model as compared to the client/server model is that in the former approach, the service user has no guarantees whatsoever concerning the execution of its agent. In the client/server approach, the service user relies on many assumptions that are so basic that one hardly ever thinks of them. Nevertheless, these guarantees allow to implement certain types of behaviour in the client part of the distributed application that can not be implemented in conventional agent systems (e.g., code will be executed at most once, code will be executed correctly, availability of a reasonably reliable time service, etc.). This advantage relies on the fact that the client implementation is under the physical control of the user. Based on these assumptions, a client can observe what is happening in the system and notice irregularities. Thus, it is able to react accordingly, for instance, to interrupt an ongoing transaction. Another possibility to take advantage of these assumptions from the client's point of view would be to log any irregularities at the client side so that they can be provided as evidence in the case of a dispute with some service provider.

We intend to create an environment for mobile agents that allows them to base their execution on similar assumptions, so that it becomes possible for a mobile agent to protect itself from a malicious service provider.

3 The notion of trust

We already mentioned the importance of trust for security in distributed systems and pointed out the lack of a clear definition of what is meant by the terms *trusted principal* or *trusted system*. In the following we will present our analysis of possible trust relations with other principals.

A reason for the lack of a clear definition of trust could be that trust is more a social than a technical issue and consequently quite difficult to tackle entirely in a technical approach. The major problem stems from the fact that the notion of trust mixes the goals of a principal with its behaviour to achieve these goals. In order to trust some principal it is usually necessary to concur with or at least approve of its goals (which are not always clearly stated) and to believe that it will behave accordingly. In our definition of trust, we will try to clearly separate these two issues by gathering the goals of a principal in a policy, which is a set of rules that constrains the behaviour of this principal for all conceivable situations. This policy has to be written down and made available to all other principals that interact with the issuer of the policy. Then, we define *trust in another principal* as the belief that it will adhere to its published policy.

The question of whether a certain principal can be trusted now consists of (a) checking its published policy in order to decide if it is acceptable and (b) to establish a motivation for the belief that it will adhere to its published policy. The former is quite difficult but can be supported by a formal specification of the security policy (similar to the approach in [20]). The latter, however, is a problem that is quite difficult to formalize. In order to better grasp the possible motivations for such a belief, we identified the following four foundations for trust:

- blind trust
- trust based on (a good) reputation
- trust based on control and punishment
- trust based on policy enforcement

Blind trust indicates that there is no particular motivation to believe that a principal will adhere to its published policy other than its own assertion. It is obviously the weakest foundation for trust and not recommended for any important transaction.

Trust based on (a good) reputation stems from the fact that the principal in question is well known and has very little to gain through a violation of its own policy but a lot to lose in case a policy violation is discovered. This loss is supposed to transpire from the lost revenue due to customers taking their business to another principal. Reputation is an asset that is very expensive to build up and that is invaluable for any company. Thus, a principal would not risk to lose its good reputation for a small gain and will consequently adhere to its policy.

Trust based on control and punishment is quite similar to trust based on reputation (even though it actually means that we do not trust the principal at all, but rather the underlying technical and legal framework to ensure the principal's proper behaviour). The idea is to artificially introduce the same tradeoff described above by enacting appropriate laws. Here, the short term gain that might be achieved through a policy violation is supposed to be negated by appropriate punishment.

The main problem of the last two approaches is the difficulty to discover a policy violation in the first place. If the risk for such a discovery becomes so small that it can almost be neglected by an offender, there is much less motivation for a principal to adhere to its published policy. In such a situation it is necessary to improve the controls to make a discovery sufficiently probable. Further problems are the enforcement of laws, which can be quite expensive for a single person, but also the different perceptions of punishment – a person who has not much to lose might readily risk some years of imprisonment for the possibility of a relatively large gain.

Another problem in the approach stems from the fact that many abuses of confidential information are not necessarily conducted for the purposes of the company that holds this information, but rather by malicious insiders of such a company, who do it for strictly personal reasons or financial benefits [23, 25]. Such abuses are even more difficult to discover (there are less people involved) and to punish (it has to be decided if only the employee for malicious behaviour, only the company for negligence, or both have to be pursued).

The problem to reliably discover a policy violation could be resolved by requiring a very high degree of transparency. However, this is difficult to achieve and it is quite likely that even trustworthy principals with a very good reputation might not accept it. We therefore assume that complete transparency is not a very useful tool for supervision. A better approach would be to designate specialized companies that execute frequent in-depth controls of the conduct of companies.

Finally, neither of the two approaches can prevent malicious behaviour, but they only try to compensate for it after it has been discovered. For many situations in real-life, where an offence might have an irreparable effect (e.g., the collapse of the Barings Bank, where the Bank officials trusted their trader [10]) or where a proper functioning of the system is absolutely essential, this guarantee might not be strong enough.

We would like to remark that most of these problems are also present in our every-day life and therefore quite well understood. However, the question stands if we can do better than that.

Trust based on policy enforcement would obviously be the best foundation for trust, since this means that a principal simply *can not* violate its published policy (actually, this implies that we do not have to trust the principal anymore) and we can solely rely on this policy to verify that the principal's behaviour will be acceptable. The behaviour of the principal becomes completely transparent as far as it is constrained by its policy without the need to actually supervise any particular action. If the policy prescribes a particular action for some event and if the policy is enforced then it is guaranteed that the action will take place. In reality, things are probably not that easy. A policy might be composed of many different rules, of which only a subset is enforceable, while others can at best be controlled; for these we still have to rely on other foundations for trust.

There is no possibility to enforce rules within a policy without relying on some piece of trusted and tamper-proof hardware [6]. In the following section, we will describe such a piece of hardware and the requirements that have to be met so that it can be used to enforce certain rules of a policy.

4 Tamper-proof hardware and the CryPO protocol

We will first present the execution environment that we rely on and then describe the CryPO (cryptographically protected objects[2]) protocol that uses it. Figure 1 gives an overview of the principals in the system.

The TPE manufacturer produces the TPEs, which can be bought by any agent executor (service provider). An agent owner (service user) has to trust the TPE manufacturer to do this properly (see Section 6). The broker is basically a directory service to find the other principals.

4.1 Notation

The described approach relies on public key cryptography [7] (such as RSA [19]). A detailed description of cryptography and the corresponding notations is not within the scope of this presentation, for information on this topic see, for instance [3, 21]. The notation we will use is as follows.

[2]We have originally chosen the term object since it is more general than the term agent.

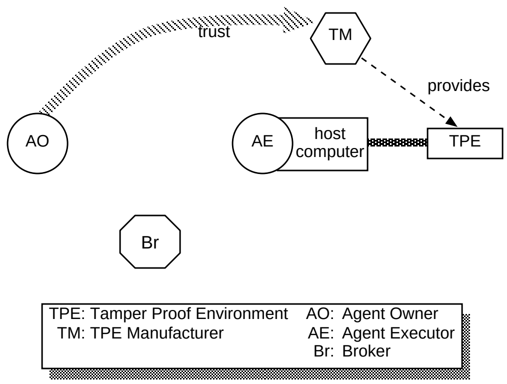

Figure 1: Overview of the Principals in the CryPO protocol

A principal P has a pair of keys (K_P, K_P^{-1}) where K_P is P's public key and K_P^{-1} its private key. Given these keys and the corresponding algorithm, it is possible to encrypt a message m, denoted $\{m\}_{K_P}$, so that only P can decrypt it with its private key. A signed message, including a digital signature on the message m, generated by P is denoted $\{m\}_{S_P}$.

In the following we assume the usage of optimization schemes such as encrypting a large message with a symmetric session key, which in turn is encrypted using public key cryptography and prepended to the message as well as the use of hash algorithms to reduce the amount of data that has to be signed. However, for ease of presentation, we will not make this explicit.

4.2 The execution environment

As we have noted above, there is no way to enforce any particular behaviour from another principal without a piece of trusted and tamper-proof hardware. We will discuss the problem of trust in the tamper-proof hardware in Section 6. The concept of tamper-proofedness usually applies to a well-defined module, sometimes called black-box, that executes a given task. The outside environment cannot interfere with the task of this module other than through a restricted interface that is under the complete control of the tamper-proof module (see Section 4.4). We will call this device *tamper-proof environment* (TPE). The TPE provides a full execution environment for agents, which can not be inspected or tampered with. Any agent residing on the TPE is thus protected by the TPE both from disclosure and manipulation.

The TPE is a complete computer that consists of a CPU, RAM, ROM, and non-volatile storage (e.g. harddisk or flash RAM). It runs a virtual machine (VM) that serves as execution environment for agents and an operating system that provides the external interface to the TPE and controls the VM (e.g., protection of agents from each other). Furthermore, the TPE contains a private key K_{TPE}^{-1} that is known to no principal other than the TPE – also the physical owner of the TPE has no information concerning this private key. This can be achieved by generating the private key on the TPE[3]. Using this approach, the key is never available outside of the TPE and, thus, protected by the operating system and the tamper-proofedness of the TPE. The secrecy of the private key is a crucial requirement for the usage or the TPE to enforce a particular behaviour.

The TPE is connected to a host computer that is under the control of the TPE owner. This host computer can access the TPE exclusively through a well defined interface that allows, for instance, the following operations on the TPE:

- upload, migrate, or remove agents;
- facilitate interactions between host and agent or between agents on the TPE;
- verify certain properties of the TPE (such as which agents are currently executing).

Due to its implementation as a tamper-proof module and the restricted access via the operating system, it is impossible to directly access the information that is contained on the TPE.

This property is ensured and guaranteed by the TPE manufacturer (*TM*), which also provides the agent executor *AE* with a certificate (signed by TM). The certificate contains information about the TPE, such as its manufacturer, its type, the guarantees provided, and its public key. The agent owner *AO* has to trust the TM (see Section 6) that the TPE actually does provide the protection that is claimed in the certificate.

4.3 CryPO protocol

The CryPO (cryptographically protected objects) protocol transfers agents exclusively in encrypted form over the network to a TPE. Therefore, it is impossible for anyone who does not know the proper key to obtain the code or data of such a protected agent.

The protocol is divided into two distinct phases. The first phase consists of an initialization, which has to be executed once before the actual execution of the protocol. The second phase is concerned with the usage of the TPE and the actual transfer of the agent. The protocol is based on the interactions given in Figures 2 and 3.

[3] Other, more sophisticated approaches to create the pair of keys could be envisaged, which could also incorporate key recovery mechanisms (e.g., escrowed key shares).

4.3.1 Initialization: In the initialization phase, the participants establish the trust relations that are associated with the different keys:

- the TM publishes its certification key K_{TM}.
- the TM sends the certificate $Cert_{TPE} = \{K_{TPE}\}_{S_{TM}}$ to the AE.
- the AE registers its reference[4] with one or several brokers.

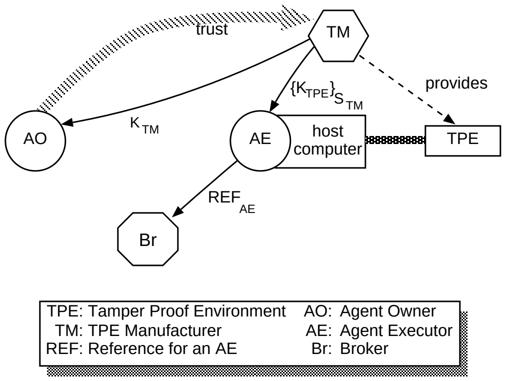

Figure 2: Initialization of the CryPO protocol

4.3.2 TPE usage: After the participants have finished the initialization, they can execute the usage part of the CryPO protocol:

- The AO queries the broker for the reference to the AE with which it wants to interact (or it already holds this reference from a previous interaction).
- The AO verifies the certificate $Cert_{TPE}$ to check the manufacturer and the type of the TPE, in order to decide if it satisfies the security requirements of the AO. If it is not satisfied with these checks, it will abort the protocol.
- The AO sends the agent encrypted with the public key of the TPE, $\{A\}_{K_{TPE}}$ to the AE.

[4] A reference to an AE consists of its physical address in the network and the certificate $Cert_{TPE}$ for its TPE. The broker can also verify that the AE actually controls the corresponding TPE by executing a challenge-response protocol with the TPE via the AE.

- The AE cannot decrypt $\{A\}_{K_{TPE}}$ nor can it do anything other than upload the agent to its TPE.

- The TPE decrypts $\{A\}_{K_{TPE}}$ using its private key K_{TPE}^{-1} and obtains the executable agent A, which will eventually be started and can then interact with the local environment of the AE or other agents on the TPE.

- The agent can, after it has finished its task, send a message back to its owner or request its migration back to its owner or to another AE to which it holds a reference.

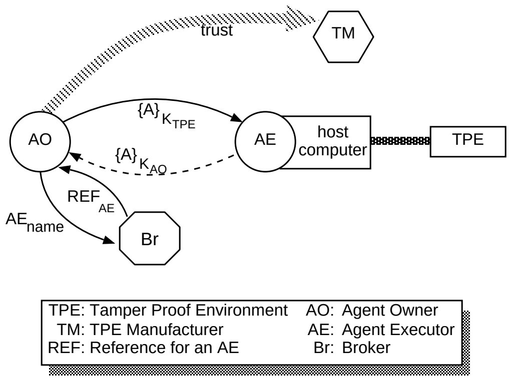

Figure 3: Usage of the CryPO protocol

The obvious problem of protecting the TPE from malicious agents is independent of the described approach and has to be tackled with additional mechanisms, such as code signing. The problem of protecting the TPE from tampered agents can easily be solved by concatenating the agent with a well known bit-pattern (magic number, *MN*) before encrypting it $\{A, MN\}_{K_{TPE}}$. The TPE simply has to verify the correct *MN* before starting the agent.

4.4 Notes on feasibility

The actual construction of a tamper-proof module in the real world is difficult; nevertheless, there are many applications that rely on them (e.g., payphones, debit cards, or SIM cards for GSM). Given sufficient time and resources, it becomes very probable that an attacker can violate the protection of such a module [1].

We believe that the actual realization of the presented TPE with reasonably strong guarantees in real-world settings is also quite difficult, but nonetheless feasible. Especially, since we only require the detection of tampering for most envisioned applications.

We imagine the TPE as a regular computer with a special operating system. It is physically protected with a special hardware that can effectively be sealed to detect tampering, is under continuous video surveillance similar to the systems used to supervise ATMs, and is subject to challenge inspections by the TM or an independent appraisal and inspection organization.

As explained in [1], such an installation is conceivable and can even resist massive attacks. A thorough analysis of the remaining risks has to be undertaken, but this is not within the scope of this presentation.

4.5 Related work

The idea to use trusted hardware to ensure a certain behaviour of a system, that was presented in this section, has previously been explored by several authors. In [15], Herzberg and Pinter describe a device that can be used to protect software against piracy. In [5], Chaum and Pedersen describe an architecture of a wallet, which carries a database with personal information that allows to protect the data in this database from unauthorized access (including the owner of the wallet). The described system incorporates trusted hardware (called observer) which is comparable to the one presented above but it is explored in a very different setting. A more recent approach by Yee and Tygar [27] has many commonalities with the one presented here. However, the authors are rather interested in the classical security aspects of how to ensure the secure functioning of the system, as opposed to our interest in privacy. It might be possible to define a device that can be used for both approaches. This will be a subject of our future work.

5 Usage of the TPE to enforce policies

The CryPO protocol and the TPE described above guarantee the integrity of the execution environment to the AO and protect the code and data of an agent against manipulation and disclosure, both in transit and during execution. This guarantee is based on the trust relation between the AO and the TM, where the AO trusts the TM to properly manufacture its TPEs and to control them regularly (if necessary) so that the claimed guarantees hold. The certificate enables the AO to ensure that it really deals with a TPE from a certain manufacturer.

We will now investigate how these guarantees can be used to formulate rules of a policy, that can effectively be enforced by a TPE. If the enforced policy rules provide sufficient protection for a given agent, then the user does not need to trust the AE, but it suffices to trust the TM. The problem of why the agent owner should trust the TM is dealt with in the following section.

A proof or simply a convincing argument that the enforced rules do provide a sufficient protection depends not only on the rules and the task of the agent, but also on what a particular AO considers to be a sufficient protection. Thus, the answer can not be given in its full generality but has to be decided on a case by case basis. A formalization of the presentation of the provided guarantees and the required protection, together with a mechanism to show that the provided guarantees do provide the required protection is an interesting question for further research.

5.1 Protecting the shopping agent

We will explore this question considering the shopping agent from Section 2.1. For the sake of this discussion we assume that the TPE of the service provider enforces a well defined set of rules, detailed in its policy. We will subsequently show that this set of rules is sufficient to provide the desired protection for the shopping agent. The rules enforced by the TPE are the following:

a) the code of an agent will never be disclosed by the TPE.

b) the data of an agent can exclusively be accessed and manipulated through the interface of the agent. If the agent does not provide methods to directly access a particular data item, its value can at most be inferred from the responses to other method invocations.

c) the TPE guarantees that any invocation of the agent's methods will be executed exactly according to the code in the agent.

d) the TPE will allow agents to leave the TPE only in encrypted form and it will provide the certificate of the designated receiver to the agent, which can decide whether it wants to be transferred to this receiver or not. The TPE will honour the agents decision.

e) the TPE provides an internal clock with reasonable accuracy (on the order of several seconds). If the TPE did not succeed to establish this time, it will inform the agent about this problem.

f) the TPE provides a small amount of non-volatile storage for a fixed period of time even to a removed agent.

Based on these guarantees, it is now possible to implement an agent that will under no possible circumstances reveal more than one payment information. During its execution, the agent will interact with the service provider in order to determine whether it will purchase the item or the service requested by the user from the service provider. If the agent decides to commit the purchase, it will ask the service provider about the supported payment methods. The agent will then choose the preferred method of the user among those supported by the service provider and finalize the purchase by the exchange of payment information. In order to implement the desired protection, we use the guarantees as detailed below.

According to guarantees a) and b), the code and the data of the agent are protected against tampering and disclosure and according to c), it will be executed exactly as it was programmed. Therefore, we can rely on the programmed methods to not disclose more than a single payment information. Since the agent will not accept to be sent to any other TPE in the network and since the TPE guarantees in d) that it will honour this decision, the only TPE that we have to consider, is the one that provides the above guarantees. These are not sufficient to implement the desired protection for the shopping agent since we have to consider the following attack.

The agent executor could store the originally received, encrypted agent before uploading it to its TPE and obtain the first payment information by normally interacting with the agent. In a second step the agent executor could then replace the agent on the TPE with the stored version and conduct another interaction with the agent, this time requesting the second payment information.

To counter this attack, the agent has to ensure that it will be executed at most once on the TPE in question[5] or rather that any subsequent execution will produce the same result. To implement this, we will take advantage of the guarantees e) and f). We provide the agent with a limited lifetime (on the order of a few days — if it has not accomplished its task by then, there is probably a major problem that the agent can not solve). Upon its arrival the agent verifies that the TPE has a sufficiently accurate clock, it inquires whether the TPE considers this time to be correct, and then it checks that the current time is still within its attributed lifetime. If either of these requests results in a negative answer, the agent termi-

[5] If reliability is an issue in the application, the agent can use checkpointing and recovery mechanisms to ensure that a failure of the TPE will not prevent the agent from accomplishing its task.

nates. This limited lifetime establishes a time T after which the agent can never be executed any more. In order to ensure the at most once execution before time T, the agent verifies that its identity is not already stored in the non-volatile memory of the TPE, which provides the proof that the agent has not previously been instantiated on this TPE. Finally, it stores its identity together with its lifetime in the non-volatile memory of the TPE, in order to prevent further instantiations of the same agent. As an alternative, the agent could store the chosen payment information in the non-volatile memory of the TPE and allow several instantiations of the same agent. Each of these instantiations, however, will always provide the same payment information.

The lifetime associated with the stored information can be used by the TPE to remove entries that are no longer needed (from agents whose lifetime has expired) to prevent a memory overflow in the TPE. If an agent tries to allocate too much memory, the TPE can refuse to provide the amount requested by the agent. The agent can still be executed at a later time when more memory is available, provided that its lifetime has not expired.

Since the service provider has no possibility to directly access the payment information within the agent (b), since the agent will be executed correctly (c), since the agent is programmed to only release a single payment information during any particular execution, and since the agent can be executed at most once on a particular TPE (e) and (f), we can conclude that the enforcement of the above guarantees allows us to create an agent that holds information for several payment methods, but that will disclose at most one to the selected service provider.

6 Trust in the TPE manufacturer

We have just introduced the mechanism, with which an agent can take advantage of the guarantees enforced by a TPE. However, as we have mentioned above, in order for a user to trust in these guarantees, it is necessary that he also trusts the TPE manufacturer to properly design, implement, and produce its TPEs. Since there is no way (to the knowledge of the authors) to enforce a correct behaviour of the TPE manufacturer, it seems that the presented approach simply replaces one required trust relationship with another one. This is a correct observation from a theoretical point of view. Nevertheless, we believe that this replacement of trust in an arbitrary service provider with trust in a TPE manufacturer has several more subtle implications. We will briefly discuss the following advantages that we identified:

- better understanding of security and privacy problems
- centralized control
- resources to build reputation
- separation of concern

The TPE manufacturer is a specialized service provider, which primarily deals in the field of the provision of security devices. Therefore it has a better understanding of security and privacy problems, which makes it a much more capable entity to ensure this service since it is more aware of the potential problems and pitfalls.

We assume that there will be relatively few TPE manufacturers (on the order of several hundreds) compared to the number of possible operators of the TPE (on the order of several millions). This makes the control of their behaviour much easier for expert appraisal organizations. Also, it is quite conceivable that a TPE manufacturer might invite external experts to control its internal operation, in order to obtain a better position in the market (similar to the approach for quality assurance in the ISO-9000 or the approach taken by Intermind[6]).

The production of TPEs is considered to be a difficult task (see Section 4.4). Therefore, we assume that it will be undertaken by major corporations, which have the necessary resources to build a good reputation and an incentive to protect this reputation, which allows us to rely on good reputation as foundation for trust in the TPE manufacturer.

The TPE manufacturer that is responsible for the enforcement of the proper policy rules on the TPE, has nothing to gain by not accomplishing its task. Since the TPE will be operated independent from the TPE manufacturer by a completely different principal and the TPE manufacturer has no means to access the data that is processed on the TPE (no physical connection), there is no possibility for the TPE manufacturer to draw a direct benefit[7] from a TPE that does not properly enforce its policy.

We assume that the above arguments of high expertise, effective controllability, good reputation, and lack

[6]Intermind was evaluated by a Big 6 accountants firm to verify the implementation of its privacy policy. For further information see press release from June 6, 1997 on `http://www.intermind.com/`.

[7]There is the possibility that a TPE operator bribes a TPE manufacturer to provide an incorrect TPE. We assume that such a behaviour is a severe offence that is subject to criminal investigation and not within the scope of this discussion.

of incentive are sound reasons to trust a TPE manufacturer to build reliable and powerful TPEs. The main advantage of the approach lies in the possibility to leverage this trust in the TPE manufacturer onto a completely different principal in the role of a service provider, which

- does not have the proper expertise to ensure a secure operation of its hardware and to guarantee the protection of the processed data.
- is quite difficult to control, due to the sheer number of such service providers.
- has no particular reputation (and therefore none to lose).
- might have short term goals that (in its point of view) justify a policy violation.

With the presented approach, such a service provider can easily define the policy rules that it would like its TPE to enforce (by selecting from the options offered by the TPE manufacturer) and buy the appropriate TPE from a reputable TPE manufacturer. The service provider can then immediately benefit from the trust that users have in the TPE manufacturer of its TPE to convince them that it will not abuse any confidential data sent by the users. With this, the approach favours openness in the sense that it becomes much easier for a new service provider to establish itself in the market.

6.1 A note on open systems

In the open systems philosophy, any principal can possibly become a provider of services. The technical problems associated with this approach are currently being tackled, for instance in the context of CORBA [18] and TINA [9]. However, if a client wants to use the services of some provider he has never heard of and whom he does not trust and if the provider needs some confidential information from the client, in order to provide the service, there is a dilemma. Trust based on reputation is difficult and expensive to build (and thus hardly an option for a small company) and trust based on control and punishment might not be sufficient for the client especially if the provider is located in a different country with an unknown law-system.

The client will not simply surrender the information since he has no trust in this provider to properly handle the confidential information and the provider can not build up its reputation, since it needs some users that take the risk of blindly trusting him. In this context the solution described above allows for much simpler bootstrapping of trust. The provider simply has to obtain the trusted hardware from some reputable provider and advertise this with his service offer. If the rules in its policy that can be enforced by the trusted hardware are sufficient to convince a client that his confidential information is sufficiently protected then the two can start to do business with each other.

7 Conclusion

In this paper we have discussed the notion of trust in the context of mobile agent systems and introduced a structuring for this problem domain. Starting from this structure, we have proposed an approach that relies on trusted and tamper-proof hardware, which allows to prevent malicious behaviour rather than correct it. We believe this to be the better form of protection for personal data. We have shown how the approach can be used to effectively protect some specific information contained in an agent using a simple example in a real-world setting. Finally, we identified the positive implications that the presented approach can have on the construction of open mobile agent systems, where any principal can become a service provider and receive mobile agents.

In real-life, there are limitations to the approach. Given sufficient time and resources, a TPE operator might succeed in breaking the system and it would thus be possible for him to violate even those parts of the policy that should be enforced by the TPE. Our goal is to make this approach so costly that it would negate a possible gain (there may be many different implementations of TPEs that provide different security guarantees). A non-repudiable proof for a policy violation of an enforced policy or of an attempted or successful breaking of a TPE might be punished much more severely than a mere policy violation since it proves a much larger determination to commit a criminal offence.

References

[1] R. Anderson and M. Kuhn. Tamper resistance — a cautionary note. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, pages 1–11, Oakland, California, November 1996.

[2] M. Blaze, J. Feigenbaum, and J. Lacy. Distributed trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173, May 1996.

[3] G. Brassard. *Modern Cryptology – A Tutorial*, volume 325 of *Lecture Notes in Computer Science*. Springer Verlag, 1988.

[4] A. Carzaniga, G. P. Picco, and G. Vigna. Designing distributed applications with mobile code paradigms. In R.Taylor, editor, *Proceedings of the 19th International Conference on Software Engineering (ICSE'97)*, pages 22–32. ACM Press, 1997.

[5] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Advances in Cryptology: Crypto'92*, volume 740 of *Lecture Notes on Computer Science*, pages 89–105. Springer, 1992.

[6] D. M. Chess, B. Grosof, C. G. Harrison, D. Levine, C. Parris, and G. Tsudik. Itinerant agents for mobile computing. *IEEE Personal Communications*, 2(3):34–49, October 1995.

[7] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22(6):644–654, 1976.

[8] DoD. Trusted Computer System Evaluation Criteria (TCSEC). Technical Report DoD 5200.28-STD, Department of Defense, December 1985.

[9] F. Dupuy, G. Nilsson, and Y. Inoue. The TINA Consortium: Toward networking telecommunications information services. *IEEE Communications Magazine*, November 1995.

[10] The Economist. The collapse of Barings, March 4 1995.

[11] J. S. Fritzinger and M. Mueller. Java security. White paper, Sun Microsystems, Inc., 1996.

[12] J. Gosling and H. McGilton. The java language environment. White paper, Sun Microsystems, Inc., 1996.

[13] R.S. Gray. Agent Tcl: A transportable agent system. In *Proceedings of the CIKM Workshop on Intelligent Information Agents*, Baltimore, MD, December 1995.

[14] C. G. Harrison, D. M. Chess, and A. Kershenbaum. Mobile agents: Are they a good idea? In J. Vitek and C. Tschudin, editors, *Mobile Object Systems: Towards the Programmable Internet*, volume 1222 of *Lecture Notes on Computer Science*, pages 25–47. Springer, 1997. Also available as IBM Technical Report RC 19887.

[15] A. Herzberg and S. S. Pinter. Public protection of software. In *Advances in Cryptology: CRYPTO'85*, pages 158–179, Santa Barbara, California, August 1985.

[16] ITU. *ITU-T Recommendation X.509: The Directory – Authentication Framework*. International Telecommunication Union, 1993.

[17] P. Johns. Signing and marking ActiveX controls. *Developer Network News*, November 1996.

[18] OMG. *CORBA: The Common Object Request Broker Architecture (Revision 2.0)*. Object Management Group, July 1995.

[19] RSA Data Security, Inc. *PKCS #1: RSA Encryption Standard*. RSA Data Security, Inc., November 1993.

[20] R. A. Rueppel. A formal approach to security architectures. In *EuroCrypt*, pages 387–398, Brighton, England, 1991.

[21] B. Schneier. *Applied cryptography*. Wiley, New York, 1994.

[22] J. G. Steiner, C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the USENIX Winter 1988 Technical Conference*, pages 191–202. USENIX Association, Berkeley, USA, February 1988.

[23] New York Times. U.S. workers stole data on 11,000, agency says, April 6, 1996.

[24] J. E. White. Telescript technology: The foundation for the electronic market place. White paper, General Magic, Inc., 1994.

[25] I. S. Winkler. The non-technical threat to computing systems. *Computing Systems, USENIX Association*, 9(1):3–14, Winter 1996.

[26] T. Y. C. Woo and S. S. Lam. Authentication for distributed systems. *IEEE Computer*, 25(1):39–52, January 1992.

[27] Bennet Yee and Doug Tygar. Secure coprocessors in electronic commerce applications. In *Proceedings of The First USENIX Workshop on Electronic Commerce*, New York, New York, July 1995.

[28] P. Zimmermann. *PGP User's Guide*. MIT Press, Cambridge, 1994.