A TECHNICAL APPROACH TO PRIVACY BASED ON MOBILE AGENTS PROTECTED BY TAMPER-RESISTANT HARDWARE

THÈSE N^{O} 1961 (1999)

PRÉSENTÉE AU DÉPARTEMENT D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Uwe Georg WILHELM

Diplom-Informatiker, Universität Kaiserslautern, Allemagne
de nationalité allemande

acceptée sur proposition du jury:

Prof. A. Schiper, directeur de thèse
Prof. J.-P. Hubaux, rapporteur
Prof. F. Mattern, rapporteur
Dr M. Reiter, rapporteur

Lausanne, EPFL
1999

Abstract

We address the problem of protecting the privacy of individuals in the information society. Our goal is to devise technical means that allow users to actively participate in the management and use of information related to them.

The advent of the information society creates serious challenges for the privacy of individuals. Due to the drastically improving communication infrastructure, ever larger amounts of ever more precise information become available. The problem with the free availability of this information is not only the risk that the information can be abused by powerful institutions, but also that this can lead to an unconfined mutual surveillance of individuals, which can have adverse effects on society as a whole.

We argue that individuals should be empowered to define for themselves the level of privacy they are comfortable with. This can be achieved by notifying them whenever information on them is created, accessed, or modified and by giving them some control over the use of this information. The notification informs individuals who is using what information on them and allows to detect possible problems with this use. The control allows individuals to resolve most (or at least some) of these problems. Obviously this requires that the individuals can trust the users of information to properly implement these notifications and to offer an effective control. We analyze the concept of trust more closely and distinguish between the optimistic and the pessimistic approach to trust, which can both provide the foundation for the protection of privacy. The former is based on the classical concepts of control and sanctions, while the latter tries to prevent malicious behaviour.

We choose to pursue the pessimistic approach and investigate in technical means that can be used for this purpose. A promising technology is the mobile agent paradigm, which is a new approach to structure distributed applications. Its main idea is to move both the code and the state of an object to another principal for remote execution. This indicates that the mobile agent paradigm also embraces the object-oriented programming paradigm, which allows us to encapsulate a data item and to specify an access control policy on it. Since the mobile agent is physically moved to a remote location that is under the control of a different principal, it needs to be protected from this principal who is responsible for its execution. This problem constitutes the major difficulty for using the mobile agent paradigm for privacy protection and is explored in great detail. Based on the discussion in the relevant literature, we decide on an approach that relies on a trusted and tamper-resistant hardware device, which is developed on a conceptual level.

The approach is further explored in the context of the mobile agent paradigm, where it allows us to realize more elaborate protection goals that may be desirable for the owner of the mobile agent. These are developed in the form of conducts, which regroup the goal, the requirements, as well as a specification of the necessary collaboration to achieve this goal.

Finally, we return to the original problem and describe how the presented technology can

be used to improve the protection of privacy. This results in a rather complex framework, in which information on individuals cannot be used freely, but where this use is constrained by the level of privacy desired by the subject of the information. The major problem of this framework is the increased complexity that individuals have to deal with. This problem is addressed with an additional level of indirection that attempts to confine the complexity and to delegate it to trusted experts.

We believe that this approach, despite its complexity, is a viable means to address the urgent problems of privacy protection, which do not lend themselves to simple solutions.

Résumé

Nous nous intéressons au problème de la protection de la vie privée des individus dans la société d'information. Notre objectif est de concevoir des moyens techniques qui permettent aux utilisateurs de participer activement à la gestion ainsi qu'à l'utilisation de l'information qui se réfère à eux.

L'arrivée de la société d'information pose d'importants défis pour la vie privée des individus. Les progrès considérables des systèmes de communication permettent d'accéder à des quantités d'information toujours plus grandes et toujours plus précises. Le problème du libre accès à cette information n'est pas uniquement le risque qu'elle puisse être utilisée de manière abusive par des institutions puissantes, mais également que cela puisse mener à la surveillance mutuelle et sans bornes des individus, ce qui peut provoquer des effets défavorables sur la société dans son ensemble.

Nous soutenons que les individus devraient pouvoir définir pour eux-mêmes le niveau de vie privée qui leur convient. Ceci peut être effectué en les avertissant chaque fois qu'une information sur eux est créée, accédée, ou modifiée et en leur donnant un certain contrôle sur l'utilisation de cette information. Cet avertissement informe les individus de qui utilise quelle information sur eux et leur permet de détecter les problèmes possibles de cette utilisation. Le contrôle permet aux individus de résoudre la plupart (ou au moins quelques uns) de ces problèmes. Bien sûr, ceci nécessite que les individus puissent avoir confiance dans les utilisateurs d'information afin d'implémenter correctement ces avertissements et d'offrir un contrôle efficace. Nous analysons le concept de confiance plus en détail et différencions une approche optimiste d'une approche pessimiste de la confiance. Toutes deux peuvent fournir les bases pour la protection des données personnelles et ainsi de la vie privée. La première est basée sur le concept classique du contrôle et des sanctions, alors que la seconde essaie de prévenir les comportements délictueux.

Nous choisissons de poursuivre l'approche pessimiste et de rechercher les moyens techniques pouvant être utiles à ces fins. Une technologie prometteuse est le paradigme des agents mobiles, qui est une approche récente pour structurer des applications réparties. Son idée principale est de déplacer à la fois le code et l'état de l'objet vers une autre entité pour une exécution à distance. Ceci indique que le paradigme des agents mobiles englobe également le paradigme de la programmation orientée objet, qui nous permet d'encapsuler un ensemble de données tout en specifiant une politique de contrôle d'accès sur celle-ci. Puisque l'agent mobile est physiquement déplacé vers un endroit distant qui se trouve sous le contrôle d'une entité différente, il doit être protégé de cette entité qui est responsable pour son exécution. Ce problème constitue la difficulté majeure pour l'utilisation du paradigme des agents mobiles pour la protection de la vie privé et est étudiée sous tous ses aspects. Basés sur la discussion dans la littérature appropriée, nous décidons d'une approche basée sur un périphérique inviolable et dans lequel on peut avoir confiance. Un tel périphérique est développé de manière

conceptuelle.

L'approche est étudiée plus en détail dans le contexte du paradigme des agents mobiles, où elle nous permet de réaliser des objectifs de protection plus élaborés, qui pourraient être désirés par le propriétaire de l'agent mobile. Ceux-ci sont développés sous forme de conduites, qui regroupent l'objectif, les exigences, ainsi que la spécification de la collaboration nécessaire pour atteindre cet objectif.

Enfin, nous retournons au problème initial et décrivons de quelle manière la technologie présentée peut être utilisée pour améliorer la protection de la vie privée. Ceci aboutit à un cadre assez complexe, dans lequel l'information sur les individus ne peut être utilisée librement, mais où cette utilisation est contrainte par le niveau de vie privée désiré par le sujet de l'information. Le problème le plus important de ce cadre est la complexité accrue, que les individus doivent traiter. Ce problème est abordé avec un niveau d'indirection supplémentaire qui tente d'isoler la complexité et de la déléguer à des experts dignes de confiance.

Nous pensons que cette approche, malgré sa complexité, est un moyen viable d'aborder les problèmes urgents de la protection de la vie privée, qui ne se prêtent pas à des solutions simples.

To Aldous Leonard Huxley,
for his brave vision.
May the warning be understood.

To M-J,
for ...[1]

[1]This is a work on privacy, remember?

Acknowledgements

I have been saving the writing of this special part of my thesis until the very last moment in order to have some more time to contemplate the many people who have somehow paved the way that led to this Ph.D. Unfortunately, I will still forget to mention some of you. Therefore, I would like to thank you first.

I am very grateful to Professor André Schiper, my academic supervisor, for his support and the confidence he has put in me when I embarked on the idea of writing a thesis on the esoteric topic of *privacy protection*. I hope he has not worried too much about this decision. Another major thanks goes to my parents (all of them!) for the assistance and the understanding they have given me during all these many years of education that just never seemed to be finished. *Vielen Dank für die Unterstützung und das Verständnis was Ihr mir während dieser vielen Jahre der Ausbildung, die einfach nicht Enden wollten, gegeben habt*[2].

For the time and effort they have invested in judging the contents of this thesis, I want to thank the members of the jury, Professor Jean-Pierre Hubaux, Professor Friedemann Mattern, and Dr. Michael Reiter as well as the president of the jury Professor Claude Petitpierre. I particularly want to thank Michael Reiter, who is at least partially responsible for the direction of this thesis, for his advice and his always open ears during those last years.

I want to thank all the past and present members of the LSE for making work a pleasure (almost all of the time :-) and for teaching me french – in particular François Pacull for teaching me the "real" french, Pascal Felber for making sure that my body stays in shape, Xavier Defago for helping me publish my only paper in french, Sebastian Staamann for spawning the idea of using mobile agents, and Kristine Verhamme for taking care of all the small and big problems in administration. Also a great thanks to Levente Buttyán, Simon Znaty and the other people from the ICA who were great to collaborate with. A special thanks goes to Levente Buttyán, Xavier Defago, Stefan Pleisch, and Sebastian Staamann who have spent a considerable amount of their precious time proof-reading this text and looking for flaws. I am indebted to the numerous members of the "Kolleg für Sicherheit in der Kommunikationstechnik" for the many wonderful workshops and the inspiring critique – in particular I want to thank Andreas Pfitzmann, who is simply a great person besides being a brilliant researcher, for sharing his enthusiasm with us.

Most of all I want to thank Marie-Jo for keeping me sane during these past months as well as for countless other things.

Finally, I would like to acknowledge the EPFL as the institution that has provided the financial support for my research ("Privacy" project).

[2]Ich bin mir aber trotzdem noch nicht sicher, ob ich jetzt wirklich "fertig" bin.

Contents

Introduction **1**

Protection of Confidentiality 2

Protection of Privacy 2

Goal of this Thesis 3

The Overall Approach 4

Contributions of the Thesis 5

Organization of the Thesis 5

1 The Protection of Privacy **7**

1.1 Introduction 7

1.2 The Concept of Privacy 8

1.2.1 Examples where Privacy is at Stake 9

1.2.2 Dataveillance 11

1.2.3 Privacy is not the Exclusive Goal 12

1.2.4 Complete Privacy is not Achievable 12

1.3 Current Approaches to Privacy Protection 13

1.3.1 Data Avoidance 13

1.3.2 Data Protection 13

1.3.3 Multilateral Security 15

1.4 Basic Principles for Privacy Protection 16

1.4.1 Notification 17

1.4.2 Control 18

1.4.3 Trust 19

1.5 Two Approaches to Trust 19

1.5.1 The Policy 19

1.5.2 The Optimistic Approach 21

1.5.3 The Pessimistic Approach 21

1.6 Enforcement of Privacy Protection 22

1.6.1 Enforcement in the Optimistic Approach 22

1.6.2 Enforcement in the Pessimistic Approach 23

1.6.3 A Similar Approach 25

1.6.4 An Overview of our Approach 26

1.7 Relevance of Privacy 26

2 System Model and Mobile Agents 29
2.1 Introduction 29
2.2 The World of Mobile Agents 29
2.2.1 The Environment for Mobile Agents 30
2.2.2 An Example 32
2.3 Mobile Code 33
2.3.1 Some History 33
2.3.2 The Basic Concepts of Mobile Code 35
2.3.3 Simple Paradigms for Mobile Code: REV and COD 36
2.4 The Mobile Agent Paradigm 37
2.4.1 The Mobile Agent 38
2.4.2 The Agent Platform 39
2.4.3 Some Agent related Terminology 39
2.5 The Services of the Mobile Agent Paradigm 42
2.5.1 Mobility Support 43
2.5.2 Agent Execution 45
2.5.3 Resource Management 46
2.5.4 Heterogeneity Support 47
2.5.5 Communication 47
2.5.6 Security 48
2.5.7 Fault-tolerance Support 51
2.5.8 Directory Assistance 52
2.5.9 Agent Management 52
2.6 The Use of Mobile Agents 53
2.6.1 Advantages of the Mobile Agent Paradigm 53
2.6.2 Applications of the Mobile Agent Paradigm 55

3 Protecting a Mobile Agent with Tamper-resistant Hardware 57
3.1 Introduction 57
3.2 Related Work 59
3.2.1 Cryptographic Traces 61
3.2.2 Forward Integrity 62
3.2.3 State Appraisal 63
3.2.4 Co-operating Agents 63
3.2.5 Obfuscation 64
3.2.6 Tamper-resistant Hardware 65
3.2.7 Computing with Encrypted Functions 66
3.2.8 Evaluation 66
3.3 Tamper-resistant Devices 67
3.4 Realization 68
3.4.1 Notation 69
3.4.2 The Trusted Processing Environment 70
3.4.3 Controlling the TPE Manufacturer 72
3.4.4 Key Management 73
3.4.5 The CryPO Protocol 73
3.5 Problems 76
3.5.1 Trust in the TPE Manufacturer 76

3.5.2 Limitations 77
3.6 Open Systems 78

4 Usage and Extensions 81
4.1 Introduction 81
4.2 Conducts 82
4.2.1 Basic Protection 82
4.2.2 Migration Control 84
4.2.3 Protected Invocations 85
4.2.4 Authentic Code 86
4.2.5 Synchronized Clock 87
4.2.6 Local Logging 88
4.2.7 External Logging 90
4.2.8 Anonymity 91
4.3 An Example: The TTP Agent 93
4.3.1 The Shopping Agent 93
4.3.2 The Logging Service 94
4.3.3 Other Services 95
4.3.4 Discussion 95
4.4 Other Application Areas 96
4.4.1 Copyright Protection 96
4.4.2 Secure Circuit Evaluation 96

5 A Framework for the Protection of Privacy 99
5.1 Introduction 99
5.2 An Example: Mobile Telecommunications 101
5.3 A Simple Framework 103
5.3.1 The Use of Data Agents 104
5.3.2 The Distribution of Data Agents 105
5.3.3 Withdrawal of Data 106
5.3.4 Enforcing the Use of Data Agents 107
5.4 Making it Manageable 107
5.4.1 The Data Agent – Accessor Agent Paradigm 108
5.4.2 Using the da^3 Paradigm 109

Conclusion 111
Assessment of the Protection of Mobile Agents 111
Assessment of the Framework for Privacy Protection 112
Personal Conclusion 112

Bibliography 114

A Enforcing an Itinerary 129

B Policy Issues 131

Curriculum Vitae 133

List of Figures

1.1 Assessing the adequacy of a policy 20

2.1 The basic idea of mobile code 36
2.2 The Infrastructure for Mobile Agents 38
2.3 A strategy to guarantee agent survival 52

3.1 Overview of the Principals in the CryPO protocol 69
3.2 The trusted processing environment 70
3.3 Initialization of the CryPO protocol 74
3.4 Usage of the CryPO protocol 75

5.1 Location management in GSM 101
5.2 Location management with a data agent 102
5.3 The distribution of data agents 106
5.4 The data agent – accessor agent paradigm 108

List of Tables

2.1 The Agent Lifecycle 41

3.1 The Detection Mechanisms 60
3.2 The Prevention Mechanisms 60

Introduction

They that can give up essential liberty to obtain a little temporary safety deserve neither liberty nor safety.

– Benjamin Franklin

After the industrial revolution, which replaced muscle power with machines and thus enabled people to become exceedingly more productive, the information revolution promises a similar upheaval in our working environment as well as in our social system. The information era's prospect of high quality information at the right place and time, which allows companies and individuals to make better decisions, will again considerably increase the productivity of people.

This trend can be illustrated with many examples from the *World Wide Web* (WWW), which enables us, for instance, to almost instantaneously access scientific publications or to effectively plan business trips. Another, less visible, but equally potent example is the emergence of data-warehouses within large corporations. These data-warehouses gather and concentrate all the available information within the corporation, make it amenable to sophisticated analysis, and assist in the decision process.

New developments will further intensify this by providing users with on demand access to information and communication services at any time, any place, in any volume, and in any form [BBI93]. All the different communication infrastructures (e.g., POTS, ISDN, GSM, UMTS, or Internet) will become increasingly integrated. Information and communication services will be adapted to share the same terminal equipment and will thus be accessible from a wide range of deployed hardware (e.g., cell phones, TVs, or game consoles).

Due to the requirements of work, people become more mobile and more physically dispersed. This is already the case today and technologies such as the telephone or satellite TV allow people to stay – more or less – in touch with the people that are important to them or with their home country (i.e., their culture). New communication infrastructures can make this situation more tolerable. For instance, imagine a group of old high-school friends meeting regularly in a bar in cyberspace where they can engage in group interaction as well as have private chats, while being physically scattered all over the globe. Other, more straightforward usages of the new technology are to conduct various transactions, that are currently accomplished by physical interaction, via the new information and communication infrastructure, such as booking a flight from one's work PC or buying a cinema ticket via a cell phone.

On the other hand, all of the interactions carried out on digital media are potentially subject to eavesdropping, spoofing, or traffic analysis. It is therefore vital to effectively protect these interactions.

Protection of Confidentiality

The first requirements for the protection of information and communication were identified by the military (e.g., Caesar's Cipher [Sch94] or Enigma [Sch28]), which has led to a close association between research on encryption technology and military research [Kah67]. With the appearance of information processing, the role of information and communication protection has also gained importance in several information centric branches of the commercial domain (e.g., banks and insurance companies had to protect financial transactions). The major requirements in these applications are the authentication of well-known principals as well as the confidentiality and integrity of information communicated between mutually trusting principals. These requirements can be fulfilled by symmetric encryption mechanisms and symmetric message integrity mechanisms.

In the information era, information has become one of the most valuable goods. The fact that this good needs protection has meanwhile been recognized and applied in the context of most companies, which protect their operational data as well as their data exchange with geographically dispersed branch offices but also with competitors. This has led to the development of powerful concepts to establish protection domains as well as more open and flexible confidentiality and integrity protection mechanisms, such as public-key encryption and digital signatures. The in-depth discussion of these complex technologies is not within the scope of this thesis. For more information, please refer to [MvOV97, For94, CZ95].

Protection of Privacy

While the necessity to protect military or operational data of corporations is widely accepted, the same is not the case for information on individuals. If we consider this type of information, then we often talk about *privacy*, which was defined by Westin in [Wes67] as "the claim of individuals, groups, and institutions to determine for themselves when, how, and to what extent information about them is communicated to others". More precisely, we are concerned with the protection of *informational* privacy, which was identified by Rosenberg in [Ros92] as one of three aspects of privacy. The other two aspects, which are territorial privacy and privacy of the person, relate to physical issues. In this context, we will also refer to individuals as *data subjects*.

The increasing use of communication and information systems makes a lot of data about our everyday life available in digital form, which was previously not even accessible to ourselves (e.g., how many tubes of toothpaste a person consumed during the last 5 months). This data can be used in many different ways, some of which are desired by the data subject (e.g., to receive a quantity discount or for accounting purposes), while others are not (e.g., a medical insurance company that covers expenses for dentist treatment may use the data as an argument to raise premiums due to an increased risk or even to reject future payment due to presumed negligence on the part of the insured person).

An important difference between the information on individuals and the information in military or commercial systems is that in the former case the principal that generates and holds the information, which is often called *controller*, is not the only one who has a legitimate interest in the information.

There are many arguments both in favour and against the deliberate use and exchange of information on individuals. However, due to the discrepancy in power between individuals, who usually are the data subjects, and governments or multinational corporations, who usually are the controllers, and due to the fact that individual users are not necessarily very well informed about their rights, there is a strong argument supporting the regulation and control of this issue. This has led to the whole discipline of *data protection*, which is one of the measures that can ensure informational privacy [FH98]. The advent of powerful and ubiquitous communication and information services will make the protection of informational privacy much more important but also much more difficult. This is due to the dissolution of two "natural" protections of privacy: the cost of information processing and the compartmentalization of information (at different organizations). We will further discuss the issue of privacy in Chapter 1.

Goal of this Thesis

We want to address the problem of data protection and privacy on a technical level. The visionary goal would be to define an architecture that allows to realize arbitrary information and communication services in such a way that they protect the privacy of their users sufficiently well. This requires a precise definition of what constitutes a sufficient protection of privacy, which is not only dependent on the individual to whom this protection should apply, but also on the society in which he[3] lives. The discussion of this is far beyond the scope of this thesis (and probably also of computer science as a discipline).

Therefore, we want to concentrate on three simple principles, which we consider to be basic building blocks to achieve the visionary goal: *notification*, *control*, and *trust*. The principle of notification states that data subjects should have complete knowledge about what data on them exists, who has access to this data, and who actually uses this data (under special circumstances, that have to be defined by legal institutions, this notification may be made available with a certain delay). The principle of control states that data subjects should have some control over this data (within limits that have to be defined by legal institutions), so that they can actively participate in the effort to keep the data accurate and maybe even restrict the access to data they would prefer to remain confidential. The principle of trust states that the data subject has reason to believe that the first two principles are actually observed. Thus it is not directly concerned with the management of personal data, but provides the basis on which the others are built.

The first two principles are grounded in the *right of informational self-determination*, which was identified by the German Constitutional Court in its Census Decision of 1983 [FH98] and are, for instance, identified in Art. 10 to 12 of the EU-Directive on Data Protection [EC95], which identifies the data subject's right to information about the collection of data and the right to rectification, erasure, or blocking of incorrect or illegally stored data. The data subject then has to trust the legal system to actually enforce these principles. We believe the principles to be ethically neutral in the sense that we assume that nobody would reasonably object to them in the generic way in which they are stated.

[3]We are aware of the problems caused by pronouns that are biased towards a single gender. However, due to the complicated nature of our subject, we do not want to complicate the presentation further by using multiple or alternating pronouns. The words "person", "he", and "him" (in the corresponding context) are supposed to refer to all human beings.

The Overall Approach

In contrast to the most frequently pursued purely legal approach, which has to rely on supervision and sanctions to enforce the basic principles (see also Section 1.6.1), we want to devise a technical mechanism that supports the direct enforcement of the basic principles. This should give legislators technical tools to accomplish their task more effectively. The idea is to replace the supervision of compliance with the basic principles by a supervision of proper usage of the technical enforcement mechanism. The latter can be more tractable if the technical enforcement mechanism is constructed such that an improper usage is easy to detect and prove. This should then also increase the trust of the data subject in the enforcement of the principles of notification and control.

The mechanism we propose, consists of encapsulating data on individuals in an object. This object can be located in a domain that is physically controlled by the principal who uses the data, but protects its data in the interest of the data subject. The object does this mainly by communicating information on accesses to its data back to the data subject and by implementing fine-grained access control on its data.

The main problems that we encounter here are scale and enforcement. The problem of scale is concerned with the sheer scope of the proposed solution, which requires a means to notify the data subject about data accesses, some support for the data subject to process these access notifications, and the definition of an access policy for each particular data item.

The problem of enforcement is actually twofold. First, it requires a mechanism to prevent that the notification about the access is suppressed or the access control of the encapsulating object is broken. Second, it must be prevented that data that was correctly accessed is simply copied, so that it can later be accessed without the protection of the encapsulating object, potentially by a different principal.

The latter is a fundamental problem, for which no satisfying technical solution exists. It is the reason why our approach can not enforce the basic principles directly, but can only support their enforcement. The problem is that once some information is disclosed to another principal, the original holder of the information loses all control over this information. The receiver can use it without any constraints, duplicate it, or disclose it to a third party. This problem has also been explored in the context of the protection of intellectual property [Ste96] and constitutes a problem for electronic cash (to prevent double spending [Cha92]). We will return to this issue as well as to the problem of scale in Chapter 5.

The first problem of enforcement is explored in the context of *mobile agents*, which suffer from a very similar problem. Mobile agents are active objects that can move to remote locations in order to accomplish some task for a user (see Section 2.4.1 for a more precise definition). These remote locations provide an execution environment where the agent can accomplish its task (for the user), which is under the physical control of another principal. Since a mobile agent may contain confidential or otherwise valuable information, it should be protected from access and manipulation by possibly malicious principals that may try to abuse the mobile agent. This problem has been explored in a considerable body of work, which identifies several different approaches to the problem. The one we have chosen is based on tamper-resistant hardware [AK96, NIS94] and public-key cryptography [DH76, MvOV97]. The notion of tamper-resistance indicates that the hardware is physically protected from its environment and that it only interacts with its environment through an interface it controls. We will present the mobile agent paradigm in Chapter 2 and discuss our approach to protect mobile agents from their execution environment together with several other approaches in

Chapter 3.

We can now interpret the object that encapsulates the data on individuals as a mobile agent, and will also refer to it as *data agent*. It is, thus, possible to apply the same protection mechanisms that protect a mobile agent when executing at a remote location, to a data agent. These mechanisms can prevent that the access control of the data agent is broken. The problem of preventing that the access notification is suppressed can be resolved by delaying any access until the data agent has received an acknowledgement from the data subject, stating that the access notification has been received. If the tamper-resistant hardware is trusted by the data subject, then he has a reason to believe that the principles of notification and control will be enforced.

The close connection between data agents and mobile agents explains why a lot of the work presented in this thesis is centered on the protection of mobile agents. This topic is in itself an interesting and challenging problem and has led to some results that are not immediately relevant to data agents. However, it was originally only addressed as a vehicle to solve the problem of informational privacy.

Contributions of the Thesis

The thesis provides an analysis of the problems related to privacy in the information society and points out the importance of trust in this context. A closer examination of trust leads us to the identification of the mobile agent paradigm as a technical means to address the problem of trust.

We discuss the mobile agent paradigm and recognize the problem of protecting a mobile agent from its executor as the major issue that needs to be resolved. We then classify the approaches to this problem that can be found in the literature and develop our own approach, which is based on a trusted and tamper-resistant hardware device.

The investigation of the proposed approach in the context of the mobile agent paradigm leads to the identification of numerous conducts. These describe how and under what assumptions particular protection goals, which may be desirable for the owner of a mobile agent, can be realized.

Finally, the thesis presents a conceptual framework, in which the presented approach is used to address the problem of privacy protection and proposes a mechanism to reduce its complexity.

Organization of the Thesis

As explained in the previous section, the work presented in this thesis is centered on two related issues:

- protection of privacy: this is discussed in Chapters 1 and 5;
- protection of mobile agents: this is discussed in Chapters 2, 3, and 4.

More precisely, the thesis is structured as follows. In Chapter 1 we further explore the issue of privacy, identify the importance of trust for security (and consequently for privacy) in distributed systems, and discuss why we believe that a technical approach to enforcement can be superior to other approaches. Chapter 2 introduces the mobile agent paradigm and identifies the particular problem we are concerned about. This problem, which is the protection of a mobile agent from its execution environment, is addressed in Chapter 3 with an approach based on tamper-resistant hardware. This chapter also presents alternate approaches to address the problem. In Chapter 4 we further develop the approach by introducing several extensions to the basic technology and explore their usage for various purposes. Then, in Chapter 5 we return to the problem of privacy protection and explain how the presented technology can be used to support the enforcement of the principles of notification and control. Finally, in the Conclusion, we summarize the major results of this work and outline future research and possible developments around the presented technology.

$\infty\infty\infty$

Chapter 1

The Protection of Privacy

If privacy is outlawed, only outlaws will have privacy.

– **Phil Zimmerman**

1.1 Introduction

To illustrate our concern for privacy, we want to return once more to the industrial revolution and explore the problem of safety, which we believe has certain similarities with the problem of privacy today. The following discussion is based on the extensive presentation by Leveson given in [Lev95].

In the beginning of the industrial revolution, the working conditions in factories were harsh and dangerous. Workers were considered expendable since they cost nothing to hire or replace. Modern standards for working safety that are defined and enforced by government institutions to protect the life and health of workers were almost non-existent. On the contrary, it was considered the responsibility of the workers to protect their safety. There was no social protection (other than family and friends) for a worker who suffered an accident or contracted a disease that was caused by the conditions in his working environment. A worker would have had to sue his employer to obtain some compensation, which was expensive and difficult. The workers were dependent on the salary paid by their employer and often not even aware of potential or actually existing dangers (e.g., toxic chemical substances).

This situation changed only gradually, when more and more workers started to organize in unions to protest against their working conditions and when social ideas (i.e., communism and socialism) gained more political influence. This led to laws for workers' compensation, security insurance, and the establishment of safety standards. When employers were forced to pay for accidents of their workers, they attempted to prevent these accidents. Another market motivated factor was that the machines in the factories became more complex and required extensive training of workers who only became productive after a considerable time. In the case of a severe accident, the overall productivity was diminished and the investment in the training of the employee was lost. This general interest in safety sparked increased investment in safety technology (e.g., two-handed operation or improved illumination and ventilation), which actually protected the life and health of workers (instead of offering financial compensation for caused harm).

Fortunately, a violation of privacy does rarely endanger the life and health of an individual and is therefore less critical than a violation of safety. Nevertheless, as we will discuss below, it may have severe effects on the social life of a person and thus on society in general.

The similarities that we identified between safety and privacy are the following. In both cases there is a substantial discrepancy in power between the principals, i.e., the employers or controllers on one side and the workers or data subjects on the other. This is further aggravated by a discrepancy in knowledge about the actual operation of the system and the legal situation. An evaluation phase is necessary to detect and assess the inherent dangers, which results in an organization of the weaker principals into associations that promote the common interest (i.e., unions and consumer rights groups). Eventually, these interest groups gather sufficient influence to constrain the more powerful principal either by changing the legal situation, by creating or exposing market oriented incentives, or by a combination of both. This issue has not been fully realized in the context of privacy protection but there is evidence for increasing political support [EC95] as well as for organized self-regulation (e.g., TRUSTe [Dys97]). The last step, which consists in the development of privacy technology and privacy protecting infrastructures that can be used by individuals to actively protect their privacy, has started in particular application domains, such as communication or payment (e.g., PGP [Zim94], Mixes [Cha81], or anonymous electronic cash [Cha85]). A more comprehensive effort that aims to define a specification for privacy policies in the context of WWW usage data is the Platform for Privacy Preferences Project (P3P) of the W3C [RC98]. The goal of this thesis is to address the underlying problem of the dissemination and usage of personal data on a technical level and in a more generic way.

The question of whether and to what extent privacy should be protected is difficult to answer. Issues of this type have been discussed by Weinberg in [Wei72], where he introduces the concept of trans-science. “Many of the issues that lie between science and politics involve questions that can be stated in scientific terms but that are in principle beyond the proficiency of science to answer [...] I propose the term “trans-scientific” for such questions [...]”. Answering trans-scientific questions requires the establishment of *values* which must be determined by political processes. This is clearly beyond the scope of this thesis, but we would like to point out that privacy has been declared a fundamental human right in Art. 12 of the Universal Declaration of Human Rights of the United Nations [UN48].

1.2 The Concept of Privacy

In order to discuss the technical problems it is necessary to have a clear understanding of the concept of privacy, beyond the intuitive one that we have assumed until now. An exact meaning of privacy is quite difficult to define since it has many different meanings in every day language. For instance, the *Webster's Third New International Dictionary of the English Language* gives eight definitions for the adjective and the noun *private* and five more for the noun *privacy.* This ambiguity can be useful for a spoken language, but it makes a scientific approach difficult.

One of the earliest legal definition of privacy was given by Warren and Brandeis in [WB90], who identified privacy as “the right to be left alone”. This definition is very general and includes various issues concerned with physical intrusion that we will not address. This distinction has been identified by Rosenberg in [Ros92], where he presents three aspects of privacy:

- territorial privacy, which is concerned with the physical area of a person (e.g., his home);
- privacy of the person, which is concerned with direct physical access to a person (e.g., a physical search);
- informational privacy, which is concerned with information about a person (e.g., the processing of personal data).

The last aspect is the one that Westin has identified in one of the classical works on privacy [Wes67], in which he discusses the importance of privacy for free societies. There he defines privacy as "the claim of individuals, groups and institutions to determine for themselves, when, how and to what extent information about them is communicated to others". This definition is well accepted by consumer rights groups (e.g., Center for Democracy and Technology – CDT, Electronic Frontier Foundation – EFF, Electronic Privacy Information Center – EPIC, Privacy Rights Clearinghouse – PRC, or Privacy International – PI) and in concordance with the information related parts of the definitions given by Margulis [Mar77] and Lunheim and Sindre [LS94]. We will later see that it also matches well with our technology driven approach to address the protection of privacy (see Section 1.4)

Since we adopt this somewhat restricted view of the concept of privacy, we will in the following understand the term privacy to refer only to the aspect of informational privacy. We believe that this restriction causes no problems for our application domain due to the inherent lack of physical interaction in the use of communication and information systems.

We now want to give an overview of the actual problems related to privacy. These are mainly concerned with the increasing amount of data on individuals, the data subjects, that is created in modern communication and information systems. This data is generated and held by a principal, the controller, who shares a legitimate interest in the data with the data subject. Due to the decreasing cost of data storage and processing it becomes technically possible and economically justifiable to gather and analyze large amounts of data. Furthermore, due to the improving communication facilities, data can easily be accessed from a physically remote location and linked with local data. This leads to a considerable refinement of the quality of information that is contained in the data[1] and can be used to supervise the behaviour of data subjects.

1.2.1 Examples where Privacy is at Stake

In order to more easily situate our concerns and to illustrate the type and depth of information that is created, we want to start by discussing some real-life examples. We will see that these examples touch almost every aspect of our daily life.

Shopping

The increasing use of cashless payment methods and personalized discount cards allows shops to identify customers and to analyze their buying habits. This is obviously not restricted to the purchase of physical goods, but also applies to other merchandise, such as entry tickets to a theater or a museum. This will probably be magnified by electronic commerce systems, which make the collection of data more efficient and comprehensive (a customer who is repeatedly looking at the description of a particular item is probably interested in buying it).

[1]We will often use the terms data and information in a very similar way. Usually both are interchangeable, with data referring to the encoding in bits and information referring to the interpretation of these bits.

Telecommunications

Telecommunication systems always gathered information about the communication relations of people. This is called *transactional data* and concerned with who is communicating when, how often, and from where with whom. With the advent of mobile communication systems (e.g., GSM, IS-41, IS-95, JDC) this information becomes much more personal and reflects the communication relations of a single person instead of a group of people (e.g., a family that shares a single phone). It also reduces the amount of anonymous communications, which occurred naturally when people used public phones. Modern mobile communication systems create a new type of information since they have to store the location of a user in order to route an incoming call [MP92]. This allows to construct a complete trace of the movements of a user carrying a mobile phone in standby-mode. These movement profiles [FJK$^+$97] will become more and more accurate with the projected decrease of cell sizes [Ste94].

Medical Information Systems

The increasing cost of healthcare and the specialisation of practitioners mandate a more efficient sharing of medical information. Researchers also have a legitimate requirement for comprehensive access to medical data (which can be anonymized) for statistical analysis. Even though people may be quite willing to provide all relevant information to doctors and hospitals, they might be reluctant to provide their medical information to employers or insurance companies [And96]. The increasing interconnection of these systems and the growing number of principals who can potentially access them will make an effective control of who accesses what information for what purpose an indispensable requirement. Otherwise, people might hesitate to speak freely with their doctor about their health problems.

"Public" data

All the messages that are posted to usenet newsgroups or the information on WWW pages are public data that can be accessed by everyone. However, the creation of huge news archives (such as *Deja News* [Dej] with powerful search primitives that allow us to find all the messages posted by a single person over an extensive period of time to different newsgroups, gives these messages a new dimension by revealing the various interests and opinions of the person in question. The same is true for personal pages on the WWW, which are temporarily buffered by search engines (e.g., *Google* [Goo]) but also permanently archived by *The Internet Archive* [Kah97] to save them for historical research. This precludes the possibility of the poster of the page to simply remove it by removing the access rights.

A related problem is the global distribution of public records that are collected for specific purposes and only relevant in restricted contexts (e.g., Florida's list of Sexual Predators [Flo97] or the National Association of Security Dealers' list of stockbroker's backgrounds [NAS]). This information, which previously had to be searched for in the appropriate context, may now easily show up in a general search on a person's name (this may even cause problems for different people who have the same name).

Identification Systems

Several proposed systems (some of which are in experimental stages) for access control to or localization within buildings (active badges) [WHFG92], public transport systems (personalized token cards), or highways (automatic toll collection) will also allow their operators to create movement profiles.

Even more futuristic proposals to use radio emitters for precisely localizing trucks[2] (to track the location of goods), cars (to prevent theft), or even children (to protect them from crimes) can hardly be evaluated in their possible effects.

1.2.2 Dataveillance

The information that is generated in the cited examples can obviously be used for surveillance purposes. In this context we will also refer to this form of surveillance as *dataveillance*, which is a term coined by Clarke in [Cla89], as meaning "the systematic use of personal data systems in the investigation or monitoring of the actions or communications of one or more persons".

There are two well understood arguments against dataveillance. First, there is a possibility that the system does not function correctly and returns wrong information on the data subjects. This is a well-known problem of current information systems, which are frequently found to contain erroneous data due to false entries by operators or an incorrect operation of the system. Such a slip can lead to serious problems for a data subject (e.g., refused credit applications). This is also the reason for the right of the data subject to rectify inaccurate data, which is identified in many legislations on data protection (see Section 1.3.2). The main problem for the data subject then remains to find out about the existence of the inaccurate data. On the other hand, the same argument of inaccurate data is also used by advocates of dataveillance systems to argue for improved systems which contain more accurate information [LS94].

Second, there is the possibility that a dataveillance system can, even though it was institutionalized by democratic forces, fall under the control of totalitarian forces. This could support or even enable an Orwellian [Orw48] type state where a dictatorship uses the available technical means to control the population. The example of the Third-Reich, which grew out of a democracy (the Republic of Weimar) shows that this issue can not simply be ignored.

According to Lunheim and Sindre [LS94] these are not the only possible problems. Even in a properly functioning democracy with strong safeguards against faulty data, there are considerable dangers due to a lack of privacy. Given the increasing amount of personal information that becomes accessible to everyone (e.g., "public" records on the WWW) there is a severe risk that "the danger of dataveillance is not that the masses may be controlled by a narrow elite – it might equally well be the elite being controlled by the masses, or the masses controlling themselves" [LS94], and in particular any possible elite that might begin to distinguish itself from the masses. This can, as Lunheim and Sindre further argue, lead to cultural inflexibility and stagnation. The problem is that a person who deviates from the norm or engages in legal but socially unaccepted actions (e.g., abortions [Tim99]) can easily become the subject of harassment, which might lead to undesirable self-restraint. Also, if every statement that is made to a well defined group of people can later be taken out of context and provided to people not known in advance, then one might be much more reluctant to speak freely, which leads to a loss of freedom of expression.

Consider, for instance, a person who committed a serious crime during his youth and

[2] In a recent report aired on German Television (ARD), a police officer explained that tired truck drivers sometimes call them for a control to be ordered by the police to take a break of several hours. The police officer explained that this was often their only possibility to take a break without subsequent problems with their employer.

served an appropriate sentence. In the past it was always possible to move to a remote place to start anew without having to cope with prejudice from others and to become a valuable and respected member of the new community. Due to the extensive amount of information available to everyone, this is hardly possible any more – often the information about a previous conviction will be the first information available to someone who investigates and will not be tempered with additional information about positive changes (a similar example is discussed by Wasserstrom in [Was84]).

1.2.3 Privacy is not the Exclusive Goal

Despite the described problems, we do not want to advocate a society where privacy is the most important social good. However, it should be clear that privacy has a social utility, which can serve to advance the goals of the entire society. In this context, privacy should be considered as a means to an end, where the end has to be defined in democratic political processes by the society as a whole[3].

This means that complete privacy is not desirable, since it may be abused by individuals (or groups) to advance goals that are opposed to those of the society (e.g., criminals or non-democratic forces). It is, thus, necessary to find a balance between the interests of an individual and those of society. Another example where privacy needs to be constrained, is medical data on an individual. Even though this data is highly confidential and any access to it should be extremely limited, it is in the data subject's own interest that any protection can easily be overcome. This is important in the case of an accident where the data subject would like to provide a doctor with complete access to his medical data, but is not capable to provide this access if he is unconscious.

Since people have very different standards concerning what constitutes an unacceptable privacy intrusion and what is still acceptable[4], it seems appropriate to let them define for themselves the level of privacy they are comfortable with. Our goal is therefore to devise mechanisms that allow individuals to control the access to their personal data within constraints defined by society that limit the achievable level of privacy.

1.2.4 Complete Privacy is not Achievable

The possible level of privacy that is achievable is not only limited by constraints imposed by society. Even if the society would not restrict privacy in any way, there is still a principle problem, to which we will refer to as the *problem of information disclosure*. Once some information is disclosed to another principal, the original holder of the information loses all control over it. The receiver can use it without any constraints, duplicate it, or disclose it to a third party. This is a fundamental problem, for which no technical solution exists.

The same political processes that were identified to limit an individual's achievable level of privacy should also restrict the use of data legally acquired by the controller. Since there is no way to enforce such restrictions directly, our goal is to devise mechanisms that support the enforcement.

[3]This tension can be observed in the EU-Directive on Data Protection [EC95], which identifies privacy as a fundamental human right (Preamble §2) but then restricts it (Art. 13) for reasons of (among others) national security, public security, and important economical interests of the society.

[4]For instance, the UK has resisted to a photo ID for reasons of privacy, which is perfectly accepted in Germany, France, and the United States [Bel97].

1.3 Current Approaches to Privacy Protection

The concept of privacy is not only hard to define, but it is also difficult to identify appropriate measures to ensure privacy. All possible measures are mainly constrained by the problem of information disclosure. It leads to the situation that the controller and the data subject are two different principals, who both have a legitimate interest in the data but probably different goals concerning its use. This is quite different to classical security, where the controller is the only one who has a legitimate interest in the data and can therefore take every necessary measure to protect it. Since the data subject has lost any direct control over the data, he is in an inherently weaker position than the controller. This is the issue that needs to be addressed by measures to ensure privacy.

We want to briefly explore three different approaches that are most often pursued to ensure privacy protection. We consider these approaches in the context of an arbitrary interaction, which can be any form of transaction (e.g., purchase) or communication between multiple parties.

1.3.1 Data Avoidance

The simplest and also one of the most effective measures for privacy protection is not to disclose data in the first place, which limits the amount of available data. By investigating for every data item whether it is actually necessary for the desired interaction, and by omitting needless data, an interaction can be conducted with the disclosure of a minimal amount of data.

This measure is not always helpful since the data may be inferred from the context of the interaction (e.g., the time or location) or generated by another entity (e.g., the phone number of a caller in the ISDN system or the recording of a surveillance camera). Furthermore, any deliberately omitted data can often be obtained from other sources, based on the identity of the data subject. Therefore, the main utility of data avoidance is in the context of anonymous interactions, where one (or sometimes even both) of the principals does not identify itself to the other (e.g., the use of public phones for communication or cash transactions). Obviously this measure limits the possible activities of a principal to those that can be conducted anonymously and where the other party in the interaction accepts to interact with an anonymous principal (this might be mandated by legislation).

Another possible measure that leads to data avoidance is the removal of identification information when it is no longer needed. This can often be used for statistical analyses, which need to ensure that provided data is correct and authentic, but do not need the identifying information for the actual usage of the data.

A major advantage of data avoidance is, that it is self-enforcing. Data that has not been disclosed is not subject to the problem of information disclosure and can not be abused by another principal. Therefore, it should always be considered first, whenever a problem of privacy protection needs to be addressed.

1.3.2 Data Protection

The most established approach to privacy protection is what is widely referred to as data protection. Its goal is to define legislation or voluntary guidelines (the latter is often referred to as self-regulation) that regulate and control how data that was disclosed by a data subject (either deliberately or in the course of an interaction) can subsequently be used by the controller.

The measures that are most often identified are those of the *principles of fair information practices*, which prescribe basic principles that have to be guaranteed when personal data are collected or processed. For instance, they require openness about data collection and the right to rectify inaccurate data [Gel97]. The principles of fair information practices were originally developed for the United States' Privacy Act of 1974 [USC74], and can be found in many other legislations, such as the EU-Directive on Data Protection [EC95]

The Privacy Act is limited to legislation of the public sector, while the private sector in the United States is only covered by isolated and uncoordinated laws, such as the Fair Credit Reporting Act of 1970 (FCRA) [USC70] or the Video Privacy Protection Act of 1988 [USC88] (as well as others for various sectors, such as cable viewing or educational records). According to the analysis of Gellman in [Gel97], the Privacy Act was not very successful due to statutory and administrative shortcomings in its implementation.

The EU-Directive, which was adopted in 1995 and had to be integrated into the national laws of the member states by October 1998, is more comprehensive (it addresses both the public and the private sector) and is based on a strong European precedent of successful privacy protection legislation [MS97]. For instance, it prescribes that companies can only process personal data if the data subject has given his consent or if it is necessary to handle a transaction, to fulfill a legal obligation, or in the data subject's or the public interest. The law further prohibits the export of personal data to countries that do not observe the legal requirements. The EU-Directive is considered the most advanced legislation for privacy protection that "may be a valuable model for countries currently without data protection laws" [Gre96].

Examples for self-regulation are the initiatives of TRUSTe [Dys97], the Platform for Privacy Preferences (P3P) by the W3C [RC98], and the guidelines of the Individual References Services Group (IRSG) [IRS98].

The goal of TRUSTe is to offer companies a *trustmark* for their web pages. To obtain the right to use such a trustmark, a company has to choose a privacy policy from a set of policies defined by TRUSTe, undergo an audit with an approved auditing firm to verify if its internal operation complies with this policy, and pay a fee to TRUSTe. As a result, customers are informed about the privacy policy of a company and can be reasonably sure that this policy is actually adhered to.

This approach is very inflexible and will be complemented by P3P, which provides a framework for informed online interactions. The goal is to allow more fine grained definitions of the privacy policies and to facilitate automated negotiations between a service provider and a user. This informs users about the privacy practices and allows them to control the disclosure of information. The approach does not inherently enforce the privacy policies and relies on other mechanisms to achieve this (e.g., TRUSTe).

The guidelines of the Individual References Services Group (IRSG) were formulated to comply with the EU-Directive and took effect in January 1999. These guidelines urge information vendors to publish their policies for data usage and should limit the distribution of data. However, even though they prescribe a regular audit of the signers, there is no simple way to seek relief from violations of the guidelines. There is no mechanism through which a data subject can gather evidence of a violation of the guidelines [Law98].

In contrast to data avoidance, data protection is not self-enforcing, but compliance with laws and guidelines has to be actively controlled by the data subjects or some other independent institution. This is a major problem of data protection and we will defer its general discussion to Section 1.6.

1.3.3 Multilateral Security

The general idea of multilateral security (which is sometimes called multi-party security) can be traced back to Chaum, who identified in [Cha85] a "new approach [to security that] allows all parties to protect their own interests". The definition we will use here is based on Rannenberg et al. [Ran94, RPM97]: multilateral security is a concept that recognizes the fact that all participating parties in a particular service interaction should be equal and should be capable to define their security requirements, which then must be respected by the implementation of the service. It removes the still often present constraint that security and protection mainly or only apply to the operator of some IT or telecommunications equipment.

A typical example for privacy protection in the context of multilateral security is that of a principal, say *Alice*, who wants to interact with another principal, say *Bob*, without revealing her real identity. *Bob* might be willing to accept this, but still needs to protect his own interests (e.g., to be able to prosecute *Alice* in the case of a dispute). To resolve these conflicting interests, both principals can agree on a mutually trusted principal T, which is usually called a *trusted third party* (TTP)[5]. *Alice* identifies and authenticates herself to T, which in turn issues a pseudonym P_A that it guarantees to reveal only under certain well-defined circumstances. Since *Bob* trusts T to reveal P_A if he convinces T of a problem, he will accept to interact with *Alice* using the pseudonym P_A.

This approach has the advantage that *Bob* does not learn *Alice*'s real identity and, since *Alice* can (and should) use a different pseudonym P'_A in interactions with another principal, say *Carol*, *Bob* and *Carol* will not be able to exchange data on *Alice* since both have different pseudonyms P_A and P'_A that only T and *Alice* can identify to belong to the same principal (the data of *Bob* and *Carol* on *Alice* is unlinkable). If P'_A was issued by a different TTP, say T', then linking P_A with P'_A would require a collusion of T and T'. Therefore, the approach removes many of the disadvantages that we have identified for anonymity in data avoidance, while retaining most of the advantages.

Other examples for the use of protocols and technologies that can be identified as implementing the concepts of multilateral security for privacy protection are, for instance, mixes, anonymous electronic cash, or fair exchange.

The concept of mixes was first presented by Chaum in [Cha81], where he presents the general idea of hiding the communication relation between a sender and a receiver in the context of electronic mail. This is usually referred to as untraceable communication and can be achieved with the help of a special principal, who acts as a TTP and operates the mix. This mix changes the order and the encoding of messages that pass through it in such a way that an external observer can not link incoming with outgoing messages. We will briefly return to this topic in Section 4.2.4. For more information please refer to [Pfi93, Cot95].

The concept of anonymous electronic cash is also due to Chaum, who presented it in [Cha83]. It allows to transform an electronic coin in such a way that the withdrawal and the deposit of the coin can not be linked with each other. Since a user has to identify and authenticate when he withdraws money from his account, but not when he spends the money (the identity of the receiver is sufficient), a purchase performed with such a coin can

[5] The integration of a mutually trusted third party is a recurring theme in the context of multilateral security (often simply in the role of a certification authority), which explains the usage of the term multilateral.

not be linked to his identity via the payment information. More comprehensive discussions can be found in [Cha85, PWP90].

The concept of fair exchange is described in [ASW96, BP90] and is concerned with protocols for the exchange of electronic goods. These protocols have to guarantee that at the end of the exchange, either each party has received what it expects or no party has received anything.

The generic definition of multilateral security opens it up to many other usages that are not specifically geared towards the protection of privacy, but can be applied to any security related problem that needs to resolve the disparate security requirements of several principals. The principals can identify and formulate their security (and privacy) requirements, which can be very different depending on the concerns of the individuals and often be in conflict with each other. This then necessitates a negotiation phase that identifies a mutually acceptable compromise. This negotiation can be accomplished in a static way, as was assumed in the pseudonym-based interaction, where *Alice* agreed to reveal her real identity to T and *Bob* accepts this arrangement since he trusts T to resolve problems, or result in a dynamic negotiation of a compromise. The latter approach is often inevitable since the security requirements of the interacting principals may sometimes not be known in advance. If no mutually acceptable compromise can be found, then this can also lead to a complete abort of the interaction. Finally, the requirements that were agreed upon in the compromise will then be respected by the implementation of the interaction.

An example for a not primarily privacy related usage of the concept of multilateral security is the reachability manager [DFRB97, RDFR97], which allows a callee to specify when and by whom he wants to be reachable. Among other possible requirements, it allows a callee who does not want to be disturbed by incoming phone calls to specify a deposit (payable in electronic cash) that a caller has to transfer before a connection will be established. If the callee decides that the interest of the caller was legitimate, he will refund the deposit. However, if he feels disturbed by the call, he can keep the deposit, which provides some compensation for his disturbance and might prevent the caller from repeating the interaction.

Many other examples for security and privacy measures in the context of multilateral security are discussed in [MP97].

1.4 Basic Principles for Privacy Protection

One of the foremost tasks that has to be accomplished for the conception of this thesis is to obtain a more pragmatic understanding of the abstract concept of privacy, which does not lend itself to direct technical approaches (i.e., it is difficult to identify technical measures that protect privacy in general). Since it is our explicit goal to address the problem of privacy protection on a technical level, we have to base it on more tractable principles. These principles should be easy to analyze in respect to how they can serve to protect privacy and it should be possible to address them with technical measures. This has resulted in the identification of the following three principles:

- notification, which states that a data subject should be notified about the flow and usage of data related to him;
- control, which states that a data subject should have some well-defined control over the flow and usage of data related to him; and

- trust, which states that a data subject should have reason to believe that the data related to him will be used fairly and that the principles of notification and control are actually observed.

The notion of fair use means that data is used to the advantage of both the controller as well as the data subject or in a way that is mandated by society. This aspect of trust is highly subjective and can not be addressed with technical measures. However, since the data subject has some limited control over the use of data, we would assume that he can withdraw the right to use the data when he discovers unfair use. In the following, we will use the term trust only in the second meaning, i.e., the data subject has reason to believe that the principles of notification and control are observed.

The three principles can immediately be mapped to Westin's definition of privacy. ("Privacy is the claim of individuals, groups and institutions to determine for themselves, when, how and to what extent information about them is communicated to others".) According to his definition, the data subject himself is in control of the flow of data, which obviously requires that he is notified about this flow of data. The definition further identifies some other entity as the designated receiver of the data, which the data subject implicitly has to trust to use the disclosed data fairly. Since, according to the definition, the data subject is still in control of the flow of data even though it has been disclosed to some other entity and the problem of information disclosure applies, the data subject must have reason to believe that this other entity will not circumvent his control.

The principles cover in addition also the usage of data, which Westin's definition ignores. Therefore, they allow us to define a stronger protection of privacy. On the other hand, they are also weaker than Westin's definition, since he does not foresee any restriction to the control. We could strengthen the principles by requiring that the data subject has complete control over the flow and usage of data, but this would be counter to the limitations of achievable privacy, discussed in Sections 1.2.3 and 1.2.4.

This set of principles is not necessarily complete and it is certainly possible to identify others that are not covered by the ones mentioned above, but that can nevertheless serve to protect privacy. However, it should be clear that a proper implementation of the ones we identified, would lead to a considerable empowerment of the data subjects. They would be better informed about who uses data related to them for what purposes and would be in a much stronger position to negotiate a fair use of this data. We will now discuss the principles in more detail, in particular their relevance for privacy protection and how they can be addressed with technical measures.

1.4.1 Notification

This principle is based on the notion that the more knowledge a data subject has on what data on him is available, who can access it, and who actually uses it, the better he can decide on possible actions to actively protect his privacy. A notification on the flow of data is identified in the sixth principle of fair information practices as stated in the Privacy Act of 1974 [USC74] as well as in Art. 10 and 11 of the EU-Directive on Data Protection [EC95].

A naive implementation of this principle would require a controller to send a notification to the data subject every time it accesses, changes, or discloses data on the data subject. The major problem with this approach is information overload, where the data subject will simply not be capable to process the large number of notifications, and in particular to extract those that signal irregularities. This problem can be addressed by aggregating and filtering the notifications, so that the data subject can, for instance, define that he only wants to be notified of important changes in the use of data related to him. In the ideal case, the aggregating and filtering would be done at the point where the notifications are created in order to reduce the communication overhead.

The requirements of this principle may be too strong for certain data accesses. It is possible to weaken them in some well-defined circumstances, where the notification of the data subject may be delayed. This could, for instance, be necessary in the context of law enforcement, where a suspect should not immediately be informed about the opening of an investigation on him. Such an operation would require the order of a judge and would deviate the notifications about the accesses in the context of the investigation to a third party that is trusted by the law enforcement agencies, but also by the data subject. This third party collects the notifications and forwards them to the data subject after the investigation is terminated.

1.4.2 Control

This principle is based on the notion that the more control a data subject has on the storage, disclosure, and processing of data related to him, the better he can initiate actions to actively protect his privacy. This inclusive interpretation of continued control over access to information can also be found in [Sch84b], where Schoemann defines that "a person has privacy to the extent that others have limited access to information about him". A much weaker form of control is identified in the second principle of fair information practices as stated in the Privacy Act of 1974 [USC74] as well as in Art. 12 and 14 of the EU-Directive on Data Protection [EC95].

The basic means of control that a data subject almost always has, is to decide whether he wants to disclose some data at all (see Section 1.3.1). However, there are many examples of interactions where this decision is severely limited, for instance in the case of data that is required to facilitate an interaction (e.g., a postal address for a mail order delivery). If not disclosing this data is the only means of control that the data subject can exert, then he can only refrain from the interaction altogether, which might be a too severe limitation.

Once the data is disclosed, a naive implementation of this principle would require the controller to query the data subject every time it accesses, changes, or discloses data related to the data subject. Actively exerting this control would completely overwhelm the data subject and prevent any effective data processing due to the additional latency on every data access. This problem can be addressed by annotating each data item with a usage and disclosure policy[6], which should automatically be derived from preferences of the data subject at the time when the data is originally disclosed. The idea of such a policy is closely related to the security requirements that were identified in the context of multilateral security. In the ideal case, the policy would be enforced at the point where the data is located in order to reduce the latency of data access by eliminating the otherwise necessary communication.

Again, the imposed requirements of this principle may be too strong and might have to be moderated according to the constraints identified in Section 1.2.3. This can be achieved by not accepting data that has been annotated with too restrictive policies or by simply ignoring

[6]This poses the additional problem of defining the policy. Since we assume that data subjects do not have great expertise, there must be some mechanism to contain the complexity of this task.

overrestrictive policies. This last measure should be a rare exception and (inevitably) result in a (possibly delayed) notification of the data subject.

1.4.3 Trust

This principle is not directly concerned with the management of data, but provides a foundation for the principles of notification and control. It is based on the notion that the more trust a data subject can have in the correct operation of the overall system, the more he has reason to believe that he will actually be notified of any access, change, or disclosure of his data. Furthermore, due to the fundamental problem of information disclosure, the data subject can not exert any direct control over the data and must have reason to believe that the control he exerts will actually be put into practice by the controller. Therefore trust represents the data subject's view on the problem of enforcement. The principle of trust is implicit, though not always justified [Gel97], in any legal approach (such as the Privacy Act or the EU-Directive on Data Protection) where it is based on the trust people have in the legal system.

Trust is itself a rather abstract concept and it is not obvious how it can be addressed with technical measures, which was one of the prerequisites that we identified for the basic principles. We have investigated this issue and are convinced that a technical approach that relies on legal support can better satisfy the principle of trust than a legal approach alone. In the following section we will present two different approaches to trust and then, in Section 1.6.2, we will explore how trust can be motivated based on technical measures.

1.5 Two Approaches to Trust

The concept of trust has long been recognized as being of paramount importance for the development of secure systems. For instance, any conceivable system for authenticating users needs trusted functionality that holds the necessary authentication information (see e.g., [SNS88, WL92]). Other examples are the trust one has in the proper operation of a TTP (e.g., a certification authority) or in the creator of a shrink-wrapped program. A pragmatic approach to address the notion of trust can be found in [BFL96], where the authors identify the domain of *trust management* as an important problem to provide security in network services. However, the meaning that is associated with trust or the notion of a trusted principal is hardly ever clearly defined and a user is left with his intuition.

A reason for this lack of a clear definition could be that trust is rather a social than a technical issue, which mixes the goals of a principal with its behaviour to achieve these goals. This means that in order to trust some principal, it is necessary to approve of its goals (which are not always clearly stated) and to believe that it will behave accordingly.

1.5.1 The Policy

In our definition of trust, we identify the central role of a policy, which is a set of rules that prescribes the behaviour of a principal for all relevant situations. This policy has to be written down, signed by the issuer[7], and made available to all other principals that interact with its issuer. Since the issuer is supposed to observe this policy, it is also in its interest that the policy is consistent with its goals.

[7]This can include a timestamp on the policy or even its certification by a notary.

With the help of this policy, the notion of trust can now be separated into two components: *adequacy* of the policy and *trustworthiness* of the issuer of the policy. In order for a principal, say *Alice*, to trust another principal, say *Bob*, *Alice* first has to verify *Bob*'s policy and must decide if it adequately protects *Alice*'s interests. This can be quite simple if the policy states "we will protect your interests" (which would in turn require a very high degree of trustworthiness from its issuer). In general, though, this is a difficult problem that can be addressed with various methods, for instance, similar to the approach for security architectures in [Rue91]. A very restrictive approach could consist of the definition of several static protection classes that can easily be assessed by *Alice* (e.g., the three classes proposed by TRUSTe [Dys97]). Another, more flexible solution is to define a formal specification of a policy and an ordering that allows us to compare two arbitrary policies to decide if one is stronger than the other. This forms the mathematical structure of a partial order. If both the specification of the policy and the ordering relation are well chosen, then *Alice* can assess *Bob*'s policy by identifying a minimal policy, which guarantees that all of *Alice*'s requirements will be met, and then comparing it with *Bob*'s policy. If *Bob*'s policy is stronger than *Alice*'s policy, then *Alice* knows that *Bob*'s policy is adequate for her (see Figure 1.1). We will return to this issue in Appendix B, where we address this problem in a particular example and introduce a simple specification of a policy together with a suitable comparison operation.

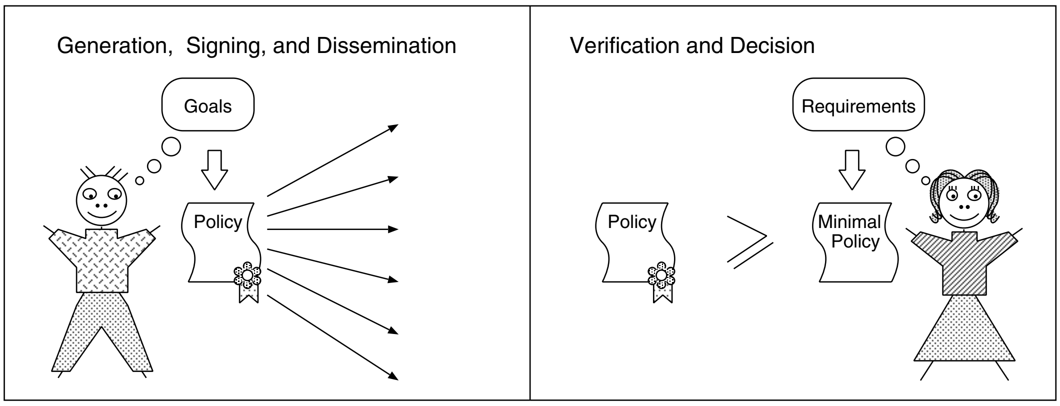
Generation, Signing, and Dissemination
Goals
Policy
Verification and Decision
Requirements
Policy
Minimal Policy

Figure 1.1: Assessing the adequacy of a policy

Then, in a second step, *Alice* has to assess *Bob*'s trustworthiness, which requires to establish a motivation for the belief that *Bob* will adhere to his published policy. This problem is quite difficult to formalize. Depending on how the trustworthiness of a principal is established, we have identified two fundamentally different approaches to trust[8]:

- the optimistic approach and
- the pessimistic approach.

In the optimistic approach, we give a principal the benefit of the doubt, assume that it will behave properly, and sanction any violation of the published policy afterwards. In the pessimistic approach we prevent any violation of the published policy in advance by effectively constraining the possible actions of a principal to those conforming to the policy.

[8]The approaches were named according to similar concepts in the domain of transaction processing.

1.5.2 The Optimistic Approach

This approach is easy to implement, since it does not require any special measures to make trusted interaction possible, which is probably the reason why it is the basis for most business conducted today. On the other hand, it requires a reliable mechanism to discover a policy violation after it has occurred. If such a mechanism does not exist, then there is no particular motivation to believe that a principal will adhere to its published policy other than its own assertion. This extreme case of the optimistic approach, which would require that one has *blind trust* in the other principal, is obviously a very weak foundation for trust and not recommended for any important interaction. It is therefore important to make the probability that a policy violation is discovered as high as possible by establishing an effective supervision of the principal's behaviour.

Once a policy violation is discovered, its effect should be compensated and, if it can irrefutably be attributed to one of the participants in the corresponding interaction, the violator should be sanctioned according to the damage caused by the policy violation. The primary goal of this sanction is to deter potential violators from committing a policy violation in the first place.

By definition, the optimistic approach cannot prevent malicious behaviour, but it only tries to compensate for it after it has been discovered. For many situations in real-life, where a policy violation can have an irreparable effect (e.g., the public disclosure of a confidential information cannot be undone) or where a proper functioning of the system is absolutely essential, this guarantee may not be strong enough. These are problems of our every-day life and therefore quite well understood — the question is if we can do better than that.

1.5.3 The Pessimistic Approach

The pessimistic approach removes these disadvantages by simply preventing any violation of the published policy. Thus, it prevents malicious behaviour in advance rather than correcting it after it has occurred. This would clearly be the best foundation for trust since we can solely rely on a principal's policy to verify that its behaviour will be adequate. 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 a rule of the policy prescribes a particular action for some event and if any violation of this policy is prevented then it is guaranteed that the action will take place.

The disadvantage of the pessimistic approach is that it is difficult and often expensive to implement, since it requires the deployment of a comprehensive infrastructure that guarantees the enforcement of the policies. It cannot be realized in its full generality, but is limited to those rules of a policy that can effectively be enforced with some uncircumventable mechanism; for non-enforceable rules, we still have to rely on optimistic approaches to trust. Finally, the pessimistic approach may not always be capable to unconditionally prevent policy violations, but is limited to making them more difficult to commit and easier to detect. Despite all these disadvantages, we believe that it still constitutes an improvement over the optimistic approach, since it allows us to reduce the number of possible violations as well as the number of possible violators.

1.6 Enforcement of Privacy Protection

We now want to discuss how the various approaches to privacy protection can actually be enforced and, in particular, how technical measures can support the enforcement of privacy protection. This discussion is organized according to the two approaches to trust, which can be interpreted as the data subject's point of view on the underlying enforcement mechanism (i.e., how the enforcement is perceived by the user).

The laws and guidelines that we identified in Section 1.3.2 as well as the security requirements of Section 1.3.3 can be interpreted as the policy for each of the two approaches to trust. Therefore, the approach used to specify the privacy protection is independent from the actual enforcement.

1.6.1 Enforcement in the Optimistic Approach

As we have already explained, this requires the cooperation of three components:

- supervision of a principal's behaviour,
- compensation for policy violations (if possible), and
- sanctioning of the violator.

Since the compensation is inherently dependent on the actual policy violation and not always possible, we do not want to comment on it any further.

An effective means to supervise a principal's behaviour is to require a high degree of transparency for all its actions, so that everybody who interacts with the principal can supervise its behaviour in respect to its policy. However, this is difficult to achieve and it is quite likely that many principals might not be eager to accept it in order to protect internal business processes. A better approach is to designate specialized appraisal companies that execute frequent in-depth controls of the conduct of companies, which is precisely the idea of TRUSTe [Dys97] (see Section 1.3.2). If a principal does not agree to voluntary audits, this approach provides no help. In such a case it is at most possible to supervise the externally visible actions of a principal and to start legal actions in the case of a suspicion. The problem of appropriate supervision is also highly dependent on the actual interaction and can not be addressed in its generality.

Convincing means to sanction a policy violation can be quite difficult to find. Depending on how such sanctions are enacted, we can break this issue further down into:

- implicit sanctions based on (a good) reputation and
- explicit sanctions based on punishment.

Implicit sanctions rely on 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. Since the policy is neutral in the sense that it allows all parties to benefit from the interaction, a policy violation can only result in a temporarily increased gain, but once it is discovered, would lead to a long term loss. 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 expensive to build up and that is invaluable for any company. Thus, we assume that a principal would not risk to lose its good reputation for a relatively small and temporary gain and will consequently rather adhere to its policy. This deterrent can obviously be increased by additional explicit sanctions (explained below).

This form of sanctions can be very effective since it puts immediate pressure on a principal who is under suspicion of improper behaviour – the market can react immediately. Furthermore, it is not dependent on national laws and thus more effective in international relations. The major problems are that it is only effective if a principal actually does have a good reputation that it is interested to keep and it can not effectively cope with malicious insiders. These can commit policy violations that are against the interest of their company for strictly personal reasons or financial benefits [Tim96, Win96].

Explicit sanctions based on punishment are the classical approach to enforce a certain behaviour. They rely on a comprehensive legal framework such as those discussed in Section 1.3.2. The goal is to create an environment where mutual beneficial behaviour is rewarded over anti-social behaviour by imposing disciplinary actions[9] such as fines or imprisonment (depending on the severity of the offence). These are supposed to negate the short term gain that may be achieved through a policy violation, which introduces a similar tradeoff as in sanctions based on reputation. However, the legal threat that deters most people from committing a policy violation may not be sufficient for a person who may readily risk some years of imprisonment for the possibility of a relatively large gain.

The non-compliance with a policy can be quite difficult to prove. This is further aggravated by many other well-known problems of laws, which are in general difficult to define, to understand, and to apply. Moreover, in the case of a particular dispute their enforcement is usually expensive, slow, and complex (in particular if the laws of different countries are applicable as can be expected for transactions on the Internet). Finally, laws require that the damaged party is aware that the legal protection exists and can only be applied after the problem has occurred, when the damage is already done. These problems have led to the institution of a data commissioner (also called data protection ombudsman), who is an expert in questions of legal enforcement of privacy protection and who can start investigations on behalf of an individual or autonomously.

1.6.2 Enforcement in the Pessimistic Approach

Due to the considerable problems of the optimistic approach, we want to devise technical measures that can support the direct enforcement of a policy. This should give legislators technical tools to accomplish their task more efficiently.

The idea is to legally mandate the use of technical measures that are assumed to prevent any violation of the policy as long as they are used properly. The supervision of a principal's behaviour can then be replaced (or, if a stronger protection is required, supplemented) with a supervision of the proper usage of the technical measures[10]. If these technical measures are constructed such that an improper usage is easy to detect and prove, then we assume that this supervision is more tractable than the supervision of the principal's behaviour. In the

[9]In the case of voluntary guidelines, which do not have the power of actual laws, this might require the establishment of a contract, which binds a principal also legally to its issued guidelines.

[10]It may be possible to conceive elaborate protocols that make an improper usage infeasible, which would make the supervision unnecessary. They could, for instance, rely on mechanisms such as contract signing without a third party [EGL85]. These protocols would be trivial from the point of view of enforcement, though highly complex in their realization. We have not investigated this approach.

If an improper usage of the technical measures were infeasible, which is a purely hypothetical assumption since no technical measures can guarantee total safety, then the supervision would be no longer necessary.

case when an improper use is detected, the approach resorts to the same means for sanctions explained above (and, if possible, compensation).

The advantages of this approach are that a policy violation becomes easier to discover and more difficult to commit[11]. A pessimistic approach that relies on well-constructed technical measures can, thus, provide a better privacy protection for the data subjects than an optimistic approach alone, since it can be pursued in addition to everything that is done in the optimistic approach. However, if the technical measures are not adequate, they provide a false security, which results in less supervision and possibly more undiscovered policy violations.

The technical measures have to be deployed such that they are not accessible to the issuer of the policy, which requires that they are encapsulated in some physically protected domain[12]. This domain can either be the administrative domain of a trusted third party (TTP) or a trusted and tamper-resistant hardware device that can safely be operated in the administrative domain of an untrusted entity. Such a trusted and tamper-resistant hardware device is physically protected from its environment and interacts with its environment only through an interface it controls. It can be a simple chip-card (also called smart card), a complex PC-card (PCI or PCMCIA), or even a dedicated computer system with its own peripheral devices. The reason why the TTP or the trusted and tamper-resistant hardware device should be trusted more than the issuer of the policy can, for instance, be based on the fact that the former enjoy a good reputation, while the issuer of the policy is not well-known. This issue requires a more profound explanation, which will be discussed in Section 3.5.1 for the case of the trusted and tamper-resistant hardware (the arguments for a TTP would be quite similar).

From a theoretical point of view, both methods are very similar and can be used to achieve many of the same goals (the trusted and tamper-resistant hardware device can be seen as a miniature TTP). However, from an engineering point of view they are quite different, since they provide almost complementary system properties. For instance, a TTP disposes of more resources (e.g., computational power or storage) and has control over its access to the communication facilities (i.e., no other entity can permanently block its connectivity). On the other hand, a trusted and tamper-resistant hardware device provides more efficient connectivity to its operator (bandwidth and latency are only limited by the local installation), offers a more predictable performance (it is not shared with other principals as is usually the case with a TTP), and will not temporarily be inaccessible due to communication problems that are not under the operator's control.

We already mentioned that the pessimistic approach can not guarantee the enforcement of arbitrary policies (e.g., it can not force the issuer of the policy to execute a particular action or prevent it from storing a value), but is restricted to the enforcement of very limited (but nevertheless useful) policies. Therefore, the policy has to be investigated by an impartial principal (either the TTP or the manufacturer of the trusted and tamper-resistant hardware device), to decide whether it can be enforced. The impartial principal will then guarantee this by certifying (digitally signing) the policy. During normal operation, the issuer of the

[11]A potential violator who wants to overcome the technical measures needs more knowledge, determination, and has to make a greater investment. This can easily be translated in more criminal energy and thus be punished more severely, which results in an additional deterrent to commit the policy violation (stealing an item from a closed car as opposed to an open car requires more determination and special knowledge on how to open a car).

[12]We will later discuss the approach by Sander and Tschudin [ST98], which allows to provide some limited protection to a mobile agent without requiring them to be encapsulated in a physically protected domain.

policy is required to cooperate with the TTP or the trusted and tamper-resistant hardware device, which will ensure that no policy violation can occur.

Based on the different properties of the two methods and for reasons that should become clear later on, we have chosen to pursue the one based on a trusted and tamper-resistant hardware device. The use of such a device constitutes an application of the general concept of transferring trust in a tamper-resistant device over to the untrusted operator of this device (e.g., a sealed electricity meter in a private home or the balance of an official buyer (of goods) calibrated and sealed by an appraisal organization). We will describe a trusted and tamper-resistant hardware device in Chapter 3 and discuss the requirements that have to be met in order to enforce certain policies in Chapter 4. The tamper-resistant hardware device that we will explore is not restricted in its use to address privacy related problems only, but can be used in other contexts. We will explore several of them before returning to the problem of privacy protection in Chapter 5.

1.6.3 A Similar Approach

A specific application that uses very similar ideas with respect to trusted hardware is the one discussed by Brands in [Bra94]. It uses a wallet with an observer for the implementation of an anonymous off-line electronic cash system. The problem with such a system is that an electronic coin, which is simply a digital information with a corresponding digital signature of a bank, can easily be copied and, thus, spent several times at different receivers (this is called double spending). Since the system is anonymous (i.e., the withdrawal and the deposit of a coin can not be linked with each other), a bank can not identify the offender.

Here, the observer is the trusted and tamper-resistant hardware device that operates in a wallet that is physically controlled by a user. Both the observer and the wallet do not trust each other. The observer participates in every withdrawal transaction, and has to authorize every payment transaction; so it knows what coins the user received from the bank and it authorizes every spending of an electronic coin at most once. Thus, its task is to implement a prior restraint on the double spending of electronic coins.

Since this pessimistic approach is entirely dependent on the tamper-resistance of the observer, the approach integrates another optimistic mechanism that provides an additional safety in case the tamper-resistance of the observer is broken. This mechanism allows to easily detect coins that were spent more than once and to deanonymize the principal who committed the double spending. This entire approach has actually been implemented in the ESPRIT Project CAFE [BBC$^+$94]. For more information on the underlying technology, please refer to [CFN88, Cha92, PWP90].

This approach is significant since it relies on a similar assumption of multiple lines of defence that we also assume for the trusted and tamper-resistance hardware device. The device forms a first, potent barrier that eliminates all occasional offenders that do not have the knowledge or the resources to effectively attack the system. Among the remaining, sufficiently potent principals, there is only a small fraction that can realistically expect a higher profit from attacking the system than by using it according to its specification. This remaining risk is then addressed with appropriate mechanisms that allow to discover an attacker. Such an attacker can then be punished for a successful attack on a legally protected system. This punishment can be significantly higher than that for proven double spending (since it constitutes a systematic attack instead of a single proven offence).

1.6.4 An Overview of our Approach

The idea of our approach for technical privacy protection is to encapsulate every data item on individuals within an object that implements the basic principles of notification and control. This means that the object sends a notification to the data subject whenever the data in the object is accessed and enforces the access control policy that was provided by the data subject. In order to accomplish these tasks, it has to be protected from its controller. More precisely, the data object, which consists of code and state, has to be protected against illegitimate access on its data and from any interference with the execution of its code.

We can interpret the object that encapsulates the data on individuals as a mobile agent (see Chapter 2), and will also refer to it as *data agent*. Therefore, the identified problem of protecting the data agent from its controller can be mapped to the problem of protecting a mobile agent from its executor. The latter problem will be addressed in Chapter 3, where we present an approach that is based on a trusted and tamper-resistant hardware device, which can protect the code and state of a mobile agent from disclosure and manipulation. These mechanisms that protect a mobile agent when executing at a remote location can also be applied to the data agent. They can prevent any illegitimate access to its data or interference with the execution of its code and allow the data agent to enforce its access control policy. The problem of preventing that an access notification is suppressed by the controller can be resolved by delaying any access until the data agent has received an acknowledgement from the data subject, stating that the access notification has been received. This may have to be cryptographically protected to prevent that the acknowledgement is forged by the controller.

Furthermore, the approach can prevent the unauthorized transfer of a data agent to another controller. The decision about a transfer is left to the data agent, which can decide based on its policy or relay the decision to the data subject. This allows the data subject to either indirectly (via the policy) or directly control the flow of data.

The presented approach can be identified as a generic application of the concept of multilateral security discussed in Section 1.3.3. The policy allows the data subject to express his security requirements within the limits of enforceable policies, which are then implemented by the data agent and enforced with the help of the trusted and tamper-resistant hardware device. This allows to specify complex and dynamic security requirements of different data subjects on top of the same generic enforcement method.

1.7 Relevance of Privacy

Numerous recent studies have shown that the concern for the protection of privacy that we have discussed in this Chapter is not only theoretically relevant, but a real issue for users on the Internet.

The 8th GVU WWW User Survey [GVU97] that was conducted internationally identified privacy (30.49%) as the most important issue facing the Internet, followed by censorship (24.18%), and navigation (16.65%). In the same study 39% (33%) of the respondents agreed strongly (somewhat) that there should be new laws to protect privacy on the Internet. This last figure was confirmed in a poll by Business Week [Ham98], which found that 53% of the respondents wanted the government to “pass laws now for how personal information can be collected and used on the Internet”. According to the same poll, 61% of poll respondents say they would use the Internet more if their privacy would be protected and 43% (42%) of the respondents say that privacy policies posted on a company’s web site help not at all (a little) to encourage them to purchase products or services from that company. This has to be seen in the context of 33% of the respondents who do not trust the company at all to actually follow its policy, while 58% trust the company at least somewhat.

This indicates that people may actually be willing to pay for privacy. This can in turn be used by companies, who succeed in convincing users that they will protect their privacy, to increase their profit. This is a market that is currently being explored by TRUSTe, but also by big accounting firms such as Coopers & Lybrand and KPMG [Dys97].

∞∞∞

Summary

This chapter defines the concept of privacy and establishes the importance of protecting it. Various current approaches to privacy protection are discussed. We present the approach that we propose to pursue, which is based on the principles of notification, control, and trust. The principle of trust is investigated in greater detail. This leads to the identification of two approaches to trust, which can be interpreted as the data subject's point of view on the underlying enforcement mechanisms. We identify the pessimistic approach to trust that is supported by technical measures as the one we want to pursue. The chapter terminates with an overview of the approach that is developed in the remainder of the thesis.

Chapter 2

System Model and Mobile Agents

And trust no agent; for beauty is a witch
Against whose charms faith melteth into blood.

– **W. Shakespeare** *(deliberately taken out of context)*

2.1 Introduction

The term *agent* is probably one of the most overused words in computer science and has very different meanings in the area of artificial intelligence, network management, or distributed systems. It is therefore very difficult, if not impossible, to give a widely accepted definition for what constitutes an agent. Adding the adjective *mobile* allows us to more clearly center the corresponding research area on distributed systems, where *mobile agents* are considered as an interesting and innovative approach to structure distributed applications [HCK97].

Since we are, in the context of this thesis, primarily interested in the protection of mobile agents and the use of these protected agents for privacy protection, we do not consider this lack of focus as a disadvantage. It is one of our explicit goals to keep the approach of mobile agent protection as general as possible so that it can be applied to most systems that fall into the category of mobile agent systems. Therefore, we will keep the following discussion of what we consider to be a mobile agent system quite broad and simply identify the basic properties and requirements of mobile agent systems that we have to assume in order to apply our approach. This is the subject of this chapter, which will explore various technical issues in the context of mobile agents in order to identify these basic properties and requirements.

2.2 The World of Mobile Agents

In order to prepare for the following detailed discussion of the possible features and paradigms that a particular mobile agent system can implement or adhere to, we first want to discuss the environment in which we envisage mobile agents to thrive and then provide a rather comprehensive example of what an agent can do. The example is supposed to illustrate some of the less technical issues in a mobile agent system. These are the autonomy of the mobile agent, which does not need to ask the permission of the user for standard decisions, and the fact that a mobile agent executes a task on behalf of a user, from whom it also obtains its authority. It is these issues that justify the use of the term mobile agent instead of the more technical mobile object.

2.2.1 The Environment for Mobile Agents

The underlying infrastructure that we envision as the environment for mobile agents is very similar to what currently exists in the Internet and in the World Wide Web (WWW). However, the Internet is not the only existing infrastructure in which mobile agents can subsist. The various interconnected national telecommunication networks also provide sufficiently powerful computing technology to serve as an environment for mobile agents and the network's operators are more ready to accept new ideas from the area of distributed computing and computer networking. As a matter of fact, it is not obvious to separate both infrastructures from each other since the Internet is in part realized on top of the telecommunication networks and conversely there are considerable efforts to provide classical telecommunication services over the Internet, such as classical voice communication in the form of IP-telephony [Sch97b].

This indicates that there is a powerful trend for convergence between the Internet with its IP-based protocols and more traditional telecommunication networks. Examples for this convergence are the adoption of ideas from the Internet for the provision of services in future telecommunication networks as illustrated in the TINA effort [DNI95] and the increasing interconnection of services via gateways as can be seen in the possibility to send SMS messages to GSM compliant mobile phones [MP92] from the Internet (SMS Gateway). Another important and very well established gateway are the Internet Service Providers (ISP), which allow to access the Internet via conventional telecommunication facilities. We therefore assume that future computing devices will be able to take advantage of any available communication links, which will all provide a connection to the Internet and, thus, to all possible service providers on the Internet:

- wireless telecommunication networks based on infrared, radio, or satellite,
- fixed telecommunication networks, and
- IP or ATM networks.

Another major trend can be observed in the increased mobility of end-users, who want to have access to communication and computing services at any time, at any place, and in any form. This can be confirmed in the rapidly increasing number of subscribers to mobile communication services, such as GSM [MP92], IS-41, IS-95, or JDC (Japanese Digital Cellular) [Lor93], but also in the more recent success of very small computing devices that are often called *personal digital assistants* (PDA), such as 3Com's PalmPilot or palm-size devices based on Windows CE. The merge of these end-user equipment will lead to completely new communication and computing devices that will be both a mobile phone and a PDA. Recent products such as Qualcomm's pdQ [Qua], which is a *PDA phone*, clearly point in this direction.

Concerning the computing hardware, also called hosts, in the infrastructure for mobile agent systems that serve as the platforms for agent execution, we can identify two different roles that these hosts can play. First, the role of a *client*, which allows a user to interact with the mobile agent (possibly via some special user interface software that is not directly part of the agent). The mobile agent will receive the necessary configuration information required to accomplish its task during the interaction with the user and deliver the final results of its task back to the user. If the configuration information with which it was launched is insufficient or if particular problems occur during its execution, the agent may also briefly return to the user during the execution of some task to receive further instructions. Second, a host can play the role of a *server* to which the mobile agent is sent in order to accomplish its task. Here, the mobile agent can interact with other agents that are located at this platform and can access resources that are available from this platform. Often it will be necessary for a mobile agent to interact with many servers to accomplish its task (e.g., TTP, broker/trader, or name server) or to even visit several of them to interact directly with local resources.

Other important factors for a host are its capabilities (i.e., available resources and connectivity) and, in particular for a host in the client role, its physical form. For a host in the server role we assume that it will have access to the fixed network, very high bandwidth communication facilities, permanent connectivity, and huge processing and storage resources. It will directly interact with end-users only for management purposes, therefore the user interface capabilities are not of immediate importance for its functionality. There will be many different actual systems, but we assume that most of them will satisfy the above requirements.

For a host in the client role we can distinguish between three different types of hardware:

- stationary clients
- portable clients
- mobile clients

The stationary clients correspond to current home PCs. These are powerful machines with high bandwidth communication facilities, permanent connectivity, large processing and storage resources, and powerful user interaction capabilities (big screen and keyboard, sound, video, etc.). A stationary client will, in general, be used by many people (such as all the members of a family, as well as, occasionally, their friends and relatives) and in special cases it might even be used as information kiosk that provides information services to anonymous users on the expense of some sponsor.

The portable clients correspond to current laptop computers. These are mid-range machines with intermittent connectivity (that may frequently change from high to low bandwidth communication facilities), considerable processing and storage resources, and usable user interaction capabilities (small screen and keyboard). A portable client will be used by a small number of people.

Finally, the mobile clients correspond to PDAs, mobile phones, or PDA phones. These are very limited machines with low bandwidth and high cost communication facilities, intermittent connectivity, small processing and storage resources, and restricted user interaction capabilities (tiny display, pen-based or telephone button input technology). A mobile client will typically be used by a single person and can, thus, be highly personalized. Since a mobile client can easily communicate with a stationary client (e.g., infrared links), it can provide its personal preferences and configuration information in order to allow temporary personalization to take advantage of the stationary client's vastly superior communication, processing, and interaction capabilities. Furthermore, it can also take the role of a personal security device in cryptographic protocols (e.g., authentication or signatures).

The distinction between these different types of hardware may become blurred, for instance in the case of very powerful or very small portable computers. Nevertheless, each of them has a rather specific set of properties, that make them suitable for different tasks, and users will adapt their choice of services to the available hardware.

2.2.2 An Example

An often cited example (see e.g., [Yee99, FGS96a, CGH$^+$95]) in the domain of agent based electronic commerce, which addresses many important issues, is that of a person, say Alice, who wants to buy an airline ticket using the facilities provided by a mobile agent system. We assume that Alice wants to fly from Geneva, CH to Nagoya, JP. As there is no direct flight, available, she will be able to choose from a large number of options varying not only in price but also in flight times and possible stop-overs.

We assume that Alice has access to a mobile agent that she is familiar with and with which she had positive results in the past. Due to the continued interaction with Alice, the agent has created an information base that contains configuration data that Alice should not have to enter repeatedly (e.g., a phone number to inform her of last minute problems, or payment data) but also many of Alice's preferences, such as preferred seating, dietary restrictions, or other personal constraints Alice regularly expresses on air-travel. While she is riding on a train to her next business meeting, Alice launches the agent, or rather the program that provides the user interface for the configuration of the agent, on her PDA and provides it with the information for the desired trip: destination, travel dates and times, business or leisure, as well as other constraints that define when an offer from a service provider is acceptable.

After the agent is configured it will eventually leave Alice's PDA. This might be accomplished immediately after Alice has finished the configuration using a low-bandwidth, high-cost radio based network that is available in the train or (if this is not available or too costly) the agent might wait until Alice approaches a public network access point with a connection to the high-bandwidth and low-cost fiber based network. The connection between Alice's computer and the network access point could be an infrared or short-range wireless (e.g., bluetooth [Blu]) link, which only requires Alice to point her computer towards a receiver or simply stay in the coverage area of the network access point for a sufficient amount of time (depending on the size of the agent, a few seconds should be enough).

The agent has to migrate to some service provider[1] that offers a suitable agent execution environment[2] and that is both capable and willing to execute it. The capability to execute it depends primarily on the used technology and limits the agent to those service providers that support the technology for which it was conceived. This problem is far from being resolved and will be subject to future standardization efforts [OMG98b, FIP97], which will probably identify a small number of technologies that may be supported in parallel (even though not every service provider will support all the competing technologies). The willingness of the service provider to execute the agent depends on security related issues that we will discuss later, but also on business related issues dealing with payment for the resources consumed by the agent while executing at the service provider (e.g., CPU cycles, memory usage, and communication bandwidth consumption).

Depending on the agent's configuration it will probably first visit a broker that will provide

[1]We will later on see that the service provider is not necessarily the principal that operates the agent execution environment, but we do not make this distinction here.

[2]We want to use this term here in an intuitive way which simply signifies an engine located at some place that executes the mobile agent properly. It will be explained in more detail in the agent execution service discussed in Section 2.5.2.

it with a first selection of possible airlines (avoiding, for instance, non-optimal itineraries that require long detours). The result of this first interaction is a list of possible airlines or rather of the network addresses of their suitable agent execution environments.

The agent will decide on an itinerary and migrate to the first airline server on its itinerary. There it will negotiate with the services accessible on the execution environment in order to find a suitable offer that satisfies all (or at least a sufficient subset) of the constraints defined in Alice's preferences and during her configuration of the agent. The agent will request the offer to be put on hold and migrate to the next airline server on its itinerary.

When the agent can not find a suitable offer, it will return to Alice's PDA, inform her about the problem, and possibly propose several relaxations on her constraints based on its negotiations with the airline servers. Otherwise, when it has visited all the airlines servers on its itinerary or upon another suitable termination condition (e.g., a certain number of suitable offers found), the agent will execute a decision algorithm that will identify the best offer (the location where this decision algorithm is executed, may also be an important issue due to security considerations and since the service providers are probably eager to minimize their computational load). The agent will then migrate to the airline server where it obtained this offer and finalize the purchase transaction. As a result of the purchase, the agent will obtain a cryptographically protected ticket. Then the agent may inform the other service providers to release the offers it had requested to be put on hold. Finally, the agent will return to Alice's PDA, possibly waiting in some mailbox in case Alice's PDA is temporarily disconnected from the network, inform her about the decision taken, and deliver the ticket to Alice. If Alice discovers a problem with the arrangements done by her agent, she can not simply undo the actions, but she rather has to take additional compensatory actions to resolve the problem

We can see that this seemingly innocuous example of an e-commerce agent purchasing an airline ticket already produces a rather big scenario, in which we have not yet tackled many interesting issues. In the remainder of this chapter, we will explore some of the technical details that we have now just glanced over.

2.3 Mobile Code

A prerequisite for mobile agents is some technology for mobile code, which allows to send a piece of code over the network to be executed at a remote host. Due to the immense success of Java [GM96], which puts code mobility in the form of Java Applets [Fla97] in the center of its design, this topic has recently returned to the forefront of attention.

We will first have a look at the history of mobile code, then we will explore the underlying concepts, and finally discuss two simple paradigms that make use of mobile code.

2.3.1 Some History

Code mobility is not a new concept. It has its origins in the 1960s where the Job Control Language (JCL) [Bro77] was used in the optimization of networked computers, enabling mini-computers to submit batch jobs to mainframes using a remote entry system (RES) [CGH$^+$95]. Another direct approach to code mobility is the `rsh` (remote shell) command that was introduced by 4.2BSD UNIX in 1984 [LMKQ89]. It allows a user to send a shell script to a remote machine where it will be executed with access to all the local resources of the remote machine, such as peripherals (e.g., a printer) or data on the local disk of the remote machine.

A rather prominent and successful example for the use of mobile code is SQL, the *structured query language* [Lyn90], which was originally conceived in the late 1970s and first standardized in 1986. Even though the mobility of a particular query was not in the center of the design of SQL, but rather the declarative, high-level, application neutral formulation of a query, the compact representation of the interpreted language make it very easy to package a query into a message, send it to a DB server, where it is interpreted with complete access to all the resources of the DB server, i.e., the data in the DB. Thus, it can perform filtering on the large amounts of data in the DB and return the (usually) small result of the query to the user. An SQL query can be interpreted as a very specialized form of agent, which is primarily conceived to be executed close to the data, to filter the data, and to return a much reduced amount of data in the reply to the query.

A less obvious use of mobile code is illustrated in PostScript [Ado85], which is a *page description language* that is primarily used for describing the appearance of printed pages. A document is sent to a PostScript capable printer, which will execute the code, construct the described page, and print it. PostScript is actually a fully functional language that can be used for general purpose computing and served as inspiration for the M0 agent language that is used in the Messenger Agent Project [TMMH96].

In the domain of distributed operating systems the question of code mobility has been studied quite comprehensively in the context of process migration, which deals with the problem of moving a running system level process from one machine to another. This was mostly done with the goal to improve system efficiency by providing load balancing capabilities, but also to allow a dynamic reconfiguration of the system so that a process can be moved to another machine if its current host has some scheduled down-time. Examples for systems that provide process migration are Amoeba [MvRvRvS90], Sprite [DO91], or Condor [TL95]. A more fine-grained approach to similar problems is studied in the context of object migration, which enables the mobility of language level objects that can be relocated from one address space to another. The reason for this relocation are also efficiency concerns similar to those that motivated process migration, but also more subtle advantages, such as the migration of parameter objects to a remote site for the duration of an object invocation. Examples for systems that provide such facilities are Emerald [JLHB88] and Obliq [Car95]. An overview of systems providing process or object migration facilities is given in [Nut94].

A major obstacle for truly mobile code that can potentially be executed everywhere, is the heterogeneity of the possible execution environments (hardware, operating system, and installed software). This has led to the development of scripting languages that were conceived to overcome these problems, such as Perl [WCS96], Tcl [Ous94], or Python [Lut96]. These scripting languages are executed in the context of a runtime system that interprets the commands of the scripting language. Since the runtime system itself is software, it can be ported to any suitable hardware platform, which can then execute mobile code of the corresponding scripting language, thus providing a homogeneous execution environment for the programs coded in the scripting language. All of these scripting languages have to some extent been augmented with mobile code facilities (agentPerl [Pur98], agentTcl [Gra95], Rover [Rus98], Ara [PS97]).

The most recent contenders in the area of mobile code systems are Telescript[3] [Whi94] and the already mentioned Java. Telescript was from its conception designed as a language for mobile agents and explicitly geared towards its use in electronic-commerce. As a consequence

[3]Telescript has meanwhile been abandoned by General Magic in favor of Odyssey, which is based on Java.

it was right from the start designed with strong security features in place [Vigna[91]]. Java was originally conceived as an interpreted, architecture-neutral, portable language[4]. However, it was soon discovered that these properties made it into a suitable language for automatically downloadable programs on the World Wide Web. With the integration of the Java virtual machine into the Netscape Navigator in October 1995 it became the first widely deployed system for mobile code. A considerable number of systems try to leverage the code mobility of Java into a full-fledged mobile agent system, such as Aglets [LC96], Concordia [WPW+97], Jumping Beans [Ast98], Mole [SBH97b], Odyssey [Whi97, Mag], Voyager [Obj], WASP [FM99].

2.3.2 The Basic Concepts of Mobile Code

Given all of these very different approaches, we want to briefly summarize the defining characteristics that are relevant (important) for code mobility. These are (mainly) centered around the concepts of :

- location and
- migration.

These can also be found in [CPV97], where Carzaniga et al. informally define code mobility as “the capability to reconfigure dynamically, at run-time, the binding (this implies the migration) between the software components of the application and their physical location within a computer network”.

The basic idea as depicted in Figure 2.1 is to transfer a package that contains the code and possibly additional (data or) parameters from a sender’s site S_A to a receiver’s site S_B, where it is, eventually, executed. During this execution, the code may produce a result (which can be a message or even another piece of mobile code) that can be sent to another site or back to the original sender.

This indicates the necessity for two different but closely related technologies. The first one is necessary to instantiate the mobile code together with the data or parameters it requires on a given location and provides the execution environment for the mobile code. We will refer to this technology as the *platform*. This platform can be very simple as in the case of process migration, where it is identical to the actual hardware, or it can be rather sophisticated as in the case of a runtime system for scripting languages. We will return to this technology in Section 2.4.2, where we discuss the agent platform. The other technology is responsible for the transport of the agent to another location. It consists of a mechanism to marshal the mobile code into a transportable format that can be sent via some communication infrastructure to the new location – we will further discuss this issue in Section 2.2.1. Again, this technology can be quite simple as in the case of scripting languages, which usually require simply the sending of an ASCII-text file or it can be relatively complex as in the case of process migration, where the code segment, data segment, and execution state of the running process have to be extracted from the current location and properly packaged to allow its transmission to and rebuilding at the new location. As can be seen in this last example, where the mechanism that marshals the code requires extensive support from the platform in order to accomplish its task, it is not always easy to clearly separate the two technologies, since they are strongly dependent on each other.

[4]The complete list of buzzwords can be found in [GM96].

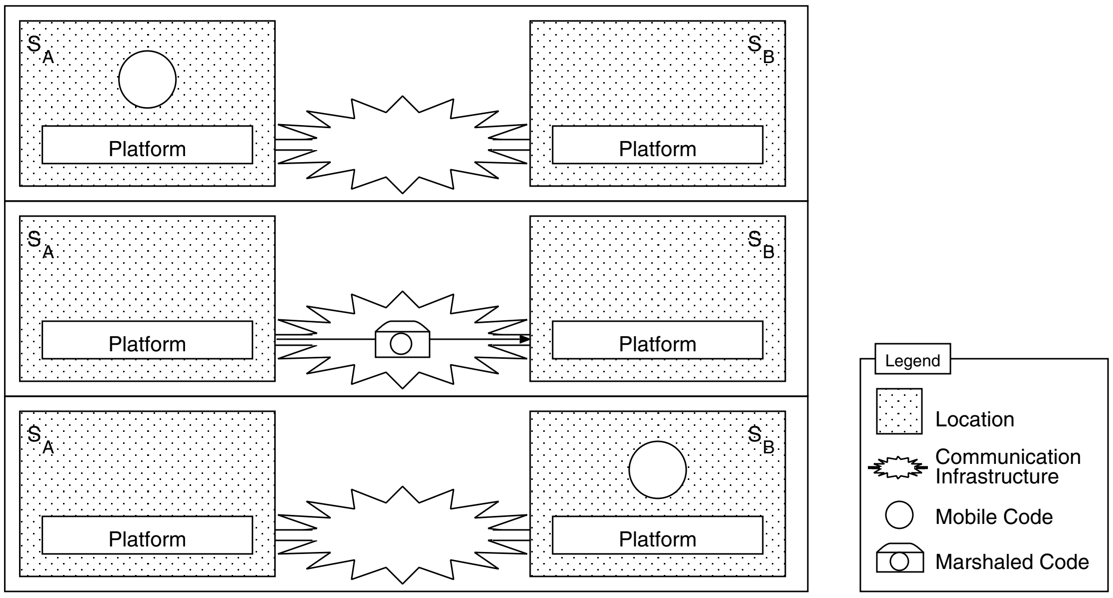

Figure 2.1: The basic idea of mobile code

2.3.3 Simple Paradigms for Mobile Code: REV and COD

The actual usage of mobile code in a real system requires to resolve the question how the migration of the mobile code is initiated. In the example given above, this can be either one of the two principals. This differentiation results in the identification of two different paradigms for mobile code [CPV97]:

- remote evaluation (REV) and
- code on demand (COD).

In the REV paradigm, it is the sender of the mobile code who takes the initiative to transfer the mobile code to the receiver. The idea of the REV paradigm is, for instance, that a principal can customize the basic low-level services of a provider by transferring some additional code that composes these low-level services into a high-level service. It is therefore a straightforward extension to the well-known RPC paradigm, which only allows to call the basic services of the provider. Without the REV paradigm, an RPC based service has to provide complex and difficult to manage high-level functions in order to save communication bandwidth and to reduce latency.

In the COD paradigm, it is the receiver of the mobile code, who takes the initiative to request the mobile code from the sender. The idea of the COD paradigm is that a computation running on a particular platform can download and link some code from a provider to perform a particular task. This can, for instance, be a user interface, that interacts with the user to seize a certain amount of data, or a computational component that is required to perform a particular task. This is exactly the paradigm that is naturally supported by Java applets [Fla97], which consist of a set of Java classes that can be referenced in an HTML document and are subsequently downloaded and executed when this document is accessed by a remote user.

The two paradigms are of high importance since they are easy to implement in real systems and, thus, more readily available for experimentation. Examples for a straightforward implementation of the COD paradigm are Java applets [Fla97] and ActiveX controls [Cha96]. Furthermore, in Java it is also easy to implement the REV paradigm based on the object serialization that is provided with the Java Remote Method Invocation (RMI) [Sun98]. To do this, it is sufficient to implement a simple server that accepts serialized Java objects and instantiates them locally.

The problems raised by mobile code and the services required to tackle these problems, in particular in the context of security are very similar to those of mobile agent systems, which are discussed in Section 2.5.

Many problems that we will discuss in the context of this thesis are not dependent on the full mobile agent paradigm, but could be addressed with one of the simpler paradigms (see, for instance, the stationary agent in Section 2.6.2). However, for ease and brevity of presentation we will concentrate on the more general mobile agent paradigm. The changes to our approach if applied to one of the other paradigms should be obvious and relatively minor.

2.4 The Mobile Agent Paradigm

In this discussion of the mobile agent paradigm, we are primarily concerned with the supporting infrastructure that provides the necessary services for mobile agents. This infrastructure, to which we will also refer to as *mobile agent system* is made up of several components, as depicted in Figure 2.2. The functionality of a mobile agent system is to allow the creation of a society of mobile agents and can be compared to that of a middleware for distributed applications. The goal of such a middleware is to provide the infrastructure that enables a federation of interacting objects. To achieve this, it has to specify and standardize services for the applications as well as the way how the applications interact with these services. There are many different approaches to implement such an infrastructure and it has taken the distributed systems community more than a decade before converging on CORBA [OMG95] and DCOM [Red97] as the prevailing architectures for middleware for distributed applications. We assume that it will probably take the mobile agent community a considerable amount of time before settling on a small number (ideally one) of valid architectures for mobile agent systems. It may be possible to learn from the experience made in the design of CORBA to shorten this time and there are even efforts that try to integrate a mobile agent system into CORBA [Vin98, OMG98b, Obj].

As we have already mentioned in the introduction of this chapter, we do not want to focus on a mobile agent system in particular, but we are concerned with the protection of mobile agents in general. Therefore, we will not describe a single mobile agent system, but rather identify the issues that have to be resolved for our approach to be applicable and point out possible design alternatives.

In the example of Section 2.2.2, we have described a mobile agent system from the viewpoint of the different application level services, such as the server of an airline operator or the broker. We will now analyze this system from the viewpoint of a lower abstraction layer and look at the components that make up the the infrastructure and, in the following Section 2.5, at the services that have to be provided by the mobile agent system. In Figure 2.2 we have identified five different components: the mobile agents, the agent platform, the operating system, the hardware, and the communication infrastructure. For the sake of the discussion

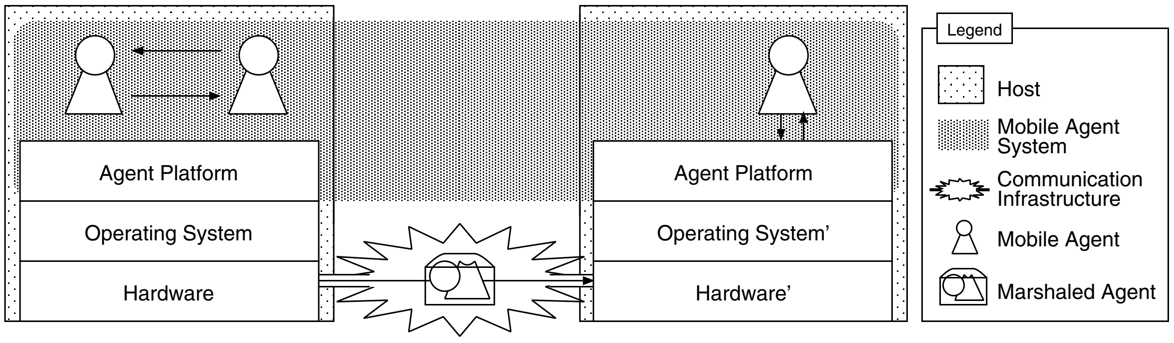

Figure 2.2: The Infrastructure for Mobile Agents

of the services, we want to concentrate on the two components that are specific for a mobile agent system:

- the mobile agent and
- the agent platform.

We assume that the remaining components provide the well-understood services of a modern operating system running on a contemporary computer and the connectivity to an extensively deployed wide area communication infrastructure, such as the Internet. We will not comment any further on how these components generate their services and refer to the appropriate literature, e.g., [SG94, Tan90, Tan88].

2.4.1 The Mobile Agent

In the mobile agent paradigm, the mobile agent itself obviously plays a central role. It is the entity that migrates between the agent platforms, in order to accomplish its task. Since there is no real consensus on what a mobile agent precisely is, we introduce the following definition for our work, which identifies the generic concepts that mobile agents exhibit in most existing mobile agent systems. A *mobile agent* is a piece of software that has the following three properties:

- it is an object (or set of objects) that consists of code and state (i.e., it is active and has an execution thread (or several threads) of its own),
- it can (autonomously) migrate from one agent platform to another where it will interact with local objects, data, or facilities, and
- it executes a well-defined task on behalf of a user from whom it also obtains its authority and tries to accomplish this task without further intervention from the user (unless unexpected events occur).

Thus, the important properties for a mobile agent in the context of the mobile agent paradigm are that it is an active object that moves in order to do something *for* a user. Other properties that are often associated with agents, such as intelligence, learning, or collaboration are not necessary requirements for what we consider to be a mobile agent (even though an agent may exhibit them).

On a more general level, we assume that a mobile agent is implemented with some object-oriented methodology, which allows us to encapsulate its state (other than the current execution state that is managed by the execution environment, see Section 2.5.2) within the data structures of the object. This state is then manipulated exclusively via the methods of the object, of which some are purely internal and not accessible from the outside, while others are explicitly made accessible and constitute the public interface of the mobile agent. Furthermore, a mobile agent has an ID that is composed of some unique ID of its owner (see Section 2.4.3) and a sequence number (attributed under the control of its owner).

2.4.2 The Agent Platform

The agent platform is the other fundamental part of the infrastructure and represents the static component of a mobile agent system. It belongs to a certain principal (see below), who also owns the data and facilities that are the primary reason why agents migrate to its platform.

We have already briefly discussed the agent platform's general function in the example of Section 2.2.2 as well as its counterpart for mobile code execution in Section 2.3.2. This general function is to serve as a landing pad or docking station for mobile agents, i.e., to offer them an execution environment that supports the language in which the mobile agent was conceived, hides the heterogeneity of the underlying hardware, and provides the necessary resources for the agent's execution. The designers of a particular mobile agent system have to define which of the services described in the following Section 2.5 have to be implemented by the agent platform and how the mobile agents that are part of this mobile agent system can access them. In this sense, the tasks of an agent platform are very similar to those of an operating system, which provides standardized services (such as process execution, resource management, interprocess communication, and protection) and a well-defined means to access these services (e.g., POSIX [IEE96] or Win32API [SGB96]).

In order for mobile agents to be able to migrate to a certain agent platform, the platform has to be stationary and must have a valid network address. This indicates that an implementation of an agent platform has to be built around a stationary server that is permanently connected to the communication infrastructure. We assume that the relevant information will be gathered in a structure that consists of the name of the agent platform's operator, the physical address in the network, and the available resources. A mobile agent can obtain them from a broker and combine them to create an itinerary.

2.4.3 Some Agent related Terminology

Before we discuss the services of a mobile agent system in detail, we want to establish some terms for the roles that principals can take over and relate them to the lifecycle of a mobile agent. These concepts should help to make the following discussion clearer.

Roles in the Mobile Agent Paradigm

A principal can perform several roles. We want to identify the following roles that are relevant for the mobile agent paradigm:

- the agent creator also referred to as programmer,

- the agent owner also referred to as user,
- the agent executor also referred to as platform operator, and
- various third parties, such as name server, broker, or trusted third party (TTP).

The *agent creator* writes the necessary code that has to be executed in order to accomplish a given task (we will refer to this unconfigured form of an agent as *agent program*). It is possible that a user writes his own agents but we consider this to be a rare exception. An agent program can be very specific for a certain task, which will limit the usefulness of the resulting agent, or it can be highly generic so that it can be configured to accomplish several different tasks, which will make the corresponding agent more useful to its users. Consider, for instance, an agent that can only purchase long-distance flights from a single airline as opposed to an agent that can purchase arbitrary transportation tickets from any airline, train, or bus operator. The latter agent will be more useful, since it can organize trips with different means of transportation. Obviously, the possibility of whether an agent can be highly generic and configured to accomplish different tasks depends heavily on the amount of standardization that is in place in a certain application domain. Without such standardization, the agent has to implement many different interaction protocols for the various providers, which make it more error-prone and more difficult to handle due to its increased size.

The *agent owner* creates the agent from the agent program, configures it with the data that is needed to accomplish a certain task, and provides it with the required authority to accomplish this task. (It would be possible to split this role into an *agent configurator* who provides the configuration information and an *agent sponsor* who lends the agent his authority – for most discussions we will limit ourselves to the single agent owner role.)

The *agent executor* provides and manages the agent platform on which agents can be executed. It needs to have some vested interest in executing an agent, such as having a business relation with the agent owner, where it requires payment for the service of executing the agent, or the agent executor uses the mobile agent paradigm only for the purpose of interacting with users, where it recovers payment indirectly. In this case it may provide a completely different service, such as the sale of physical goods or some value added service for which it will charge agent owners.

Finally, the various *third parties* provide facilitation services that are required for the smooth interaction of components and principals. For instance, to allow agents to discover the right agent executors (i.e., those that are capable and willing to execute a particular agent) or to allow agent executors to verify the authority of a particular agent. If the agent interacts with these third parties by migrating to an agent platform that they operate, they become regular agent executors (with the special task of a facilitator). However, we assume that an agent can also interact with these by simple message passing or via RPC interactions (see Section 2.5.5).

This technical, agent centric view of the roles of principals in a mobile agent system has to be matched with the business oriented, application centric view on the same system. In the case of electronic commerce, for instance, we can identify the following roles:

- customer
- service provider
- TTP (e.g., banks, certification authorities (CA))

	agent creator	agent owner	agent executor
design	x		
implementation	x		
distribution	x	x	
instantiation		x	
configuration		x	
migration		(x)	x
operation			x
termination		x	(x)

Table 2.1: The Agent Lifecycle

- non-trusted facilitator (e.g., broker, trader, directory service)

The agent owner can usually be identified with the customer in an e-commerce scenario. However, this is not the case for the agent creator or the agent executor, which can both be associated with the service provider, a non-trusted facilitator, or a TTP. For the ease of presentation, we will often identify the agent executor with the service provider and the agent creator with a non-trusted facilitator, even though this is not always accurate.

Agent Lifecycle

The above discussion of the different business roles also hints at the lifecycle of a mobile agent. This lifecycle is depicted in Table 2.1, where the lines signify the various steps of the agent lifecycle and the columns depict the role of the principal with whom it interacts at a certain step.

The agent is designed and programmed by the agent creator in order to accomplish a particular class of tasks. It is then acquired by the agent owner or customer, who needs an agent for this class of tasks. The business model of how this acquisition takes place can be rather complex: there might be agent programs that are in the public domain and that everyone is free to use (these might be sponsored by the service providers who want to sell their actual value added services), there might be agent programs that a customer can buy and is then free to use at his own discretion, or there might be agent programs that require payment of a certain fee for their use (based on the number of usages, the usage duration, the value of the mediated transaction, or any other appropriate metric). Once an agent program has been acquired by a customer, he will instantiate it and provide it with the necessary information for the task at hand[5]. The agent will interact with the customer in order to obtain all the information that it (or rather its programmer) considers necessary to accomplish the task without further interaction with the customer. Depending on the task, the same agent may request very different types of information from the customer during this interaction. Obviously this requires a very deep understanding of the task by the agent creator, who has to anticipate as many possible outcomes of the agent execution as possible. If the customer has previously interacted with the same agent, the agent may even have stored certain static information (such as the customer's name, address, or payment data) or it may

[5]For the sake of simplicity we will assume that the agent contains the user interface for the interaction with the customer. This assumption might prove rather inefficient in reality, but the user interface could easily be factored out from the mobile agent and become a standalone program on the customer's machine.

read this data from a special preferences file that can be shared among different agents in order to avoid that the customer has to enter the same information several times.

When the agent is satisfied with the information supplied by the customer or if the customer considers the information to be sufficient, he will instruct the agent to migrate to the first platform with which it will interact. It is possible that the agent will never leave this first platform (this is the case for stationary agents, see Section 2.6.2 or for very simple agents that may simply return a result to their owner) but most often the agent will acquire a certain amount of information, (which can be a single address for the next host that it will visit or some complex information that it extracted during the interaction with other agents or from resources available on this platform) and then move on to another platform where it will continue its task.

It is possible that the agent needs some additional information from the customer when some unforeseen or highly unlikely situation occurs and the original information is not sufficient for the task (e.g., in the case of an agent that searches for an airline ticket, no single seat for the given date available). In such a case the agent can return to the customer and request the relaxation of various constraints (e.g., a different date, an additional stopover, another nearby airport as destination/origin, a different class, a larger group of possible carriers). However, this interaction is very similar to a regular migration, with the local resource required by the agent being the agent owner.

Finally, when the agent has accomplished its task it can either send the result in a message to the agent owner or return to the agent owner's platform to deliver it there. After it has delivered this result, the agent terminates.

2.5 The Services of the Mobile Agent Paradigm

A mobile agent system has to provide the necessary services to enable mobile agents to accomplish their tasks. These are rather abstract services that are highly dependent on each other and can be implemented in many different ways. Some of them can clearly be allocated to one of the components of the mobile agent system (e.g., agent execution is a service that must be provided by the agent platform), while others can not as easily be allocated and may be realized in different components or even be partitioned over several components that have to cooperate to provide the service. Even though the interface between a mobile agent and the underlying agent platform has to be well-defined in an actual implementation of the mobile agent paradigm, there is still a large amount of flexibility in where to put the software that implements a particular service, provided that the underlying layer offers sufficiently powerful primitives. For instance, an agent can integrate the software or at least parts of the software for agent mobility as explained in Section 2.5.1. This software does not necessarily have to be programmed by the agent creator, but can be provided by the author of the mobile agent system and it may not even have to be sent with the agent since it will be available on all agent platforms of this mobile agent system in the form of libraries. Nevertheless, since the software is still logically part of the agent, it can, if necessary, be changed by the agent creator.

In the following discussion, we will describe the service in the context of the component that we consider the most appropriate for the service's realization. The goal of this discussion is to present an overview of what exists and to discuss the services on a conceptual level without too much technological detail.

2.5.1 Mobility Support

One of the defining features that distinguishes mobile agent systems from other infrastructures for distributed computing and consequently one of the most important services in a mobile agent system is the service that enables agents to be mobile.

In principle all code is mobile. It may have to be physically transported (e.g., on a floppy disk or CD-ROM) or electronically transferred (e.g., via ftp, http, or smtp) to its new location, then manually installed by a human operator (which may include a compilation into executable code if the original code was transmitted as source), and explicitly started by the operator. On the other hand, a normal program that is installed on a certain host, which may even be connected to a network, can not simply suspend its execution and migrate to another host even though it might be capable to exchange messages with this host. This implies that agents themselves are *not* mobile, but have to rely on the support and the cooperation of the mobile agent system in order to migrate. In the following, we will assume at least some minimal support for agent mobility and want to note that one of the distinguishing features of mobile agent systems is the degree of automation for this mobility support.

The problem of a mobile agent, say A, migrating from the agent platform on which it is currently executing, say P_{origin}, to another agent platform, say $P_{destination}$, can be divided into two rather independent parts that have to be dealt with on the respective agent platforms in a way that allows interoperation.

- On P_{origin}, the agent A has to be prepared for migration, packaged in a suitable transport format, and the marshaled agent $\boxed{A}$ must be sent to $P_{destination}$.
- Conversely on $P_{destination}$, the marshaled agent $\boxed{A}$ has to be received, extracted from its transport format, admitted for execution on $P_{destination}$, instantiated on $P_{destination}$, and launched.

The decision of whether to actually admit a received agent and to launch it on the destination platform is a very difficult problem that depends on various issues. It is mainly a question of access control, which will be discussed in Section 2.5.6 on security.

In the context of support for agent mobility, we now want to address the more technical problem, of how a flow of execution on P_{origin} can be transferred to $P_{destination}$. The most complex problem in this context is due to the necessary separation of an agent's state into its data and its execution state. This distinction is important for the mobility support and is anchored in the way execution environments are implemented. The data of an agent comprises the values of the instance variables of the set of objects that make up the agent. This information can easily be accessed and packaged during an agent migration. New objects that are dynamically created by the agent have to be added to its set of objects. One of the problems here is that references held by an object O_1 to another object O_2 have to remain valid after the migration to a new platform.

The execution state of an agent comprises the values of the registers of the virtual machine on which it executes, the *thread information*, and the execution stacks for each of the threads, which hold the *dynamic variables* of the *active methods* and the *method invocation history.* This information is entirely managed in the execution environment and, thus, not directly accessible to the agent.

In order for an agent to be able to migrate from P_{origin} to $P_{destination}$, the following problems have to be resolved:

- The code of the agent has to be transferred to $P_{destination}$: this is normally quite simple since the code of an agent is usually immutable, so that the original code that arrived with the agent can simply be stored and sent as is, when the agent wants to migrate.
- The data of the agent has to be transferred to $P_{destination}$: this requires a complete list of the objects that belong to the agent as well as access to the instance variables of these objects. It can be realized with the help of a special method of the object that returns a stream of bytes, which encodes the values of its instance variables, similar to the object serialization in Java [Mic97].
- The execution state of the agent has to be transferred to $P_{destination}$. This is the most difficult part of the agent migration and can be handled in several ways, depending on the support provided by the agent platform. In the following we want to discuss two extreme cases for this support.

The literature on mobile agents (e.g., [Ord96, CGPV97] identifies mobile agent systems that provide *strong* support for agent mobility, in which case the agent platform takes care of all the actions required to transfer a mobile agent, and systems that provide *weak* support for agent mobility, in which case most of the required actions are taken over by the mobile agent and the agent platform only provides the support that is absolutely required to enable the agent to accomplish this task. Other approaches to agent mobility, in which the mobile agent and the agent platform partition the required actions in some meaningful way are also possible.

In the case of strong support for mobility, P_{origin} suspends the agent's execution, extracts the current execution state of the agent from its data structures, and marshals it together with the agent's code and data into the transport format. This functionality of the platform is usually coupled with the actual migration of the agent and offered to the agent in some form of "`go (destination)`" command. The migration is accomplished by sending the marshaled agent to the destination platform given in the command issued by the agent. The coupling of the marshaling with the actual migration makes sense since any further actions of the agent on the current agent platform would not be reflected in the marshaled state of the agent. When $P_{destination}$ receives the agent and decides to admit it for execution, $P_{destination}$ instantiates the objects of the agent, integrates the agent's execution state in its data structures, and restarts the agent where it left off.

In the case of weak support for mobility, most of the actions of the agent platform described above have to be taken over by the mobile agent itself. In the following we will assume an absolutely minimal support from the underlying agent platform. This minimal support for agent mobility consists in accepting a message that contains code and data, to install the code with proper access to its data, and to launch its execution on the local agent platform. The mobile agent must be capable to analyze its current execution state (if necessary by using additional internal object variables to encode this state) and it must have access to its own code (again, if this is not supported by the underlying agent platform, the agent needs to carry a copy of its own code in the data stored in its internal object variables). Since the agent is capable to read all of its data, it can use the information stored therein, to create a message that consists of the agent's code and its data, which in turn contains the agent's current execution state (as encoded in the additional object variables).

With the help of the communication service discussed in Section 2.5.5, the agent can then send this message to $P_{destination}$. This agent platform must be capable and willing to instantiate the agent contained in the migration message and the instantiated agent must further be capable to restore itself according to the state encoded in its data, so that it can continue its execution where it left off on the previous agent platform. The original agent, which initiated the migration will wait for an acknowledgement from the migrated instance of the agent or from the destination agent platform, which confirms the successful migration of the agent, and can then terminate itself in order to finalize the migration (or it can continue its execution which effectively results in a cloning of the agent).

All of this rather complex functionality can (and should) extensively be supported by appropriate library routines or even completely automated as in the WASP system described by Fünfrocken et al. in [Fün98]. In this approach, the code is instrumented at compile time with additional instructions, which take care of encoding the state of the agent in its data, so that it can simply be migrated by shipping the code and the data of the agent's objects. This allows the creator of the agent to use the system as if it provided strong mobility and, thus, to concentrate on the actual task of the agent.

Even though it is possible to layer support for transfer of the execution state on top of a system with support for weak mobility, we consider a strong support offered by the agent platform as the most appropriate approach. However, this is not the approach taken by a majority of the experimental systems that offer agent mobility. The reason for this lack of support from the agent platform is that the capture of the execution state of an agent is quite complex and, for instance, not supported by the Java virtual machine, which serves as the execution environment in many of the experimental systems (Aglets, Voyager, Jumping Beans, Odyssey, Mole, etc.). In this situation, it might be simpler and more efficient to create a custom solution, which only encodes the relevant state information (such as what was the best offer and where this offer was found) in the agent's data and otherwise restart the agent in some well-defined initial state on each new agent platform. This is the solution that is proposed for creating agents in the aglets system.

In the following we will usually not explicitly distinguish between code, data, and execution state of a mobile agent. This distinction is not relevant for most other considerations on mobile agent's, where it is sufficient to distinguish between the immutable part of the mobile agent (its code and static data) and the mutable part (its variable data and execution state). For ease of presentation we will mostly use the term code to designate the immutable part of the mobile agent and the term state for the mutable part.

2.5.2 Agent Execution

The basic task of any mobile agent system is the actual execution of mobile agents. This means that the code of a mobile agent has to be executed in the context of the data and facilities available at an agent platform, which were the original reason for the migration of the mobile agent to this agent platform. The execution of an agent's code is accomplished in an *execution environment*, which is thus the incarnation of the agent execution service. The execution environment is an integral part of the agent platform and is also responsible for the provision of a standardized runtime library with a well defined API (application programmers interface). This runtime library implements the interface between the mobile agents and the other services offered by the agent platform. It also allows to reduce the size of mobile agents since they do not need to carry the code which they can be confident to find on the agent platform.

The code of a mobile agent is written in a certain language that must be supported by the execution environment. We can identify three conceptually different levels for this language, which can be:

- machine instructions for a particular physical machine, which reduces the execution environment to a virtual machine monitor (e.g., ActiveX even though there is no execution environment that controls the execution of the code);
- an intermediate bytecode, which are the machine instructions for some virtual machine that is implemented by the execution environment (e.g., Java bytecode).
- a high-level programming language, which has to be interpreted by the execution environment (e.g., Tcl, Perl, or Python);

In all three cases, the execution environment has to execute the code correctly (or in the first case supervise its execution), constrain its access to the allocated resources (this is a very important issue to which we will return to in Section 2.5.3), and provide the access to the runtime library. The three levels are blurred by the fact that the machine instructions of a particular machine can be interpreted as intermediate bytecode and executed by a virtual machine on a completely different physical machine or the high-level language can be compiled into an intermediate bytecode or actual machine instructions before being executed.

It should be clear from the above discussion that the agent execution has to be closely integrated with most of the other services, notably agent mobility (the agent has to be executed in the correct state and should start its execution at a well-defined point), but also resource management, heterogeneity support, security, and fault-tolerance. The agent execution is a very basic service that must be provided by the agent platform, which in turn may rely on lower layers for additional support.

2.5.3 Resource Management

For its execution on an agent platform, a mobile agent needs access to a certain amount of resources on this agent platform, such as CPU cycles, memory, stable storage, or communication bandwidth. Moreover, the mobile agent needs access to the particular data and facilities of this agent platform. The allocation of these resources must be guided by some policy of the agent platform, which identifies what kind of and how many resources should be made available to a mobile agent.

Once certain resources have been allocated to an agent, confining the agent to these resources is a very important issue. Most of the mechanisms required for this purpose are well-known operating system mechanisms. The CPU can be protected from monopolization or excessive use through a clock interrupt, which limits the amount of time an agent can execute. The creation of new threads can be regulated via a limitation on memory usage and on the number of CPU cycles consumed by all the threads that belong to a certain agent. The access to memory can be protected with base and limit registers or paging mechanisms. This issue is very important in the context of security and allows us to protect the agent platform from possibly malicious mobile agents and mobile agents from one another. We assume that any access to resources other than the CPU and memory (which are required for code execution) will be mediated through functions in the runtime library offered by the execution environment (this includes the allocation of new memory). It is therefore the task of the execution environment to implement generic access control mechanisms for other resources and to enforce the access control policy defined for the agent platform. For further information on these mechanisms, please refer to [Pfl89, SG94].

2.5.4 Heterogeneity Support

The notion of heterogeneity has two independent aspects, which are the heterogeneity of the underlying hardware and the heterogeneity of mobile agent systems. Both of these can cause problems for the interoperability of mobile agents. To alleviate the first problem, the agent platforms that belong to a particular mobile agent system should hide the heterogeneity of the underlying hardware from the mobile agents, so that a mobile agent can be executed on any suitable hardware. This service must be provided by the agent platform and is offered by almost all experimental mobile agent systems, which can rely on the wide availability of interpreters for high-level languages, such as Tcl, Perl, or Python, as well as that of Java virtual machines for agents based on Java bytecode.

The second aspect of heterogeneity is, due to the immaturity of current implementations, often neglected. It would also be desirable to hide this heterogeneity, in order to enable agents from different mobile agent systems to interoperate. The issues that have to be resolved in this context are different programming languages for mobile agents (possibly even within a single mobile agent system), different APIs that offer similar but not interoperable services, and the underlying system philosophy, which may have serious effects on the way agents are designed. For instance, the mobility service in the *JumpingBeans* system [Ast98] does not allow mobile agents to move directly from one agent platform to another, but requires them to move to a central host before moving on.

A possible approach to this issue (as indicated in [CGH+95]) could be to implement the execution environments of several mobile agent systems within the same agent platform. The services of these mobile agent systems, which are offered to the mobile agents via the runtime libraries of the execution environments, should be shared as much as possible in order to allow a more efficient interaction of mobile agents. This should be completely transparent to mobile agents, which only see their execution environment and the available services. A mobile agent that wants to interact with another local agent in a different execution environment does not need to be aware of this separation since any interaction has to be mediated via well defined interfaces.

2.5.5 Communication

Communication is another indispensable service for mobile agents, which can be used as the foundation for many other services (mobility, fault-tolerance, directory assistance, or agent management). The communication services of a mobile agent system should allow mobile agents on an agent platform to communicate with agents on the same platform, but also with mobile agents on other agent platforms as well as with local and remote entities that are not part of the mobile agent system. For this purpose, the agent platform can offer various communication protocols (e.g., TCP/IP, IIOP, SMTP, or HTTP), that can be used to implement different models of interaction, such as message based, event based, RPC, or remote method invocation. Another possible option is the realization of multiparty communication, such as message multicast, group based communication, or publish/subscribe based approaches. In the following we will, without loss of generality, assume that mobile agents interact via (remote) method invocation on their public interface with both remote and local entities. Depending on the actual implementation, this may have to be explicitly supported by the mobile agent system with the dynamic creation of local proxies.

An obvious difficulty in the communication with mobile agents is to invoke a method on a receiver that does not have a fixed network address. There are several solutions to this, such as a registration server, which is updated whenever the agent moves to a new location and forwards invocations on the agent to its current location, or a proxy, which remains on the agent platform when the mobile agent moves on and forwards invocations to the new location (in order for this to be viable, the new location of the mobile agent has to be propagated backwards to all the proxies). Many optimizations for this problem can be conceived, such as the ones discussed in [IDM91]. We assume that this forwarding of invocations on moving agents is not a major problem in mobile agent systems, since agents will only rarely receive invocations that they did not request previously and if an agent engages in a more active interaction, it can remain stationary for the duration of this interaction.

Similar to the agent mobility service, it would be possible to implement a considerable part of the communication protocols in the mobile agent. This requires only a minimal support from the underlying agent platform, such as the possibility to open a TCP/IP based connection. However, since many of the communication services are already provided by the underlying operating system, the agent platform can simply make these services accessible to mobile agents – under the constraints imposed by resource management.

2.5.6 Security

The security problems in mobile agent systems are manifold and have been identified by many authors as one of the crucial issues for the acceptance of these systems [CGH+95, VST97, KLO97]. These problems are due to the fact that the various principals in a mobile agent system who are the operators or owners of the different components and who lend their authority to these components (see Section 2.4.3), can not be assumed to trust each other.

Security in a mobile agent system is concerned with the protection of the two primary components that we have identified: the mobile agent and the agent platform. The mobile agent has to be protected from tampering and from disclosure while it is in transit, but it should also be protected when it is executing on an agent platform. The agent platform in turn must be protected from malicious or simply buggy mobile agents.

Protecting the mobile agent while traveling over an untrusted network, when it is marshaled into its transport format, amounts to the well-known problems of confidentiality and integrity protection of messages, as discussed in, e.g., [For94]. A simple method to achieve this is to use the Secure Socket Layer (SSL) [FKK96]. Due to the special structure of mobile agents we can also differentiate between the mutable and the immutable parts of an agent, which can be independently protected. Provided that the mechanisms used for integrity protection support origin authentication, for instance by using appropriate public-key signatures (see [MvOV97]), the immutable part can be signed by the agent owner to assert that it has not been tampered with while in transit. Also, the immutable parts of a publicly available agent may not have to be confidentiality protected, unless the user wants to hide the fact that he is using this mobile agent. The mutable part of a mobile agent can be integrity and confidentiality protected with the help of well-known cryptographic mechanisms [MvOV97]. If additional accountability is required, the intermediate states of the mobile agent can be signed by the agent executors and gathered in the mobile agent (see Section 3.2.2). The two parts should be tied together with appropriate cryptographic mechanisms to prevent an attacker from arbitrarily composing these parts.

The problem of protecting a mobile agent when it is executing on an agent platform consists of protecting it from other agents on the same agent platform and from the agent platform itself, which may be operated by a malicious agent executor. The former problem is a well-known problem from operating systems that has to be addressed in the resource management (see Section 2.5.3). As we have already discussed in the corresponding section, the agent execution service can rely on memory protection and access control mechanisms to prevent local agents or other entities from accessing or tampering with the code or state of a mobile agent. The latter problem, which is concerned with preventing undesired access to or manipulation of the mobile agent's state or code by the agent platform, is widely accepted as an important and difficult problem (see Section 3.1). Currently most mobile agent systems do not explicitly address this problem, but make the assumption that the agent executor is a trusted principal that behaves correctly [Ord96, Vig98a]. Therefore, the mobile agent does not have to be protected from the agent platform. As we have noted in the beginning of this Section, this assumption is not always justified. In the following Chapter 3 we will discuss various approaches from the literature and introduce our approach, which is based on trusted and tamper-resistant hardware.

The remainder of this section is concerned with the problem of protecting the agent platform from malicious mobile agents that try to circumvent the protection mechanisms established by the agent platform. These mechanisms are designed to prevent that a mobile agent can misuse the data (e.g., by destroying, manipulating, or disclosing it) or the facilities (e.g., excessive and uncontrolled use of CPU, memory, or communication bandwidth) of an agent platform, which are the primary assets of the agent executor. In addition, they are also responsible to protect other mobile agents executing on the same agent platform from malicious agents.

This problem is very similar to the problem of protecting a host from downloadable mobile code. Both these problems can essentially be addressed with very similar approaches. The main difference is that since a mobile agent may visit the agent platforms of several agent executors it becomes more difficult to identify who is responsible for the behaviour of such an itinerant mobile agent. Therefore, even though a mobile agent originates from a trusted principal and its immutable part has not been tampered with, the agent executor can not be sure that the agent has not become malicious [FGS96a]

Due to the tremendous success of Java applets, which can be executed in a Web browser, this topic has lately received a lot of attention. A recent survey article by Rubin et al. [RDEG98] has identified the following four different approaches to address this problem (a fifth approach results from a combination of the first two):

- code signing,
- sandboxing,
- firewalling, and
- proof carrying code.

Code signing is the most classical of these approaches. Its goal is primarily to establish accountability by authenticating the principal that lends a mobile agent its authority and that vouches for its correctness. This can be the agent owner, the agent creator, or even both. If the agent executor trusts the signer of the mobile agent, it will execute it with full access to all the resources available in the execution environment. As we have noted above, it is not always easy to establish who is responsible for the behaviour of a mobile agent that has visited many different agent platforms. Moreover, it is possible that a malicious agent does not act immediately, but simply installs a Trojan Horse that can be activated later. Tracing this installation of a Trojan Horse back to a particular mobile agent can be extremely difficult.

Sandboxing has recently been popularized by Java, where it is implemented within a Web browser to limit an applet's access permissions to a well-defined set of resources defined in a security policy (that is the same for all remote applets[6]). This is achieved by a complex interaction of various components that conduct static checks on the code before it is executed (bytecode verifier), create separate name spaces to isolate trusted code from untrusted code (class loader), and supervise the actual access to resources during execution of the code (security manager) [FM96]. Similar goals have also been pursued by Ousterhout et al. [OLW97] in the design of Safe-Tcl, by Wahbe et al. [WLAG93] in the context of software-based fault isolation, and by Schneider [Sch98] who describes security automata for policy enforcement. All of these mechanisms can easily be adapted to work for mobile agents.

Both of these approaches suffer from an all-or-nothing problem that either allows complete access if the signer of the mobile agent is trusted or the same very limited access for all mobile agents. This can be alleviated by combining them into a hybrid approach. The idea is to use code signing to identify partially trusted mobile agents that can be executed under a less restrictive security policy. For instance, an authenticated mobile agent from a registered user may have access to the entire database of the agent executor, while an unknown mobile agent can only access a small publicly available part of the database. (This approach has actually been realized by certain Web browsers in the context of signed Java applets.)

Firewalling, which was independently developed by Malkhi et al. in [MRR98] and by Digitivity Inc. [Cit], tries to achieve similar results as sandboxing by using a simpler approach that is less error-prone. It consists of defining an additional protection domain for the host of the agent platform, which can be located in the perimeter network between an internal and an external firewall [CZ95]. The host of the agent platform is considered a sacrificial machine that does not have immediate access to valuable resources. This requires the use of a trusted server on a protected machine that implements a security policy. This server has access to the local resources that should be made available to the mobile agent. The approach is very much geared towards Java applets and their primary task of providing graphical user interfaces for a user. It is not clear if it can provide additional security in the context of mobile agents, which usually have a more complex interaction with the agent executor's environment and may in any case be executed on a dedicated machine.

Finally, a potentially very powerful approach is proof carrying code, which was introduced by Necula et al. in [NL96]. It builds on the formal verification of software, which consists of constructing a proof that the agent's code respects a given security policy. Since such a proof is difficult and expensive to create, it is created once by the creator of the mobile agent, encoded in a safety certificate, and appended to the mobile agent. This proof can then easily be verified by the agent executor. If it is valid, then the agent executor is ensured that the mobile agent will not violate the security policy. The approach is problematic since it requires advance knowledge of the security policy under which the code will be executed and it is currently not possible to produce proofs for arbitrary code automatically [NL96].

[6]Newer versions of the Java security architecture provide more flexible solutions to this problem.

Due to the various possibilities to implement security for the agent platform, it is not clear where this service should be realized. In the case of sandboxing or firewalling, the agent platform may rely on many services of the underlying infrastructure for its realization, while the other approaches have to be addressed in the agent platform. The protection of the mobile agent in transit can easily be achieved using cryptographic protocols that are readily available in the form of software libraries from the underlying infrastructure. The protection of the mobile agent from other local agents must be realized by the agent platform with mechanisms from operating systems, which can, again, be supported by the underlying infrastructure. The remaining problem of protecting the mobile agent from the agent platform will be addressed in the following Chapter 3.

2.5.7 Fault-tolerance Support

The support for fault-tolerance in a mobile agent system is interesting, since a mobile agent can be executing for an extensive amount of time and thus dispose of rather valuable information. This information could be completely lost if the mobile agent cannot survive a simple crash of the host it is currently executing on. Most existing agent platforms do not deal with this issue, which is an important problem for an actually deployed system. If users have to pay for their agent's execution on a particular platform, they do not want the results of such a computation to be lost by the crash of another platform that is visited by the agent at a later point in time. Furthermore, the detection that an agent has been lost due to the crash of a platform can take a considerable amount of time, which can be longer than a successful execution of the agent. Users may not be willing to wait for such long periods for the result of a mobile agent.

The following discussion of support for fault-tolerance is obviously far from being exhaustive, but an appropriate discussion would be beyond the scope of this thesis. For more advanced discussions of this topic, please refer to [JMS$^+$98, MvRS96, Sch97a, RS98].

The minimal goal for fault-tolerance in the context of the mobile agent paradigm would be to guarantee the survival of a mobile agent despite the possibility of hosts and the corresponding agent platforms crashing. The realizability of this goal depends on the assumptions that we make about the underlying hardware. We assume that a host that operates an agent platform has an unlimited amount of stable storage available (modern hard disks make this an almost realistic assumption), can suffer from crash failures only, and recovers within a bounded amount of time.

This allows us to model the migration of a mobile agent as a transaction between the sending and the receiving host [RS98], which will eventually be successful provided that the uptime of both hosts is sufficiently long to perform the migration transaction. After the migration transaction has succeeded, the marshaled agent is stored on stable storage and ready for execution. Since the host may also crash during the execution of the mobile agent, this again requires a mechanism to ensure that the mobile agent will eventually be executed on this host. This can also be modelled as a transaction that is successfully completed when the mobile agent terminates or a subsequent migration transaction with the next host on the mobile agent's itinerary succeeds. The described procedure for fault-tolerant migration is depicted in Figure 2.3.

In the above discussion, most of the necessary actions to ensure fault-tolerance have been attributed to the agent platform, which we consider to be the most appropriate component. Similar to the agent mobility service, it might be possible to allocate some functionality to

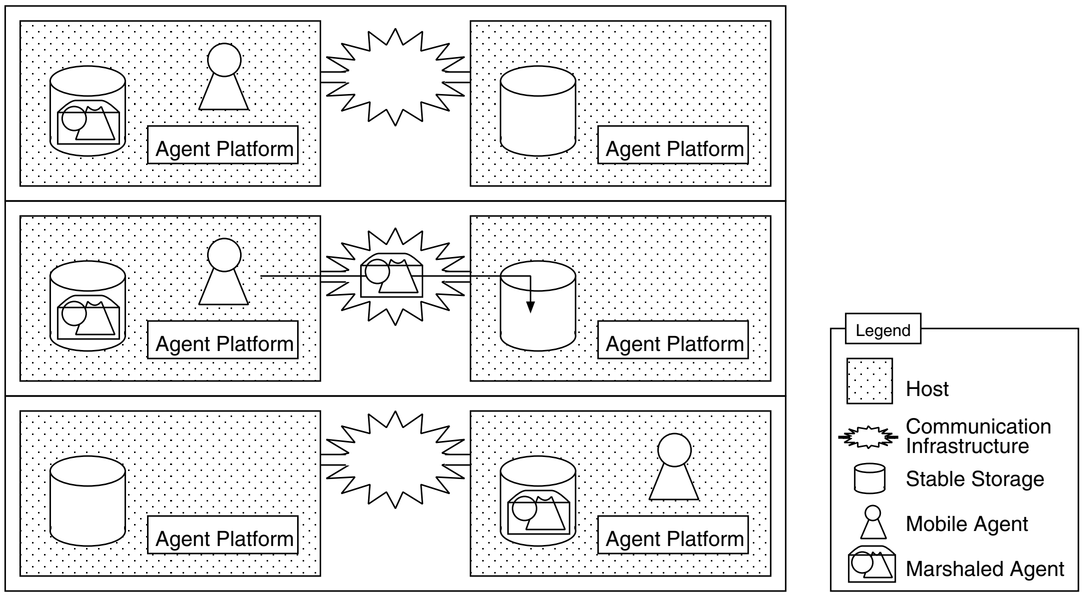

Figure 2.3: A strategy to guarantee agent survival

the mobile agent, which would require some not yet specified basic support from the agent platform.

2.5.8 Directory Assistance

Mobile agents should be capable to dynamically discover required services and resources before actually migrating to a new agent platform. The problem of efficient service discovery is a general problem of distributed systems, such as CORBA or DCOM, where it is addressed with specialized servers (e.g., broker or trader) [OMG98a, Chapter 16].

The major problem that needs to be addressed in the mobile agent paradigm is the highly dynamic behaviour of the system. It is possible to use a directory service on a particular agent platform as a registry for mobile agents, which allows locally executing mobile agents to discover the presence of other agents on the same agent platform. However, due to the latency of a remote communication it is difficult to provide this same service to remote agents.

A mobile agent system does not necessarily have to implement this service as part of the system, but it may simply integrate existing solutions that are made accessible via the communication service.

2.5.9 Agent Management

The agent management service is concerned with the control of active mobile agents. An agent owner may want to know about the progress of a mobile agent's execution or simply its current location. The latter information can then be used to send a message to the mobile agent, or even to issue a command that instructs the mobile agent, for instance, to change its itinerary, to return to its owner, or to terminate. If an agent owner launches several collaborating mobile agents, then it may be necessary to address these as a group and to be able to multicast messages to all of them. This does obviously have to be coordinated with the security service, to prevent an attacker from issuing a command to the mobile agent of another principal.

This service does not cause any principal problems, since a mobile agent can simply report all of its actions back to the agent owner, in order to keep him completely informed (which is the approach taken by most currently existing mobile agent systems [Obj, LC96]). However, such an intensive communication goes against some of the reasons to use mobile agents in the first place and is not scalable to a widely deployed mobile agent system with a large number of mobile agents.

2.6 The Use of Mobile Agents

The mobile agent paradigm is a means to structure distributed applications and has often been identified as an extension or generalization to the well-established *client-server* paradigm [CGH^{+}95, CPV97]. The "new concept" that is introduced by the mobile agent paradigm is *location awareness*. While the location of a particular computation has always played a major role in distributed computing, this was mostly conceived as an artifact of the underlying system that causes additional problems (which must be overcome).

The usual solution consists in hiding the location and making the computation independent from it. This provides a simple abstraction that makes the design of distributed applications more tractable and is well suited for tightly coupled distributed applications (e.g., on a LAN). On the other hand, it can introduce new problems (e.g., partial failures) and result in considerable inefficiencies for loosely coupled distributed applications that have to operate on a WAN, such as the Internet.

By making a computation explicitly location aware and movable, mobile agent systems provide a new degree of freedom. The location does not need to be considered during the design of a distributed application, but can be adapted dynamically during its execution, to take account of the quality (and cost) of interaction between components.

We will first examine the perceived advantages that the mobile agent paradigm has over the classical client-server paradigm and then illustrate briefly what particular applications can exploit them.

2.6.1 Advantages of the Mobile Agent Paradigm

The advantages of the mobile agent paradigm as compared to the classical client-server paradigm can broadly be organized into two categories:

(1) reduction of network traffic, which provides support for

- high bandwidth interactions and
- server customization.

(2) decoupling of the interacting principals, which provides support for

- mobile users,
- robust handling of partial failures,
- asynchronous computations, and
- real-time interactions.

The support for high bandwidth interactions is the most often cited reason to promote mobile agents. It consists of moving the computation (assumed to be small) to the source of data (assumed to be large). The same approach can be found in systems for remote database access that allow a remote host to send SQL queries. In a mobile agent system this becomes more flexible since a mobile agent is programmed in a computationally complete programming language and can more naturally deal with unstructured data. The goal of using this approach can be to overcome a low bandwidth connection or to use a high-cost network connection more economically.

The mobile agent paradigm enables a service provider to offer a fine granular API, for the access to its services. This allows a mobile agent to customize these services to his particular needs or even to create new services that previously had to be installed by the service provider. The problem with this approach is that it usually necessitates the composition of several calls to the server's API for the provision of a complex service. This does not constitute a problem for a mobile agent, but is impractical for client-server interactions. Again, SQL queries provide an example for this argument.

Mobile users who access the network via small, resource limited computing devices, may not be able or willing to remain connected for the duration of a particular service provision. Since a mobile agent can act autonomously without requiring a permanent interaction with a user, it can be configured while being disconnected and then injected into the network to accomplish a well-defined task on behalf of the user. Once the user reconnects to the network (possibly upon receiving an alarm from its mobile agent), he can collect the result of the agent or deal with the problems his agent encountered. A particular support for mobile users can be provided by resident agents, which only perform a single migration from their owner to the agent platform where they take residence. These can be configured to handle routine actions or simple management tasks for their owner (see below).

A well-known problem in distributed client-server applications is that of a partial failure. Since it is in general undecidable whether an unresponsive communication partner has crashed, is partitioned, or is simply slow, it is difficult to determine where and how to react in order to recover a distributed application in a consistent state. This problem may be simplified in the mobile agent paradigm, by gathering the interacting components on a single host, which can then use checkpointing algorithms to ensure the correct and complete execution of a mobile agent. If the agent platform of the service provider is well managed and can be assumed to never crash forever, it can guarantee that a mobile agent will eventually be executed exactly once (this requires that a mobile agent has a unique ID and that the service provider stores the result of a mobile agent until it has received a positive acknowledgement from the agent owner for the reception of the result).

The support for asynchronous computations means that the machine of a user is not loaded with the task of the mobile agent and is free to work on other tasks for its owner. This is particularly interesting in the case of mobile clients (PDA or mobile phone), which are rather resource limited computing devices.

A mobile agent can interact with remote components directly, without having to suffer from a communication delay due to an intermediate network. This may occasionally be important when a user wants to use services, that depend on a (soft) real-time interaction, via a high-latency network connection. An extreme example of this is the control software for space probes [HCK97] (e.g., the Mars Pathfinder Sojourner robot).

None of the discussed advantages mandate the use of mobile agent technology as the only possible solution to resolve a particular problem (with the possible exception of real-time interactions) and can rather be seen as engineering issues. Nevertheless, these can be decisive for the actual deployment of a particular application.

2.6.2 Applications of the Mobile Agent Paradigm

Mobile agents are often proposed for electronic commerce applications, similar to the example in Section 2.2.2. In this environment they perform various tasks, such as searching for offers, negotiating terms of an agreement, or even purchasing goods or services. Electronic commerce applications utilize many of the advantages of the mobile agent paradigm. For instance, in the context of users who conduct routine purchases while they are on the move or who configure complex requests that require the mobile agent to visit many different service providers and to scan large amounts of data (e.g., organizing a business trip). Likewise, the negotiation of terms often requires a fine granular interaction, which results in the exchange of a large number of messages.

The use of mobile agents for searching information (which is an application that has been deployed on the WWW although with non-mobile agents [Bot]) can be seen as a special case of electronic commerce, where the good that is sought for, is information.

A slightly different approach to use mobile agents is, paradoxically, one where the agent takes residence at a service provider and remains primarily stationary (after the initial hop). This can be useful for monitoring applications, where the mobile agent is supposed to observe a particular event and consequently moves to the location where the event is expected to occur or to be observable (e.g., scanning for interesting news items or stock ticker values). This event can be rather complex, such as the sequence of a certain number of primitive events, or the succession of two primitive events within a certain time frame. The user might regularly update the agent and define new events in which he is interested.

Another interesting use for these stationary agents is in communication contexts, where the user needs a fixed address in the network in order to allow others to get in contact with him. Instead of being directly connected to this fixed address, the user can place a stationary agent at this fixed address, which will negotiate with callers on how to proceed. The stationary agent holds the current network address where the user can be reached (the user has sent this to the agent[7]), but it may, for instance during the night, decide to deviate the caller to some asynchronous communication device (e.g. voice mail or mail box). On the other hand, if the caller authenticates himself to the agent as a close friend and signals a sufficient urgency, the agent will route the call to the registered network address of the user.

Other often cited applications for mobile agent systems include workflow management systems, groupware applications, and network management [FM99, BGP97]. These applications are mostly realized as tightly coupled and closed systems, where all components are under the administration of a single principal. In such a setting the security problems, and in particular the protection of the mobile agent from the agent platform (which is our primary interest), are less relevant.

∞∞∞

[7]This is similar to the task of the HLR in GSM, which manages the current location of the user [MP92]. The location information together with the code that manages it, could be interpreted as a stationary agent that might also be configured by the user to protect his interests (see Section 5.2).

Summary

In this chapter we introduce the system model for mobile agent systems. This provides the background for the following chapters, where we study the immediate protection of mobile agents and the less obvious use of mobile agents for the protection of privacy. We present the environment in which we assume mobile agents to evolve and give an example for their expected use. The history of mobile agent systems is discussed in the context of the underlying concept of mobile code, which has given rise to the simpler paradigms of remote evaluation and code on demand. Then we discuss the mobile agent paradigm first with respect to the components that make up the infrastructure of a mobile agent system and second with respect to the services that have to be offered by a mobile agent system. We conclude the chapter with an illustration of the mobile agent paradigm's advantages over the classical client-server paradigm and give a brief overview of particular applications that are well suited to exploit them.

Chapter 3

Protecting a Mobile Agent with Tamper-resistant Hardware

Yesterday's magic is today's science and will be tomorow's technology.

(anonymous)

3.1 Introduction

In the previous chapter, we have introduced the mobile agent paradigm and identified the problem of protecting the mobile agent from the execution environment (or rather from the operator of the execution environment). We now want to present our approach to address this problem, which is based on a trusted and tamper-resistant hardware device. The problem of protecting a mobile agent from its execution environment has been identified by many authors as one of the crucial problems for the deployment of mobile agent technology [CGH$^+$95, Vig98a, VST97, PK98]. This has resulted in a considerable body of work that we will explore in the following section.

In a conventional mobile agent system, it is easy to protect the confidentiality and the integrity of the entire agent from external attackers (see Section 2.5.6). Similarly, it is also easy to protect data gathered by the mobile agent, by encrypting it with the public key of the agent owner, provided that this data is not needed before the agent returns to its owner. However, once the mobile agent arrives at its destination where it is supposed to be instantiated on some local agent platform, the code and the state of the agent have to be accessible to the agent platform (and thus to the agent executor) in order to execute it. Due to the problem of information disclosure, the agent owner loses all control over the code and state of the agent. The agent executor can:

- store the agent for an arbitrary period of time,
- execute the agent arbitrarily often,
- analyze and reverse engineer the agent's code and state,

- arbitrarily change the agent's code and state[1], or
- disclose the entire agent to another principal.

The agent owner has no way to control or even know about the actions of the agent executor, who can thus obtain confidential information from the mobile agent, execute it incorrectly, or execute it only partially[2]. This has led to the conjecture that the problem of protecting a mobile agent from the execution environment is impossible to solve and that the problem of protecting a mobile agent from the agent executor can only be solved with tamper-resistant hardware [CGH+95, FGS96b, Vig98a]. This conjecture has been challenged by the work on *computing with encrypted functions* of Sander and Tschudin (see Section 3.2.7), but it is still an open question whether their approach can provide a general solution to the problem. Currently most mobile agent systems make the assumption that the agent executor is a *trusted* principal that behaves correctly [Ord96, Vig98a] and provide no mechanisms to protect the contents or the execution of a mobile agent.

A mobile agent can be charged with a multitude of tasks that may have important security and privacy implications. One of the most frequently cited examples where a mobile agent requires protection is in the context of electronic commerce, in which a mobile agent searches for a particular service and subsequently pays for using the service or for some goods it purchased [FGS96a, Yee99]. Such an agent often visits several principals that do not trust each other and may often be competitors. Possible attacks could be to simply destroy the mobile agent, to alter or delete information the mobile agent has collected previously at competitors, to alter data in the mobile agent so that it will not perform correctly (or maybe even maliciously) on the sites of competitors, or to change the mobile agent's itinerary such that it will not visit a competitor. The goal of such an attack is to make it more likely that the mobile agent will select the malicious attacker as the provider of the service.

These direct attacks on the primary functionality of a mobile agent are, however, not the only problems. Even if a mobile agent accomplishes its task without problems or interference, problems related to the data contained in the agent may still persist .

In the example of Section 2.2.2, the mobile agent may contain the payment information (such as electronic cash, credit card, or even payment information for several methods), the user's schedule, or her personal preferences. All of this data, although necessary to process the request, is confidential information that should only be used for the purpose for which it was provided. For instance, a service provider should only obtain a single payment information in case it is selected to provide the service and the user's schedule should only be consulted to find out if a given departure time is suitable, but it should not be accessed in its entirety.

Other data contained in a mobile agent should only be accessible to the agent executor once the mobile agent has decided that an offer is acceptable. For instance, in the case of a mobile agent that mediates translation of foreign language texts, the text should only be accessible to the translation office that actually performs the translation[3]. For other applications it

[1]It is possible to protect the integrity of the immutable parts of an agent (code and static data) with simple cryptographic techniques. However, this makes it only *possible* for an agent executor to detect (at least some) tampered agents and to avoid executing them. It does not prevent the agent executor from changing the immutable parts and from experimenting with the agent in the context of a local execution.

[2]If the mobile agent is never executed to its completion (or not at all), this is often called denial-of-service. The common understanding is that this can not be prevented. Support for fault-tolerance can at most guarantee the survival of the mobile agent, which then has to find another source for the service.

[3]Such an interaction could even require the establishment of some form of contract that legally binds the translation office to keep the text confidential.

may be desirable that the mobile agent is capable to digitally sign a message or to verify the signature on a message it receives[4]. Finally, even if a mobile agent simply searches for some specific data, the query already conveys the interest in the information, which can in itself be highly sensitive. An example for this is a mobile agent that performs a patent search or the mobile agent of a major investor that requests financial information on a company.

The probability that an agent platform (or rather its operator) acts maliciously depends on many different factors. Two obvious factors, which can directly be translated into approaches to protect mobile agents are a possible (financial) gain and a lack of risk. An agent executor is more likely to attack a mobile agent if the gain (which must not necessarily be monetary) is high compared to the cost of mounting the attack, if the risks of being caught are low, and if the consequences of being caught are moderate. Approaches to protect mobile agents can therefore consist of making the cost of an attack prohibitively high (this implies that a single successful attack should not compromise the protection of the entire system), raising an attack's probability of discovery, and making an attack provable to an impartial third party (a court).

3.2 Related Work

The problem of protecting code and data of software modules from the untrusted execution environment, in which they have to be operated, is not new. A prominent example are highly sensitive authentication keys in security modules that are used by banks for the verification of customer's personal identification numbers (PINs). These have to be protected from unauthorized access by the operators of the security module [AK97]. Another, less known example is a proposal by Herzberg and Pinter [HP85] for the protection of software against unauthorized distribution (piracy), which relies on the presence of a secret key within the CPU that executes this software. In a recent announcement, Intel proposed to put cryptographic keys on a CPU, which would allow them to realize this protection. Both of these approaches rely on trusted and tamper-resistant hardware to achieve the desired goal.

The proliferation of mobile agent systems and the more recent idea to use mobile agents in the context of mutually distrusting domains (e.g., Safe-Tcl [OLW97] or Telescript [Whi94]) has relaunched the interest in this problem. This has given rise to a considerable body of work, in which several different approaches to address the problem are identified. These approaches can roughly be divided into two categories:

- detection mechanisms, which attempt to detect unauthorized modification of code, state, or execution flow of a mobile agent. These detection mechanisms can further be subdivided depending on whether they:

  D1 detect manipulations automatically or require a suspicion,
  (AUTOMATIC/SUSPICION)

  D2 detect manipulations during the execution of the mobile agent or after it has terminated, or
  (DURING/AFTER)

[4]If the corresponding keys of the mobile agent could be protected from disclosure or tampering, an attacker could still manipulate the code such that the cryptographic protection is impaired.

D3 detect all possible manipulations of the mobile agent or only some of them. (ALL/SOME)

These properties allow us to distinguish between eight different mechanisms, not all of which are useful. In the following we will discuss three mechanisms that have been identified in the literature, which exhibit the properties depicted in Table 3.1.

	D1	D2	D3
cryptographic traces	suspicion	after	all
forward integrity	automatic	after	some
state appraisal	automatic	during	some

Table 3.1: The Detection Mechanisms

- prevention mechanisms, which try to make it impossible (or at least very difficult) to access or modify code, state, or execution flow of a mobile agent in a meaningful way. These can, in turn, be subdivided depending on whether they:

  P1 prevent attacks on the entire agent or only on parts of it, (ENTIRE/PARTS)

  P2 rely on some trusted functionality, or (TRUSTED/NO TRUSTED)

  P3 prevent attacks permanently or only temporarily. (PERMANENT/TEMPORARY)

  Again, these allow us to distinguish between eight different mechanisms.

  In the following we will discuss four mechanisms that have been identified in the literature, which exhibit the properties depicted in Table 3.2.

	P1	P2	P3
co-operating agents	parts	trusted	permanent
obfuscation	entire	no trusted	temporary
tamper-resistant hardware	entire	trusted	permanent
computing with encrypted functions	entire	no trusted	permanent

Table 3.2: The Prevention Mechanisms

One of the most important differences of these two categories is that the detection mechanisms, due to the way they are constructed, can not cope with the problem of unauthorized *access* to code or state of the mobile agent. This limits these approaches to applications where the data of an agent does not have to be kept secret. The prevention mechanisms, on the other hand, rely on this protection from access to achieve the protection from modification. This is based on the assumption that if an attacker can not access the code or state, he can not change the behaviour of the agent in a meaningful way – he can at most apply arbitrary changes, which cause the mobile agent to not execute at all (this constitutes a denial-of-service).

We now want to discuss some approaches from the literature, which realize the detection or prevention with various means. We would like to remind once more that the following discussion is based on three important assumptions: the confidentiality and integrity of a mobile agent is protected from external attackers, the immutable parts of a mobile agent are integrity protected (this allows an honest agent executor to detect tampered agents), and denial-of-service can not be prevented. Furthermore, the detection mechanisms are ineffective if there is no way to sanction misbehaving agent executors by making them liable for their actions or through some form of social control (see Section 1.6.1).

3.2.1 Cryptographic Traces

In this approach, which is described by Vigna in [Vig98a], an agent executor creates a trace of an agent's execution that contains precisely the lines of code that were executed by the mobile agent as well as all the external values that were read by the mobile agent. The complete trace of the agent's execution is stored for a limited amount of time by the agent executor. When the mobile agent requests to move, a hash of this trace and of the agent's intermediate state are signed by the agent executor (this is a *commitment* that implies that the agent executor has reached this state by a proper execution of the mobile agent according to the indicated trace) and forwarded together with the actual mobile agent to the next agent executor on the mobile agent's itinerary. The following agent executor will store the commitment together with the trace it creates when executing the mobile agent.

The protocol requires several additional messages that have to be stored by the interacting parties. These establish the non-repudiation requirements that are necessary to make any misbehaving party liable for its actions (e.g., an agent executor has to commit itself to actually executing a mobile agent).

When the mobile agent returns, the agent owner receives a hash of the last trace and the final state of the mobile agent, both signed by the last agent executor on the mobile agent's itinerary. Additionally, the agent owner knows the first agent executor to which he originally sent the mobile agent. If, for some reason, the agent owner suspects that the mobile agent was not correctly executed, it will request the complete trace of the agent's execution from the first agent executor, which can not falsely deny having executed the mobile agent. The agent owner will then simulate the execution of the mobile agent based on the information contained in the trace. This simulation will result in an intermediate state and identify the next agent executor, to which the mobile agent was forwarded. The agent owner requests from this principal the commitment of the first agent executor, which contains a hash of the trace and of the agent's intermediate result, both signed by the first agent executor. If these hashes correspond with the trace received from the first agent executor and the intermediate state of the simulated execution, the agent owner knows that the first agent executor did execute the agent correctly.

This process continues until the last agent executor, from which the agent owner has received the commitment directly. If at some point a discrepancy is found during the verification of the trace provided by some agent executor, then this agent executor cheated.

This approach allows to detect any manipulation of the agent's code, state, or execution flow, without having to anticipate where such manipulations can occur or what goals they could pursue. It can only detect a manipulation after the agent has terminated its execution and returned to its owner. Then the agent owner must have a suspicion (based on the final result of the mobile agent or the charges for the resources used by the agent during execution) to launch the verification mechanism, which is too expensive to use it systematically. Furthermore, it places a heavy burden on the agent executors who have to dedicate large amounts of resources to the storage of enforcement information. Finally, the approach is only feasible for single-threaded mobile agents – otherwise it becomes too complex.

3.2.2 Forward Integrity

The basic idea for this approach has been presented by Yee in [Yee99] and was extended by Karjoth et al. in [KAG97]. The approach has been conceived in the context of best price shopping, which is concerned with a mobile agent that roams the network in search of the lowest price for some particular good. The information that is found at a particular service provider can easily be encoded in an offer (that should contain the identity of the mobile agent, the name of the service provider, the offered good, the price, and a timestamp) that is signed by the service provider to make it non-repudiable and to prevent later changes to the offer[5]. The remaining problem is to avoid that a later service provider, say S, removes an offer of some other service provider that provides a better deal, in order to trick the mobile agent into using S's services.

This problem is easy to solve if the itinerary of the agent is known to the agent owner, who can then simply verify that a signed offer from each service provider is contained in the mobile agent. The protocol described by Karjoth et al. guarantees strong forward integrity, which states that if a mobile agent visits a sequence of servers $S_1, S_2, ..., S_n$, and the first malicious server is S_k, then none of the partial results generated at servers S_i, where $i < k$, can be forged. This allows the agent owner to detect the modification or removal of an offer even if the itinerary is not known in advance.

This is achieved via hash chains, which consist of elements that contain an offer from a service provider and are linked to the preceding and to the next element by including a hash of the previous offer and the identity of the next service provider. Such an element is signed by the service provider who is responsible for the element. When the mobile agent returns, the agent owner can verify the entire chain to detect if it has been tampered with during the agent's execution.

In [Yee99], Yee described a simpler protocol that provides only weak forward integrity but does this without requiring digital signatures (this makes the offers repudiable by the service provider). An offer by a service provider is protected with a *partial result authentication code* (PRAC), which consist of computing a message authentication code (MAC) [MvOV97] on the offer based on a key k_i that is contained in the mobile agent. Once the PRAC is created, k_i is replaced with $k_{i+1} = f(k_i)$, where f is a one-way function. This new k_{i+1} is then used by the next service provider to calculate the PRAC on its offer. The agent owner, who knows the initial key k_1, can compute the complete sequence $k_2 = f(k_1)$, $k_3 = f(k_2)$,... and verify the authenticity of all offers as well as the completeness of the sequence (all the offers have to be signed with consecutive values of f).

A problem with this is that a server S_i that has received k_i, can calculate all the keys k_j, where $j > i$. If such a service provider manages to obtain the mobile agent at some later time (when it has visited several other service providers), it can change the offers of all the service providers the mobile agent visited after it has visited S_i. This problem can be overcome with a small modification described in [KAG97].

[5]If confidentiality is required, this offer can be encrypted with the public key of the agent owner.

This approach allows to automatically[6] detect manipulations of those parts of the mobile agent that are explicitly protected. This protection can be adapted to the level of protection that is required. Similar to cryptographic traces, this approach can only detect a manipulation after the agent has terminated its execution and returned to its owner.

3.2.3 State Appraisal

This approach, which is defined by Farmer et al. in [FGS96a], identifies the notion of state appraisal, which is a mechanism that allows a mobile agent to decide what privileges it will need at a particular agent platform. This allows the agent owner to restrict the authority of his agents in a flexible manner. The approach relies on the fact that the state appraisal function belongs to the immutable part of the mobile agent, which can be integrity protected by the agent owner (or even by the agent creator).

This state appraisal function is evaluated with the agent's state as parameter when the mobile agent arrives at a new agent platform and checks whether the agent's state meets important invariants. Thus, it can protect the agent owner from misuse via dangerous but detectable state modifications that would cause the agent to become harmful even though the code has not been tampered with and is presumed to be benign. Such harmful states may be defined via arbitrarily complex relations between dynamic state variables of the mobile agent.

The authors of [FGS96a] distinguish the special privilege "run" that defines if an agent platform will actually execute a mobile agent. Thus, if the state appraisal function does not request the "run" privilege (since the mobile agent is in an unsafe state), the agent platform will not execute the agent, and thus limit the damage that can be done with a tampered agent.

This approach allows to automatically detect those manipulations that put a mobile agent into an unacceptable state. These manipulations have to be anticipated and verification functions have to be implemented within the mobile agent, which is a difficult task. Provided that the state appraisal functions are actually executed (which will be the case for an honest agent platform), they allow the mobile agent to detect manipulations during its execution.

3.2.4 Co-operating Agents

Approaches that consider to replicate agents in order to leverage ideas from fault-tolerance have been proposed by Roth in [Rot99] and Schneider et al. in [Sch97a, MvRS96]. The common idea of these approaches is to share some information among the agent replicas. The mobile agents, which are now part of some larger application, have to co-operate with each other, in order to achieve the desired result. It is assumed that the shared information is protected, as long as no critical number of agent executors collaborate with each other (this is a generalization of the concept of a trusted third party and constitutes the trusted functionality on which this approach relies).

A straightforward approach is the one by Roth in [Rot99], which assumes the existence of two independent subgroups of agent executors that will not collaborate with each other in attacking the application. The application is then realized with two co-operating agents, each of which migrates exclusively in one of the two subgroups. These two agents can then use secret sharing [Sha79], remote authorization, and remote storage of commitments to protect

[6]It also requires a non-negligible computation to discover a manipulation, but the complete information to detect an attack is available to the agent owner when the agent returns.

the application. For instance, one agent can send a commitment from a service provider for an interesting offer to the second agent. The latter can independently verify that the offer satisfies the requirements of the agent owner, and, if it is satisfied, return its shares of a sufficient amount of electronic-cash to the first agent. Since no entity ever has access to both mobile agents, no other entity can reconstruct the electronic cash shared among both mobile agents or can prevent the respective other agent from returning a commitment back to the agent owner.

A simple but effective method of assuring that the agent executors of the co-operating agents will not collaborate to attack the mobile agent is to have one agent execute on a trusted host while the other agent roams the network. In this degenerate case, the co-operating agent that executes on the trusted host becomes some form of trusted third party.

The approach by Schneider et al. in [MvRS96, Sch97a] also uses replication and cryptographic mechanisms to guarantee that an application that is constructed from several mobile agents is resilient to potentially hostile actions of the agent executors. To counter these, the authors devise replication and voting schemes that can survive various manipulation attacks based on the assumption that only a minority of the visited hosts are malicious. Since the actual protocol executes in several stages where a majority of correct servers in each stage can not guarantee that the shared secret will not be discovered by the combined minorities, the proposed protocol actually deals with an even more complex problem. As Yee points out in [Yee99], even the assumption of a majority of correct processes in a single stage can not necessarily be assumed to be realistic since an attack by some malicious service provider would occur on all the sites of this service provider (i.e., all the agent platforms of a single stage) and not independently.

This approach allows to prevent attacks on the protected (i.e., shared) parts of the mobile agent (this is not limited to state, but can also be applied to code). It relies on the availability of some trusted functionality that can guarantee that an attack is permanently prevented.

3.2.5 Obfuscation

This approach, which is discussed by Hohl in [Hoh98], proposes a special form of protection that is assumed to protect a mobile agent from attacks only temporarily. It is based on *obfuscation* [CTL98], which is a mechanism that transforms an application into one that is functionally identical, but much more difficult to understand and therefore to attack. Since obfuscation can not provide a complete protection, Hohl assumes the existence of a certain minimal time interval (the *agent protection interval*) during which it is impossible for an attacker to discover relevant data or to manipulate the execution of the agent in a meaningful way. An attacker can, at most, apply arbitrary changes to the mobile agent, which result in denial-of-service. After the agent protection interval the agent and all the sensitive information it contains becomes invalid, which prevents the exploitation of attacks after the agent protection interval. An example for this invalidation of information is an electronic coin that contains an expiration time. The idea is that the coin has to be used before it is expired – after the expiration time it will not be accepted any more (this obviously requires synchronized clocks between the principal that applies the expiration time and the principal that verifies it).

The major problem of this approach is to establish a reasonable agent protection interval, during which the mobile agent can accomplish a meaningful task and that still provides a high assurance in the protection of the sensitive data. Another technical problem is that due to the technology used in the approach, a mobile agent can not use external libraries of the agent platform to process sensitive data (this would defeat the purpose of obfuscation). Thus, a mobile agent has to contain the code of all the security relevant libraries (e.g., cryptographic library), which also have to be obfuscated.

Since the protection of the mobile agent and thus of the data it contains is only temporary, this approach can only be used to protect data that can easily be invalidated – it is not appropriate for protecting long lived data (e.g., credit card numbers or address information). The advantages are that it does not require the existence of trusted functionality and that it protects the entire agent with software-based mechanisms.

3.2.6 Tamper-resistant Hardware

This approach, which relies on the existence of powerful tamper-resistant hardware that can withstand potent physical attacks (see Section 3.3) has independently been identified by Yee in [Yee99] and by Wilhelm et al. in [WD97, Wil97, WBS98a, WSB99]. We will provide an extensive description of our ideas on how to use tamper-resistant hardware to prevent attacks on mobile agents in the remainder of this chapter.

The approach by Yee, which is based on his previous work on tamper-resistant devices [Yee94, YT95], is only briefly introduced in [Yee99]. The basic idea is to encapsulate the entire execution environment within the tamper-resistant hardware, which is a physically sealed environment. It protects a mobile agent that is executing on this execution environment from any illegitimate access or modification by the agent executor. No detailed information on how mobile agents can be installed on such a protected execution environment or otherwise take advantage of the tamper-resistant hardware is provided.

Provided that the mobile agent can not be attacked before it is installed on the execution environment of the tamper-resistant hardware (see Section 3.4.5 for our solution), the approach allows to permanently prevent all attacks on the entire mobile agent. The major disadvantage is that the possibly expensive hardware has not yet been developed and must also be widely deployed before it can become useful (this is a typical chicken and egg problem).

A more limited approach that only assumes the existence of a very restricted tamper-resistant hardware (e.g., a smart card) has been proposed by Fünfrocken in [FM99]. The idea of this approach is to identify a small part of a mobile agent that will be executed in an execution environment that is protected by tamper-resistant hardware. It is conjectured that this constellation can achieve a better protection for the entire mobile agent even though parts of it are executed in an untrusted environment. The security implications of this approach are rather complex since the agent executor can control the entire communication between the protected and the unprotected parts of the mobile agent[7] (e.g., manipulate the parameters or the results of function calls between the two parts). It could, for instance, be used to realize a protected signature routine that augments messages it signs for the mobile agent with particular, application specific information, such as "this signature is only valid for a single purchase not exceeding $100". This idea, which is adapted from the "Undetachable Signatures" in [ST98], can prevent at least some abuse of the signature routine.

Nevertheless, we believe that the limited approach is only viable if the protected part

[7]A similar problem exists in the context of a secure interaction between a user and his trusted smart card via an untrusted terminal. A malicious terminal could provide completely different data to the user and to his smart card, which could trick the user into authorizing an undesired action (e.g., payment of $5000 instead of $5) [ABKL91].

is in complete control of the mobile agent's actions and uses the unprotected part only as untrusted co-processor. This means that all the operations of the unprotected part have to be verifiable by the protected part. Such an approach could take advantage of protocols similar to the ones described by Stubblebine in [DS98], which allow to securely store data outside of the tamper-resistant hardware.

3.2.7 Computing with Encrypted Functions

A particularly interesting approach by Sander and Tschudin [ST98] promises to provide a protection similar to the one that can be achieved with tamper-resistant hardware relying on software only.

The idea is to encrypt a function f to obtain some other function $E(f)$ that hides the function f. This encrypted function can then be implemented as a program $P(E(f))$, which is interpreted as a mobile agent and sent to the agent executor. The latter can execute the mobile agent $P(E(f))$ on an input value x, which leads to the encrypted result: $P(E(f))(x)$. Since the agent executor does not know what function it actually computed, it can not meaningfully tamper with the code or its execution and is restricted to random modifications (which result in denial-of-service) and replay attacks. The encrypted result is returned to the agent owner who can apply the decryption function, that only he knows, to obtain the desired result of the function f applied to the input value x: $E^{-1}(P(E(f))(x)) = f(x)$.

In its current form their approach is still quite limited since it does not allow the evaluation of arbitrary functions but is restricted to polynomials and rational functions. Another limitation is that the model presented in [ST98] does not allow for continuous interaction between the encrypted function and its executor, i.e., the function can not compute a clear-text result that is then used by the executor to provide new inputs – all the outputs of the encrypted function are themselves encrypted. Finally, since an attacker can execute the encrypted function arbitrarily often in a controllable setting he may be able to collect sufficient data to recover the original function despite the encryption. Nevertheless, if the approach can be extended to arbitrary functions that are protected with strong encryption schemes, then it has the potential to permanently prevent all attacks on the entire mobile agent or, at least, on the parts that are encrypted. This is similar to what can be achieved in the previous approach but without the need to deploy expensive hardware on a large scale.

3.2.8 Evaluation

The presented approaches provide very different protection properties at varying costs. For any specific application that is based on mobile agents, it has to be evaluated what level of protection for the mobile agent is required.

In particular it has to be decided if the mere detection of manipulation that leaves the agent executor with complete access to the mobile agents code and state is sufficient. For instance, if the amount of money that can be stolen from a mobile agent is too small to justify legal action, if law enforcement is too difficult (see Section 1.6.1), or if the mobile agent must have access to confidential information that should not readily be available to the agent executor, then the use of prevention mechanisms becomes necessary.

The application that we are interested in is the protection of the data agent in the context of privacy protection (see Section 1.6.4). According to the requirements that we have identified, which mandate the permanent prevention of disclosure of a mobile agent's data and prevention of tampering of an agent's code (that could lead to the disclosure of its data), the only adequate solutions are tamper-resistant hardware and computing with encrypted functions. An approach that requires only the protection of critical parts of the agent, which could then be based on co-operating agents, would be possible. This would correspond to the solution based on a TTP that was discussed in Section 1.6.2. Due to the close interaction between the agent executor and the data agent we were concerned with problems of scale, bandwidth, latency, and intermittent connectivity. Therefore, this approach was not retained. Finally, since we were, at the start of this work, not aware of the potential of computing with encrypted functions, which was proposed only recently, we naturally pursued an approach based on tamper-resistant hardware.

3.3 Tamper-resistant Devices

The notion of tamper-resistance usually applies to a well-defined hardware device, sometimes called black-box, that contains a certain amount of sensitive information and executes a given task. The outside environment cannot access this sensitive information or interfere with the task of this device. Any interaction with the tamper-resistant device must pass through a restricted interface, which is under the control of the tamper-resistant device. Any attempt to access the internals of a tamper-resistant device should lead to the destruction of the sensitive information it contains. Examples for tamper-resistant devices that are widely used today are the following [AK96, YT95, And94]:

- meter devices for postage, electricity, gas, or water;
- smart cards for payment systems, GSM mobile phones, pay-TV installations, or access control systems;
- security modules for payment authorization in the banking industry;
- cryptographic coprocessors for protected and authenticated communication;
- missile sensors for the supervision of access to nuclear weapons; and
- seismic sensors for the supervision of the nuclear test ban treaty.

This proliferation of tamper-resistant devices has led to the *Federal Information Processing Standards Publication* (FIPS PUB) 140-1 [NIS94], which is a standardization for cryptographic modules, as well as to the creation of certification services that verify compliance to the standard. The more advanced tamper-resistant devices usually contain a processor, memory, tamper detection circuitry, and a battery. All of these components are either contained in some form of safe, which makes them easy to maintain but vulnerable to insider attacks (e.g., security modules for payment authorization), or are arranged on a circuit card that is wrapped in many layers of fine wire and embedded in hard, opaque epoxy (e.g., cryptographic coprocessors). If the system detects any form of tampering (e.g., a physical attack or an environmental attack, such as fluctuations in voltage or temperature), it destroys all the sensitive information it contains. Recent systems, such as the IBM 4758 PCI Cryptographic Coprocessor [IBM97] have been constructed to meet the highest security requirements of FIPS PUB 140-1.

The usage of these devices is often assumed to provide a comprehensive and high level of protection, which is not always justified [AK96, AK97]. Given sufficient time and resources, it becomes probable that an attacker can violate the physical protection of the tamper-resistant hardware device. Furthermore, it is sometimes possible to extract secret information from a tamper-resistant device without violating the physical protection using electromagnetic radiation attacks[8], such as the ones described by Kocher [KJJ98]. Therefore, the level of tamper-resistance offered by any particular hardware device can be measured by the time and cost penalty that the protective mechanisms impose – these should ideally be higher than the possible gain that can be achieved by breaking the tamper-resistant hardware device.

The most difficult task for a tamper-resistant hardware device is if it must permanently operate in an untrusted environment. As explained by Anderson et al. in [AK96], such systems are very difficult to build. However, if the goal is to merely detect tampering with a high probability and if challenge inspections can be performed to verify that no tampering has occurred, then the task becomes more tractable. It seems very unlikely that a physical attack on systems such as the IBM 4758 [IBM97] or the DS5002FP (as described in [AK96]) can be conducted without leaving detectable traces. Yet, this does not protect the device from protocol or electromagnetic radiation attacks that do not need to overcome the physical tamper-resistance [AK96, KJJ98]. Both of these have to be addressed with careful design of the software and the cryptographic algorithms, such as described by Kuhn et al. in [KA98]. The latter class of attacks based on electromagnetic radiation can further be addressed with physical shielding.

Despite all the problems of the approach, it still is a well understood technology that can, if properly used, considerably improve the security of a system. This is acknowledged in several more critical discussions of tamper-resistant technologies [PPSW97, Kuh97].

3.4 Realization

We will now describe the tamper-resistant hardware device on which our approach is built. This device will be referred to as the *trusted processing environment* (TPE), since it contains the execution environment for mobile agents and must be trusted by the agent owners. According to our definition of trust in Section 1.5, all trust should be tied to a policy. In the case of the TPE, this policy is explicitly defined for and enforced by the TPE (see Section 3.4.2). This means that the TPE allows us to realize a pessimistic approach to trust with respect to the operator of the TPE, who can not violate the policy of the TPE since it is realized as a tamper-resistant device. On the other hand, the trust in the actual enforcement of the TPE's policy must be derived from yet another principal, the *TPE manufacturer*, who designs and produces the TPEs. This, in turn, is based on an optimistic approach to trust and will further be discussed in Section 3.5.1.

An agent executor can buy a TPE from a manufacturer (this makes it a *TPE owner*) and advertise this fact, together with the policy that is enforced by the TPE, to potential users (i.e., agent owners) with the help of a broker. The broker is a simple directory service that allows to locate other principals and to obtain their credentials. Figure 3.1 gives an overview of the principals in the system.

[8]The technology required to conduct or prevent such attacks is often referred to with the term TEMPEST, which stands for *transient electromagnetic pulse emanation standard.* However, due to the strong military implications of this issue, it is difficult to obtain comprehensive information [McN].

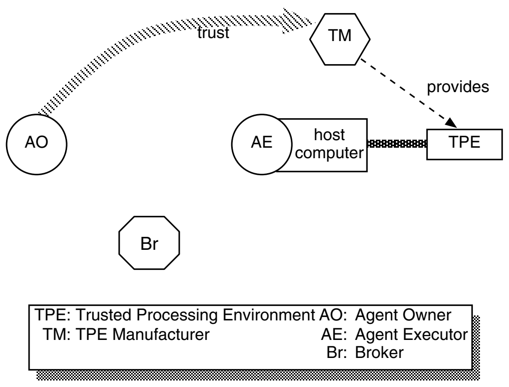

Figure 3.1: Overview of the Principals in the CryPO protocol

If an agent owner wants to use the services offered by some agent executor, he has to obtain the credentials for the agent executor's TPE from the broker or directly from the agent executor. Based on the information contained therein, he can decide if the TPE's policy is adequate and whether the TPE manufacturer (*not* the agent executor) is considered trustworthy.

Once the agent owner has decided to send the mobile agent, he needs a means to safely transfer the mobile agent to the agent executor's TPE. This is achieved via the *cryptographically protected objects* (CryPO) protocol (see Section 3.4.5).

3.4.1 Notation

The described approach relies on public key cryptography [DH76, RSA78]. A detailed description of cryptography and the corresponding notations is not within the scope of this presentation. For information on this topic see e.g., [MvOV97, Sch94, Bra88]. The notation we will use is as follows.

We will use uppercase characters for the keys and algorithms of a public key cryptosystem and lowercase characters for the corresponding keys and algorithms of a symmetric key cryptosystem. The ";" denotes concatenation.

A principal P has a pair (or several pairs[9]) of keys (K_P, K_P^{-1}) where K_P is P's public key and K_P^{-1} its private key. We assume the existence of a public key infrastructure, such as X.509 [ITU93] or SPKI [EFL$^+$98], that allows to establish the association of a principal's identity with the corresponding public key by means of certificates. These are statements that express a certain fact, which are signed by some authority that asserts the accuracy of this fact[10].

Given the proper keys and the corresponding algorithms, it is possible to encrypt a message m using the receiver P's public key K_P, denoted $\{m\}_{K_P} = m'$, such that only P can decrypt it with its private key, denoted $\{m'\}_{K_P^{-1}} = m$. A signed message, including a digital signature

[9]It is advisable to have, at least, two pairs of keys, one for encryption/decryption and one for digital signatures.

[10]For instance a Certification Authority (CA) will thoroughly check the identity of a principal (e.g., by verifying a picture ID) and then issue a certificate that asserts that the CA has conducted this check and binds the public key of the principal to his public key.

on the message m, generated by P using its private key K_P^{-1} and verifiable by anybody using the respective public key K_P, is denoted $\{m\}_{K_P^{-1}}$.

In the following we assume the usage of optimization schemes for encryption and signing. Whenever we need to encrypt a large message, we will encrypt it with a symmetric cryptosystem using a randomly generated symmetric key, which in turn is encrypted using the public key cryptosystem. To encrypt m using K_P, we create k_{rnd} and compute $\{k_{rnd}\}_{K_P}; \{m\}_{k_{rnd}}$, which is also written as $\{m\}_{E_P}$. Additionally, we will use a hash algorithm h to reduce the computational complexity of signing a message m, which means we calculate a cryptographic hash and apply the signature only to the hash of the message. To sign m using K_P^{-1}, we compute $m; \{h(m)\}_{K_P^{-1}}$, which is also written as $\{m\}_{S_P}$. For ease of presentation, we will not make this explicit.

3.4.2 The Trusted Processing Environment

The *trusted processing environment* (TPE) that is at the center of our approach is depicted in Figure 3.2. It is an advanced tamper-resistant hardware device according to the description given in Section 3.3, that should satisfy security level 4 of FIPS PUB 140-1 [NIS94] (this level foresees that the device can operate in a physically unprotected environment). According to currently available technology, this requirement excludes a smart card based system and mandates at least a PC-card (e.g., PCI or PCMCIA) based hardware, which can be operated in a regular PC or workstation.

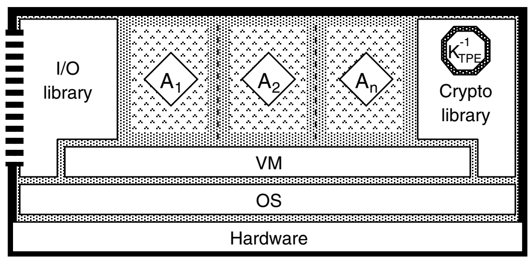

Figure 3.2: The trusted processing environment

The TPE embodies the functionality that we have identified for the agent platform in Section 2.4.2. It is a complete microcomputer that consists of a CPU, RAM, ROM, and non-volatile storage (e.g. EEPROM or flash RAM). The main task of this hardware is to run a virtual machine (VM) that, in turn, provides the execution environment for mobile agents. The VM is governed by an underlying operating system, which coordinates and controls the resources of the TPE, namely the CPU, the access to memory (which can provide an additional protection of agents from each other), the I/O library, and the cryptographic library. We will collectively refer to this essential software on the TPE that has to be installed by the TPE manufacturer as *firmware* (see Section 3.4.3).

The I/O library connects the TPE to a host computer that is under the control of the TPE owner. It provides well defined interfaces that allow the host computer, for instance, to initiate the following management operations on the TPE:

- upload, interrupt, or remove agents;

- facilitate interactions between processes on the host and agents on the TPE; and
- observe certain activities of the TPE (such as which agents are currently executing).

The cryptographic library contains a private key K_{TPE}^{-1} that is known to no principal other than the TPE – even the physical owner of the TPE does not know the private key. The secrecy of this private key is a crucial requirement for the task of the TPE and must be ensured with utmost certainty (see Section 3.4.4).

All these components are physically protected by the tamper-resistance of the TPE and any access or manipulation is mediated and restricted by the operating system, which must provide a very high level of protection. This configuration makes it impossible to access or manipulate the information that is contained in the TPE without proper authorization and allows the TPE to protect locally executing agents from disclosure and manipulation. A closer analysis of this last statement allows us to distinguish between the fundamental protection guarantees of a TPE, which have to be offered by every TPE (i.e., every access to or manipulation of information on the TPE is controlled by the operating system, see Section 4.2.1), and the policy that is enforced by a TPE (e.g., the protection of locally executing agents[11]).

The former is concerned with what protection the TPE is capable to provide. The fundamental protection guarantees are ensured by the TPE manufacturer, which designs and produces its TPEs with care, and declared in a certificate $Cert_{TPE}$ (that is signed by the TPE manufacturer). This certificate contains essential information about the TPE, such as its properties (e.g., size of non-volatile memory), its manufacturer, its type, and its public key. It conveys a level of assurance in the validity of the fundamental protection guarantees, which is dependent on the employed technology for the construction of the TPE. We assume that this will be a major subject for competition among the TPE manufacturers, but since this is mainly dependent on hardware issues, we will not discuss it any further – despite its importance for real implementations,

The latter is concerned with what protection the TPE is configured to provide. This is specified in the policy[12] of the TPE, which is a set of rules that will be enforced by the firmware of the TPE (i.e., primarily the operating system). This policy can be configured by the TPE owner according to his particular needs and within the limits of what the TPE and its firmware support. If this flexibility is not required, the policy could also be specified by the TPE manufacturer, implemented statically in the firmware, and included in the certificate of the TPE. This approach would remove some of the complexity related to the management of policies, but may be too inflexible for many applications.

In order to guarantee the enforcement of the policy, the following two conditions must be satisfied: (1) the TPE must be informed about the policy and (2) it must be capable to enforce every rule defined in the policy. The first condition requires that the policy is passed to the TPE, where it will be analyzed in order to verify if the second condition is satisfied. If this is the case, the TPE will store the policy and sign it with its private key (without this signature the policy is not valid). Once the TPE has signed a policy, it will not accept to sign another one that is weaker than its current policy. This last issue may be too restrictive,

[11]Even though the TPE is tamper-resistant, the operating system could simply give the TPE owner full access to all its data.

[12]The specification for a policy should be formally defined and easy to analyze within a program. We will present a simple specification in Appendix B, but use an informal, natural language specification for the discussion.

since it would prevent a TPE owner from ever weakening the policy of its TPE, and can be resolved by assigning an incarnation number or, if the TPE can rely on a trusted time service, a validity period to a policy. The second condition has to be decided based on the properties of the TPE and the available resources. The enforcement relies primarily on the fundamental protection guarantees of the TPE, which allow the firmware, for instance, to implement the following simple rules that are in some similar form part of every policy:

- The code of a mobile agent will not be disclosed or altered.
- The code of a mobile agent will be executed correctly (according to the definition of the used language, e.g., Java bytecode).
- The state of a mobile agent can exclusively be accessed and manipulated through the interface of the mobile agent.
- A mobile agent is protected from any interference from other agents on the same TPE (other than calls on its public interface).

This simple policy, which we will later identify as the basic protection, does already provide a rather comprehensive protection for the mobile agents executing on a TPE that enforces it. Since it allows a mobile agent to effectively hide parts of its state, it can carry cryptographic keys that can not be accessed by the agent executor (the TPE owner). This enables the mobile agent to send and receive encrypted messages, to sign messages, and to verify signatures on messages it receives (possibly using the cryptographic library on the TPE for the computation). Nevertheless, if the TPE disposes of more elaborate resources (e.g., a battery powered clock) or if the firmware on the TPE can rely on other (as yet unspecified) services, it is possible to guarantee the enforcement of more interesting policies. This will extensively be discussed in the following Chapter 4.

3.4.3 Controlling the TPE Manufacturer

In order to ensure a proper operation of TPE manufacturers, they should be tightly controlled by independent external auditors. This consists of a control of the hardware and of the firmware installed on a TPE, similar to the certification procedures defined in existing hardware evaluation criteria, such as TCSEC [DoD85], ITSEC [CEC91], or the Common Criteria [Com98].

A TPE of some manufacturer should contain the TPE manufacturer's public key, several public keys of renowned auditors, a small bootloader, and a signature verification algorithm in the ROM. The correctness of this ROM can be controlled by occasionally checking it in a randomly chosen TPE (the tamper-resistance of the TPE cannot prevent a determined and well equipped attacker from gaining access to the contents of this ROM).

The bootloader is responsible for installing the firmware on a TPE, which consists of all the components discussed in Section 3.4.2 that should be upgradeable (i.e., the VM, the operating system, and the various libraries), but also software that is not permanently needed (e.g., key management algorithms) or any other software that the TPE manufacturer does not want to put in the ROM. Before installing the firmware, it must be archived and certified (i.e., signed) by the external auditors – an actual code review might not always be possible but can later be conducted on the archived copies. The bootloader will then install this firmware only if it is certified by the TPE manufacturer and by a sufficient number of external auditors.

This scheme cannot prevent that the TPE manufacturer introduces an intricate Trojan Horse in the design of its TPE that can overcome the described mechanism (e.g., an obfuscated part of the firmware or a hidden feature of the hardware). Nevertheless, it makes this task very difficult and expensive for the TPE manufacturer and would later on allow a specialized auditor to analyze and prove an offense.

3.4.4 Key Management

One of the most important properties of the TPE, which has to be guaranteed with utmost certainty, is that the private key of the TPE (K_{TPE}^{-1}) is never disclosed. To ensure this property, the TPE should create this key itself based on random input strings from several different sources (e.g., TPE manufacturer, TPE owner, or an independent third party), which should be combined into one string that is completely random and secret if at least one of the input strings is (see [PPSW97]). This initialization is a highly critical operation that should be executed under the control of certified firmware that can be loaded on the TPE and erased afterwards. Once a TPE has generated its private key, it should be prevented from ever generating (or otherwise accepting) another one. Using this approach, the private key is never available outside of the TPE and, thus, protected by the tamper-resistance of the TPE. The proposed mechanism is compatible with the cryptographic key management described in FIPS PUB 140-1 [NIS94].

The problem with this approach is that if the TPE is destroyed, all the data (that might have been backed up in encrypted form and still be available) is also lost. To make the private key of the TPE recoverable, it must be stored independently from the TPE. To achieve this, the TPE could encode its private key after it is generated in several key shares according to some (n, k) threshold secret sharing scheme [Sha79] and encrypt these with the public keys of n independent trusted third parties[13] (escrow agencies). These escrow agencies have to store their share of the TPE's private key and it is sufficient to combine k out of these n shares, to recover the private key of the TPE and to decrypt its data. The escrow agencies have to ensure that this procedure is only initiated if the TPE was actually destroyed and that the recovered key cannot be abused. A possibility to ensure this, is to not actually provide the private key of the TPE, but to reconstruct it in a secure place, where the encrypted data of the TPE is decrypted and reencrypted with the public key of a replacement TPE. This replacement TPE has to provide at least the same physical protection as the destroyed TPE and would enforce the same policy.

3.4.5 The CryPO Protocol

The *cryptographically protected objects* (CryPO)[14] protocol is concerned with transferring a mobile agent to a TPE and should provide a protection to the mobile agents in transit that is at least as good as that of the receiving TPE. To achieve this, it transfers mobile agents exclusively in encrypted form over the network, using the receiving TPE's public key. Provided that the corresponding private key is only known to the receiving TPE, no other principal can access the code or state of such a protected agent. To prevent any tampering

[13]The transfer of these public keys to the TPE is again highly critical and should be controlled by certified firmware of the TPE manufacturer.

[14]We had originally chosen the term object since it is more general than the term agent. Also, if pronounced in German it sounds like “kripo”, which is a colloquial term for “Kriminalpolizei” – a police force that can loosely be compared to the FBI – should I put this???

with the encrypted mobile agent (or rather to detect and ignore tampered mobile agents) it is sufficient to concatenate the mobile agent A with a hash of the entire agent $h(A)$, including its execution state, before encrypting it $\{A; h(A)\}_{E_{TPE}}$. The receiver simply has to verify the correct hash before starting the mobile agent to ensure the integrity of the received agent.

The protocol is divided into two distinct phases. The first phase consists of an initialization, which has to be executed once before the execution of the second phase of the protocol. The second phase is concerned with the usage of the TPE and the actual transfer of the mobile agent. The protocol is based on the interactions given in Figures 3.3 and 3.4.

In order to send a mobile agent to an agent executor, the agent owner needs some information that we assume will be gathered in a structure that we call a *reference* to an agent executor. This structure consists of the name of the agent executor, its physical address in the network, the policy of its TPE, and the certificate $Cert_{TPE}$ for its TPE. Moreover, we assume that the agent owner can verify the validity of a public key with the help of the public key infrastructure.

Phase 1: Initialization

In the initialization phase, the participants exchange the required key information (see Figure 3.3):

- The TPE manufacturer publishes its certification key K_{TM}.
- The TPE manufacturer sends the certificate $Cert_{TPE} = \{..., K_{TPE}\}_{S_{TM}}$ to the agent executor.
- The agent executor registers its reference Ref_{AE}[15] with one or several brokers.

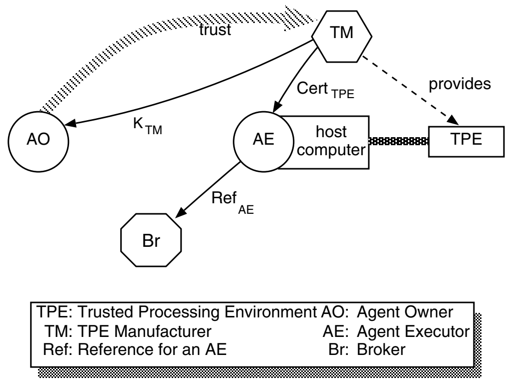

Figure 3.3: Initialization of the CryPO protocol

Phase 2: Usage

After the participants have finished the initialization, they can execute the main part of the CryPO protocol, which is concerned with the transfer of a mobile agent (see Figure 3.4):

[15]The broker can also verify that the agent executor actually controls the corresponding TPE by executing a challenge-response protocol with the TPE via the agent executor.

- The agent owner queries the broker for the reference to the agent executor with which it wants to interact (or it already holds this reference from a previous interaction).
- The agent owner verifies whether the TPE's policy is adequate and controls 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 mobile agent. If any of these checks fail, the agent owner will abort the protocol.
- Otherwise, the agent owner sends the mobile agent encrypted with the public key of the TPE, $\{A; h(A)\}_{E_{TPE}}$, to the agent executor.
- The agent executor cannot decrypt $\{A; h(A)\}_{E_{TPE}}$ nor can it do anything other than upload the agent to its TPE.
- The TPE decrypts $\{A; h(A)\}_{E_{TPE}}$ using its private key K^{-1}_{TPE} and obtains the executable agent A, which will eventually be started and can then interact with the local environment of the agent executor or other agents on the TPE.
- When the mobile agent has finished its task, it can simply migrate back to its owner (protected with the public key K_{AO} of its owner) $\{A; h(A)\}_{E_{AO}}$ or continue its itinerary as explained below.

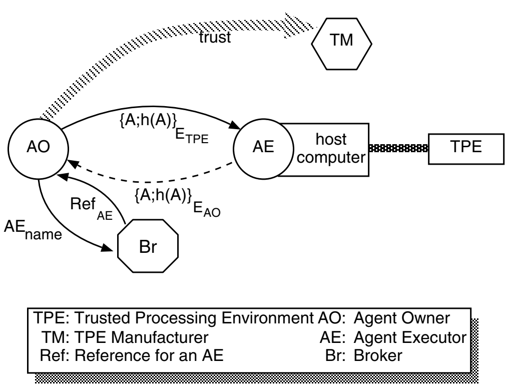

Figure 3.4: Usage of the CryPO protocol

If the mobile agent is appropriately supported by the TPE, it can follow the same protocol when it wants to migrate from the TPE on which it is executing, say T_1, to another TPE, say T_2. For reasons of simplicity we assume that the agent owner has defined a fixed itinerary and provided the mobile agent with the required references (otherwise the mobile agent needs the certification keys of trusted TPE manufacturers, has to autonomously discover the references for appropriate agent executors, and must implement the verification of the policy and the certificate). When the agent wants to migrate, it calls the appropriate service of T_1 with the public key and the network address of T_2 as parameters. T_1 will then take the necessary actions for an agent migration (which are dependent on the underlying mobile agent system), encrypt the marshaled agent with the public key of T_2 and send it to the provided network address. The receiving TPE can deal with the mobile agent as usual.

The obvious problem of protecting the TPE from malicious agents (as opposed to simply tampered agents that were discussed above) is independent of the described approach and has to be tackled with the same mechanisms that are used in other mobile agent systems (see Section 2.5.6). The concept of the TPE does not provide any particular support for this problem other than through the fact that it is a dedicated hardware that was constructed by a specialized principal explicitly for the execution of mobile agents. This constellation could make the implementation of protection mechanisms simpler or more efficient, but this is not within the scope of this thesis.

3.5 Problems

The TPE and the CryPO protocol described above allow us to protect the code and state of a mobile agent from disclosure and manipulation, both in transit and during execution. This protection hinges on the trustworthiness of the TPE manufacturer and is also constrained by other limitations of the approach. In the following we will illustrate some of these problems.

3.5.1 Trust in the TPE Manufacturer

The complex architecture built from TPEs and the CryPO protocol can offer an extensive protection for mobile agents, provided that all the components operate according to their specification. In order to rely on the enforcement of a TPE's policy, an agent owner has to trust the TPE manufacturer to properly design, implement and produce its TPEs. Due to the complexity of this task, it is impossible to enforce this. The TPE manufacturer can always introduce hidden backdoors (Trojan Horses) in the TPE, which would allow it to access any information on its TPEs. Therefore, this problem can only be addressed with an optimistic approach to trust that relies on controls and sanctions. Since this optimistic approach to trust could also be applied to the agent executors directly, it seems that the presented architecture 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 agent executor, which is the operator of a TPE, with trust in the manufacturer of this TPE has several more subtle implications. These result in the following advantages that we have identified:

- high expertise,
- resources to build reputation,
- separation of concern, and
- effective control.

The TPE manufacturer is a specialized principal, which primarily deals in the field of the provision of security devices. Therefore, it has a profound understanding of security and privacy problems and the potential pitfalls. This makes it a much more capable entity to ensure a good protection. The regular TPE owner, on the other hand, is primarily concerned with his core business and does not necessarily have the proper expertise to ensure a secure operation of its hardware and to guarantee the protection of the processed data.

The production of TPEs is considered to be a difficult task (see Section 3.3), which will only be undertaken by major corporations. Since these have the necessary resources to build a good reputation and an incentive to protect it, we can rely on this reputation as a foundation for the trust in the TPE manufacturer (see Section 1.6.1). This allows the TPE owner, which can be an individual or a small company without any particular reputation (and therefore none to lose), to leverage the trust that potential customers have in the manufacturer of its TPE over to his business relation with the customer.

The TPE manufacturer, which is responsible for the correct operation of its TPEs, has nothing to gain from not accomplishing its task. Since the TPE will be operated by another principal that works in a completely different business, the TPE manufacturer has no direct interest in the data that is processed on its TPEs and therefore no simple possibility to draw a direct benefit from a TPE that does not enforce its policy (it could only collaborate with the TPE owner or sell the data to a competitor of the TPE owner). This is obviously not the case for the TPE owner, who has an immediate interest in all the data that is processed on its TPE and who might have short term goals that (in his point of view) justify a policy violation.

Finally, we assume that there will be relatively few TPE manufacturers[16] (on the order of a few hundreds) compared to the number of possible operators of a TPE (on the order of several millions). This makes the control of the operation of the TPE manufacturers much easier for external auditors, as explained in Section 3.4.3. Even if such stringent controls might not be obligatory, it is conceivable that a TPE manufacturer might do this voluntarily, in order to obtain a better position in the market (similar to the approach for quality assurance in the ISO-9000). A comparable control of the TPE owners would be impossible due to the sheer number of principals.

3.5.2 Limitations

The presented approach does have some serious limitations, which are all concerned with the use of tamper-resistant hardware devices. These devices are perceived as:

- difficult to construct,
- a possible performance bottleneck,
- difficult to maintain, and
- expensive.

The most problematic of these limitations is the actual construction of tamper-resistant devices that can be operated in an untrusted environment. This task is extremely difficult and an attack can at most be made time consuming and expensive (see Section 3.3). If a TPE owner would succeed to obtain the private key of his TPE, he could pretend to run a TPE (using the certificate for the TPE of which he has the private key), while he is actually capable to access all the information in the mobile agents sent to his TPE, to manipulate them at will, and to completely ignore the TPE's policy.

Under the assumption that the key can not be accessed without physically tampering with the TPE, this could be countered with regular inspections of the TPE by independent

[16]There are currently three producers of smart card technology that control more than 80% of the market (Gemplus, Schlumberger, and De La Rue).

third parties (such as the TÜV[17]) that can detect such tampering[18]. If this level of security is not sufficient, this could be improved with unannounced challenge inspections of the TPE. This would at least limit the amount of time during which an attacker can benefit from a successful attack and it could provide an effective deterrent if an attempted or successful breaking of a TPE is severely punished. However, for highly sensitive installations, where eventual detection of tampering is not sufficient, other approaches have to be devised (e.g., continuous video surveillance).

Since the TPE is necessarily involved in all computation related to mobile agents, it could easily become a performance bottleneck in the installation of some service provider. This is an extremely difficult problem that requires a more thorough analysis. The only simple way to address this problem is to overdimension the system so that it can cope with most high load situations. This might not be an option for small companies who have a tight budget, but these are the ones that have the most to gain from the availability of a TPE.

The problem of maintaining and upgrading a tamper-resistant device that is by design inaccessible to its owner is grounded on an actual dilemma. It can only be alleviated by establishing a suitable maintenance contract with the TPE manufacturer, in which it ensures a proper operation of a service provider's TPE installation. Again, such a maintenance contract may not be available to small companies.

Finally, the often cited argument that tamper-resistant hardware is expensive, must be weighed against the advantages of the approach. As it is quite likely that a large installation would anyway require some dedicated hardware, the TPE would only make this marginally more expensive. On the other hand, since for a small company, that tries to establish itself in the market, some interactions with its customers would not be possible without a TPE, this might be a reasonable price to pay.

3.6 Open Systems

If agent based systems are to become a reality, they have to provide security not only for the agent executor and its installation, but also for the mobile agents themselves. This is due to the fact that truly useful mobile agents often rely on confidential information that the agent owner would rather have protected.

If a principal would like to create an agent based service, in which a mobile agent needs confidential information, then it needs to convince potential customers that it will not abuse this information. This may be very difficult for a small and unknown company. 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 customer especially if the provider is located in a different country with an unknown legal system.

With the presented approach, this principal can simply buy an appropriate TPE (i.e., one that convinces potential customers that their mobile agents will be sufficiently well protected) from a reputable TPE manufacturer, configure the policy that it would like its TPE to enforce, and advertise this to potential customers. It can then immediately benefit from the trust that customers have in the manufacturer of its TPE in order to convince them that it will not

[17]The *Technischer Überwachungs-Verein* is an independent, government-approved expert appraisal and inspection organization.

[18]As explained in [AK96], a tamper-resistant device that requires only detection of tampering is conceivable to even resist massive attacks.

maliciously abuse agents sent by the customers and that the agent's confidential information is sufficiently well protected. This allows for simpler bootstrapping of trust and favors the open systems philosophy, where any principal can potentially become a service provider.

∞∞∞

Summary

In this chapter we identify the problem of protecting a mobile agent from its execution environment or, more precisely from its operator, as an important issue. We then analyze the existing approaches from the literature and propose a classification that identifies various properties that these approaches exhibit. Based on the requirements that were identified for the data agent in Chapter 1 (prevention of access to data, prevention of manipulation of code and data, and permanent protection), we choose to pursue an approach based on tamper-resistant hardware. We examine the general possibility of using tamper-resistant hardware and develop our particular approach which relies on the *trusted processing environment* (TPE) and the *cryptographically protected objects* (CryPO) protocol.

The entire approach hinges on the trustworthiness of the TPE manufacturer which is discussed together with other limitations. Finally, we point out the relevance of the approach for the development of open mobile agent systems.

Chapter 4

Usage and Extensions

Everything should be made as simple as possible, but no simpler.

– **Albert Einstein**

4.1 Introduction

In the preceding chapter we have shown how a mobile agent can be protected from the agent executor with the help of a tamper-resistant TPE. This protection is primarily concerned with preventing access to or manipulation of the mobile agent and guaranteeing its correct execution, which constitutes the basic protection that we have already identified in Section 3.4.2 and that will be further developed in Section 4.2.1

In this chapter we explore how the basic protection offered by the TPE and the CryPO protocol can be used within the mobile agent paradigm and extended to achieve more elaborate goals that may be desirable for the agent owner. This will often require additional hardware guarantees from the TPE and/or the existence of TTPs that provide a certain service. Moreover, it is necessary that both the firmware of the TPE and the TTP (if one is required) adhere to and enforce additional rules that have to be clearly stated in a policy. The desired goal can then be achieved via a collaboration of the different components (i.e., the mobile agent, the hardware, the firmware, and the TTP). We will refer to all of the presuppositions that have to be combined in order to achieve a particular goal as a *conduct*. These conducts will be the major topic of this chapter. The ideas for the various conducts have been developed in several publications [WD97, Wil97, WBS98a, WSB99], which explore their use in the context of different applications: general protection of mobile agents, electronic commerce, and telecommunication systems.

Several of the conducts that we will explore in the next section are, in fact, inspired by the client-server paradigm. This paradigm has many commonalities with the mobile agent paradigm and most distributed applications can, in principle, be realized with either one of them [HCK97]. However, while some applications can exploit the advantages of the mobile agent paradigm (see Section 2.6), others may rely on particular safety or security properties of the client-server paradigm that are so basic that one hardly ever thinks of them. These properties allow the provision of certain guarantees to the client part of a distributed application that cannot be provided to a mobile agent (e.g., code will be executed correctly, code can rely on a reasonably reliable time service, code will not be executed more than once, etc.). This is due to the fact that the client implementation resides in a physically trusted environment of a user, where it can rely on the availability of trusted services and, for instance, log any irregularities. These logs can then be provided as evidence in the case of a dispute with some server.

In the mobile agent paradigm, on the other hand, the agent owner has no guarantee whatsoever concerning the execution of his mobile agent (see Section 3.1). Since the agent executor has complete control over the agent platform, it can, for instance, simply delete any logged data. Digitally signing the mobile agent's state can considerably limit the malicious actions of an agent executor but it cannot prevent them (see Section 3.2.2).

With the help of the TPE and the CryPO protocol, we intend to create an environment for mobile agents that allows them to base their execution on guarantees similar to the ones provided by the client-server paradigm, so that it becomes possible for a mobile agent to protect itself from a possibly malicious agent executor.

4.2 Conducts

In the following we will develop several conducts, which are concerned with the realization of different goals that may be desirable for the agent owner. We will describe each of these conducts in a structured way, which consists of:

- a brief introduction into the general idea of the conduct,
- the statement of the goal(s) that should be achieved,
- the identification of the requirements that have to be satisfied for this particular conduct[1],
- the policy, which describes the rules that will be followed by different participants in the conduct, and
- a discussion that explains how the goal(s) can be achieved based on the requirements and the policy; occasionally we will also point out additional uses for this conduct.

The following list of conducts is not assumed to be complete and we expect to expand it in the future with new ones we discover.

4.2.1 Basic Protection

This conduct describes the basic protection guarantees that every TPE must offer and on which all of the following conducts rely[2]. It is exclusively concerned with the TPE and the installed firmware.

[1]If these requirements identify another conduct, say C, as requirement, then this means that not only the requirements of C have to be satisfied, but also that C's policy must be enforced.

[2]It would be a reasonable modification to integrate the following two conducts (migration control and protected invocations) into the basic protection. However, since these are not required for many of the other conducts, and for reasons of clarity we isolate them into separate conducts.

Goals:

- The mobile agent can rely on its own resources (e.g., its code).
- The mobile agent can hide (parts of) its state (e.g., cryptographic keys).

Requirements:

- The TPE is tamper-resistant and its private key has not been compromised.
- The VM correctly implements the specification of the language used by the mobile agent (according to the definition of the used language, e.g., Java bytecode).
- The cryptographic library implements the cryptographic algorithms correctly (encryption, signatures, hashes, key generation, nonce generation, etc.).
- Any access to or manipulation of data on the TPE is under the control of the operating system.
- The operating system can protect a mobile agent from other agents on the same TPE.

Policy:

a) The TPE will not disclose the code of a mobile agent.

b) The TPE will not alter the code of a mobile agent.

c) The TPE will not permit access to the state of a mobile agent other than via the mobile agent's public interface.

d) The TPE will not permit the manipulation of the state of a mobile agent other than via the mobile agent's public interface.

e) The TPE will execute the code of a mobile agent correctly.

f) The TPE will protect the mobile agent from any interference from other agents on the same TPE (other than calls on the mobile agent's public interface).

Discussion

The implementation of this conduct is difficult, but straightforward. The most critical parts are the tamper-resistance of the TPE and the correctness of the firmware of the TPE (see Sections 3.4.2, 3.4.3 and 3.5). Any violation of the physical protection of the TPE or serious problem in the implementation of the firmware of the TPE (i.e., the fundamental protection guarantees of the TPE) will invalidate the basic protection. Obviously the TPE cannot prevent that the mobile agents it receives may have been tampered with while in transit. Nonetheless it can detect any manipulation of the mobile agent's code or state (as explained in the CryPO protocol, Section 3.4.5). In that case it can simply ignore the mobile agent and possibly inform the mobile agent's sender and/or owner of this problem.

A TPE that provides this basic protection already offers a rather comprehensive protection to mobile agents executing on it. Since the mobile agent can rely on the proper execution of its code and effectively hide parts of its state[3], it can implement and enforce an access control policy on the data it contains and disclose data only under its own discretion. Moreover, since the mobile agent can contain cryptographic keys and implement cryptographic algorithms (as an optimization, it may be allowed to use the cryptographic library of the TPE), it can authenticate other principals, sign and encrypt messages it sends, as well as decrypt and verify signatures on messages it receives.

4.2.2 Migration Control

In order for a mobile agent to actually be mobile and to be able to migrate to a different TPE, rules a) and c) of the basic protection's policy, which disallow any disclosure of the mobile agent's code or state, must be relaxed. This relaxation should remain under the control of the mobile agent.

Goal:

- The mobile agent can prevent that it will be executed on a TPE that does not provide sufficient protection (i.e., that does not offer a sufficient level of assurance in its fundamental protection guarantees or that does not enforce a sufficiently strong policy).

Requirements:

- Basic protection

Policy:

a) The TPE will allow agents to leave the TPE only in encrypted form.

b) Prior to a migration, the TPE will provide the certificate and the policy of the designated receiver's TPE to the mobile agent and ask for confirmation.

c) The TPE will honour the mobile agent's decision.

d) The actual transfer follows the CryPO protocol using the public key of the receiver's TPE contained in the certificate.

Discussion

Due to the large variety of TPEs[4], assessing the fundamental protection guarantees of a TPE based on its certificate can be a daunting task for a mobile agent. This may require an extensive support from the TPE or from external appraisal organizations (which provide some simple grading scheme that may even be included in the TPE's certificate). The evaluation of the policy on the other hand, should be supported by some formal specification (see Section 1.5.1 and Appendix B). We assume that the mobile agent contains a minimal policy that ensures that all its security related requirements will be satisfied, which can be compared

[3]A mobile agent that does not act at all, actually hides all of its state. Since we assume that the agent does act and since any action allows an observer to draw conclusions on the underlying state, we assume that the mobile agent can only effectively hide parts of its state.

[4]Even though we assume that the number of TPE manufacturers is rather small, each of them will produce several types of TPEs, which will also evolve over time.

with the policy of the destination TPE to verify that the latter is sufficiently strong. In conjunction the two checks allow the mobile agent to ensure that it will never be executed on a TPE that does not provide a sufficient protection.

Since each TPE has a unique identifier (its public key may be used for this purpose), this conduct also allows the mobile agent to follow a pre-defined itinerary. This itinerary may be given to the mobile agent in the form of a list of references to agent executors (see Section 3.4.5), which among other information contain the certificate of the TPEs. Provided that the mobile agent starts on the first TPE on its itinerary (which can be ensured by the agent owner), it simply has to verify upon each migration that it will only migrate to the next TPE on its itinerary. In Appendix B, we discuss how the mobile agent can, based on a more low-level policy of the TPE, implement most of the migration itself.

4.2.3 Protected Invocations

This conduct is concerned with the optimization of method invocations between mobile agents on the same TPE. Since the TPE is in control of all interactions between locally executing agents, it can ensure many guarantees that make further cryptographic protection unnecessary.

Goal:

- The mobile agent can rely on the confidentiality, integrity, and authenticity of local method invocations.

Requirements:

- Basic protection

Policy:

a) The TPE establishes one or several unique identifiers for each mobile agent on the TPE (this can either be the ID of the mobile agent (see Section 2.4.1) or a temporary pseudonym).

b) The TPE invokes a method exclusively and unaltered on the mobile agent with the ID provided in the invocation.

c) The TPE provides the invokee with the ID of the invoker (this information is consistent).

Discussion

The realization of this conduct is based on regular operating system mechanisms. Two interacting mobile agents can authenticate each other (using any appropriate mechanism) and subsequently interact without any further protection mechanism. No other entity can observe, intercept, or alter the information that is exchanged. A man-in-the-middle attack can be prevented with the help of the unique identifier offered by the TPE, provided that the mobile agents know each other's identity (the identity of the agent owner may be sufficient) or can rely on digital signatures for the authentication.

4.2.4 Authentic Code

For certain applications it may not be sufficient that a mobile agent knows that the agent with whom it interacts, is from a particular trusted entity (this can be verified with a regular authentication), but it might need to know exactly with what code it interacts.

Goal:

- The mobile agent can determine whether it interacts with some given code.

Requirements:

- Basic protection
- Protected invocation

Policy:

a) The TPE provides an operation that, given the ID of a mobile agent, returns the cryptographic hash of the immutable part of this mobile agent (provided that the mobile agent approves).

b) The TPE will query the mobile agent on which a hash is requested if it allows this operation to take place.

Discussion

In order for this conduct to be useful, the agent owner needs to have access to the code of some other mobile agent to analyze its behaviour (this requires either an analysis of the executable code or the use of a deterministic compiler). If he is satisfied with this analysis, he will provide his mobile agent with the hash of the analyzed code. When the agent starts an interaction with some other local agent, it has access to the local identifier and can, thus, obtain the hash of its partner in the interaction. Then the mobile agent can verify that it actually interacts with the code that was analyzed by its owner.

This can, for instance, be used in a pragmatic approach to solve the *Millionaire's Problem* [Yao82], where two principals want to compute who is the richer of both, without either one being able to determine the wealth of the other (besides what can be deduced from the result). Both principals can independently analyze a comparison agent, which is simply a piece of code that authenticates the sources for two input values, compares the two values, and returns the result to the sources. If both of them are convinced that this code computes exactly what it is supposed to compute and has no other side effects, they will proceed. They create a carrier agent that contains their wealth, the hash of the comparison agent, and a private key for authentication and signatures. Both carrier agents and the comparison agent will be loaded on a sufficiently well protected TPE, where they interact. The comparison agent will authenticate the carrier agents to prevent masquerading attacks and the carrier agents will verify that they interact with the comparison agent that was analyzed by their owner. After receiving the result, they sign it with their private key to prevent that it can later be altered.

Another possible use for this conduct could be the implementation of a verifiable mix (see Section1.3.3). The actual implementation of the functionality of a mix is well understood; in its simplest form it has to buffer a certain number of messages (of equal length), transform them (e.g., by decrypting them), reorder them, and forward them to the receivers indicated in the decrypted messages. The only additional complication is that the mix has to ignore duplicate messages[5]. We assume that this functionality is implemented in a mix agent, the code of which is publicly available. This allows the users of the mix to analyze the code and to verify that the mix implements exactly the functionality it is supposed to implement.

This mix agent is then instantiated as a stationary agent on a TPE, where it offers its services.

A message that is to be protected by the mix is encoded in a message agent, which also contains the hash of the analyzed code of the mix agent. When the message agent arrives at the TPE, it will verify that the code of the mix agent is the one that was previously analyzed by its owner and if this is the case, hand over the message to the mix agent for processing.

Since the basic transfer of mobile agents with the CryPO protocol already implements most of the encryption and decryption, the main task of the mix agent is to coordinate the ordering of outgoing message agents and the detection and elimination of duplicates.

4.2.5 Synchronized Clock

For many applications it may be important to have an approximate knowledge of the actual real time. This can be useful for an agent owner, who may want to limit the duration during which a mobile agent can be active. We will refer to this as an agent with a *limited lifetime.*

Goal:

- The mobile agent can obtain a sufficiently accurate time or is informed that no accurate time is available.

Requirements:

- Basic protection
- The TPE contains a sufficiently accurate clock (that can not be influenced by external actions; e.g., made to run slower or faster)
- The TPE cooperates with a trusted time server[6] (TTS).
- The TPE holds an authentic copy of the TTS's public key.

[5]This constitutes a serious problem since a mobile agent can not store a long message history in the limited amount of non-volatile storage that is available on the TPE. It could be addressed by concatenating the message with a random number and reencrypting it with the public key of the designated receiver, which will cause duplicate messages to have a different encoding. Nevertheless, since all duplicate messages will be sent to the same receiver, an attacker may be able to link messages via this information (in particular if the attacker is the TPE owner who can feed the mix agent on the TPE with arbitrary messages and intercept them before they are forwarded to the designated receiver). Probabilistic methods that enable the mix agent to detect duplicate messages with a sufficiently high probability but do not require the storage of the entire message history, may help to further complicate the task of an attacker. This problem requires further study.

[6]The TTS must be trusted by the agent owners – since they have to trust the TPE manufacturer in any case, it seems appropriate to let the TPE manufacturer select the TTSs with which its TPEs will cooperate.

Policy:

a) The TPE will synchronize its clock with the TTS.

b) The TPE will inform mobile agents if this synchronization does not succeed.

c) The TTS will duly reply to all requests with the signed real time.

Discussion

The TPE simply sends a request with a nonce (to prevent replay attacks) to the TTS, which will immediately reply with a message that contains the nonce and the real time signed by the TTS. Upon receiving this message, the TPE knows that its time is accurate within the delay of the request. For most practical cases, assuming that the message is not artificially delayed by an external attacker, this will be within a few seconds, which is sufficient for the intended use (we expect mobile agents to have lifetimes on the order of at least several minutes). If the TPE does not succeed to synchronize its clock, it will inform mobile agents about this problem and periodically retry the synchronization.

With the help of the accurate time from the TPE a mobile agent can now easily implement a limited lifetime. The problem is that once an agent executor obtains a mobile agent, he can store the originally received, encrypted mobile agent and execute it repeatedly for an unlimited duration, which may not be in the interest of the agent owner. To prevent this, the agent owner provides his mobile agent with an expiration time t_{exp}, which limits the agent's lifetime. Upon its arrival the mobile agent inquires if the TPE has a sufficently accurate time, requests the local time of the TPE, and checks if this time is still within its attributed lifetime. If its expiration time has passed or if the TPE did not succeed to synchronize its clock, the agent will abort; otherwise, it will continue its execution. This conduct allows, for instance, to decide that a particular mobile agent has effectively been lost – if its lifetime has expired, it can not unexpectedly reappear at some later time when it may cause damage.

Another possible use of this conduct is to realize a deferred activation, which allows a mobile agent to delay the start of its activity until some time t_{start} that is specified by the agent owner. Upon its activation, the mobile agent requests the local time of the TPE and executes some form of sleep operation that will wake it when its activation time has been reached.

4.2.6 Local Logging

In this conduct we want to further improve on the protection of the previous conduct, by not only limiting the duration, but also the number of times that an agent can be executed. This can entirely prevent replay attacks by an agent executor.

Goals:

- The mobile agent can base its execution on results from previous executions that are logged on the TPE.

Requirements:

- Basic protection

- Synchronized clock
- The TPE disposes of a considerable amount of non-volatile storage that is protected from any external interference.

Policy:

a) The TPE provides a small amount of non-volatile storage up to some expiration time even to a mobile agent that has terminated its execution on the TPE.

b) The TPE will authenticate a mobile agent that allocates such an area of non-volatile storage and store the authentication information with the data (this authentication information can be a public key or the ID of the agent owner who has signed the mobile agent).

c) The TPE guarantees that the data stored by a mobile agent can exclusively be read and/or written by a mobile agent that provides a proper authentication (e.g., one that can prove knowledge of the private key that corresponds to the public key stored with the data or belongs to the same agent owner).

d) After the expiration time, the TPE will delete the data.

Discussion

The implementation of this conduct is very simple and consists of a small database that has to be managed in the TPE's non-volatile storage. This database contains a list of entries that consist of a tag (provided by the mobile agent to access the data), the expiration time, the authentication information, and the actual data. The TPE will regularly delete entries that have expired. The major problem with this is the limited size of the non-volatile storage in the TPE[7], which must be realized with expensive technology (e.g., EEPROM of flash RAM; a harddisk inside the TPE seems currently not realistic). To overcome this problem, the TPE could use the resources of the host computer, which can easily provide several megabytes of storage on a contemporary harddisk.

The obvious problem with this is that the external storage is under the control of the TPE owner and consequently not protected from external interference, which contradicts the requirements. A similar problem has been addressed by Devanbu et al. in [DS98], where they propose to use cryptographic mechanisms to allow a resource limited trusted device to detect integrity violations on stacks and queues that are stored on an external and possibly malicious host computer. The main idea is to create a digest of the data that is held within the trusted device and to apply integrity protection on the externalized data. Whenever the data is read, the integrity protection allows to verify that it has not been tampered with, while the digest allows to verify that the data is current (otherwise the external host could try to provide outdated data). Whenever the TPE notices that its host provides incorrect or outdated data,

[7]Even a trivial example, where an entry consists of 128 bytes (a hash of the tag requires 160 bits, a timestamp 64 bits, the authentication data 512 bits, which leaves 288 bits for the data; i.e., 36 characters or 9 integer values), the average expiration time is 1 hour and the agents on the TPE create 1 entry per second requires 450 Kbytes (128 bytes $*$ 3600 sec $*$ 1 sec^{-1}). A more realistic example with entries of 512 bytes, an average expiration time of 10 hours, and 10 entries per second requires 180 Mbytes (512 bytes $*$ 36000 sec $*$ 10 sec^{-1}), which can not be handled with solid state memory within the TPE.

it will stop working and try to inform some supervising entity of this problem (e.g., the TPE manufacturer). If confidentiality is an issue, the externalized data can be encrypted (since the data will only be read by the TPE, symmetric cryptography is sufficient.)

This conduct allows a mobile agent to control the number of times it is executed on a particular TPE or to selectively disclose data based on what data it has previously disclosed in other executions. For instance, in the example of Section 2.2.2, if the mobile agent contains several payment informations, it could ensure that it will disclose at most one of them.

To take advantage of this, the mobile agent needs to implement a limited lifetime. When it is started on a TPE, it will access the entry that corresponds to a particular tag, in order to find out about possible previous executions on this TPE. The TPE will request the authentication information from the mobile agent (e.g., a public key or the mobile agent's ID) and authenticate the mobile agent against this information. If this is successful and if the authentication information provided by the mobile agent corresponds to the one in the entry (supposing it already exists), the TPE will provide the stored data or inform the mobile agent that no such entry exists[8]. In the latter case the mobile agent will create the entry with an expiration time that is sufficiently later than its own lifetime (this needs to take the accuracy of the clock on the TPE into account). The TPE may, for instance for reasons of unavailable non-volatile storage, refuse to create the entry, which will result in an abort of the mobile agent. After its execution the mobile agent will update the stored data to reflect the result of this execution. This may have to be done before actually disclosing some information to an external entity, to prevent that the mobile agent can be interrupted before performing the update.

Since the TPE will not delete the entry before the specified expiration time, the mobile agent will either be guaranteed to find the logged data or the mobile agent is past its attributed lifetime and will not execute at all.

4.2.7 External Logging

For certain security relevant or otherwise critical actions of a mobile agent it may be desirable to log this action with an external logging server, which is a principal that is trusted by the agent owner. If the mobile agent can also access logs it created previously, this can even be used to coordinate its actions across several TPEs (e.g., limit the number of items it buys at different service providers).

Goals:

- The mobile agent can store information that will be unconditionally recoverable.
- Optionally, the mobile agent can base its execution on previous executions that are logged by the logging server.

Requirements:

- Basic protection

[8]The preceding authentication of the mobile agent prevents some other mobile agent that does not know the authentication information from probing whether a particular mobile agent has recently been executed on this TPE.

- The agent owner cooperates with a trusted logging server (TLS).
- The mobile agent holds an authentic copy of the TLS's public key.

Policy:

a) The TPE allows mobile agents to perform remote invocations.

b) The TLS stores all log entries[9] it receives and replies with a signed acknowledgement.

c) Optionally, the TLS may allow a mobile agent (that is properly authorized) to access log entries.

Discussion

The implementation of this conduct is fairly easy and does not cause unsurmountable storage problems at the TLS (which can rely on inexpensive mass storage media). The mobile agent simply has to prepare the log entry, apply the necessary cryptographic protection (the inclusion of a nonce will prevent replay attacks by the TPE owner), and invoke the log method of the TLS. If the log was successfully stored by the TLS, the TLS will return a signed acknowledgement which can be verified with the help of the TLS's public key.

This can, for instance, be used by the mobile agent to log a contract, which it has negotiated with some service provider, before actually providing the payment. If the mobile agent is lost and cannot return the contract to its owner directly (as the contract is signed by the service provider, this would be sufficient for non-repudiation), the agent owner can retrieve the contract from the TLS.

The optional variant enables a mobile agent to implement the goal described in Section 4.2.6 across several TPEs, which allows the mobile agent to coordinate global actions.

A particularly interesting use of this conduct is the installation of legal backdoors within mobile agents. Assume that a mobile agent contains a data item that it will not disclose via its public interface but only use internally (e.g., a private or symmetric key). However, the mobile agent may offer a special method that will disclose this data item provided that this request is properly authorized (e.g., signed with a private key that is accepted to authorize this action, such as the key of a judge that can order the surveillance of a person). To make this mechanism acceptable for the agent owner, the mobile agent will not disclose the data item before it has logged the operation with a logging server that is trusted by the agent owner (e.g., to inform him about the operation after a certain delay, say a few months), but also by the entity that requested the data item (e.g., to not inform the agent owner earlier). The mobile agent itself must also be trusted by both parties to perform this method correctly. This can be subject to public code review, where the code itself can be authenticated (see Section 4.2.4) or ensured by a trusted agent creator. A similar mechanism for access to medical information has been proposed by Anderson in [And96], but he does not comment on how the mechanism can be enforced.

4.2.8 Anonymity

For certain actions of mobile agents on a TPE it may not be necessary to identify the mobile agent that is responsible for the action. This makes every agent currently executing on the

[9]These may also be marked with an expiration time that allows the TLS to delete unnecessary entries.

TPE equally likely as the initiatior of such an action (except when an agent is controlled by the TPE owner). To allow external entities to interact with an anonymous mobile agent, the TPE can provide temporary pseudonyms, which are exclusively known to the TPE.

Goal:

- The mobile agent can perform actions anonymously while it is executing on the TPE

Requirements:

- Basic protection

Policy:

a) The TPE allows mobile agents to act anonymously from the TPE.

b) The TPE offers one or several temporary pseudonyms for mobile agents, with which external entities can interact with a mobile agent.

Discussion

Since the TPE is in control of all the information that is communicated to the host computer, it can simply omit any information on the identity of a mobile agent that does not want to be identified. This does not preclude a proper authorization of the actions of a mobile agent, which can still be performed by the TPE or a special agent on the TPE. This can consist of an authentication of the mobile agent and the consultation of some access matrix or of more elaborate methods [BFL96]. The TPE owner will simply be informed whether this authorization was successful.

This conduct allows a mobile agent on the TPE to execute a database query without being identified to the TPE owner, which may, for instance, be useful for the patent search that was mentioned in Section 3.1.

Another possible use is to enable mobile agents to send and receive remote invocations anonymously. The mobile agent can be addressed with the temporary pseudonym attributed by the TPE or if the remote entity knows the ID of the mobile agent, it can encrypt this ID (and possibly the entire invocation) with the public key of the TPE. The TPE needs to ensure that it does not disclose any information on whether such an invocation was successfully delivered or not, since this may be used by the TPE owner to discover the destination of the invocation (a required acknowledgement should be created by the receiver).

The problem of anonymity is that it is only meaningful within a sufficiently large set of entities, who may possibly perform an action at a particular point in time. This is often called the anonymity group. If this anonymity group has only a single member at the time when the action is performed, then this member is perfectly identified. Therefore the member should be informed about the size of the anonymity group within which it operates. In the context of the TPE this could be the number of mobile agents that are currently executing on the TPE. However, since all of the other agents on the TPE may be dummy agents of the TPE owner that do not act at all, this metric may not be very helpful. The TPE may compute the number of mobile agents on the TPE from different agent owners, which can offer a better metric for the anonymity of the mobile agent. The entire domain of anonymity of mobile agents is highly complex and still subject to further research.

4.3 An Example: The TTP Agent

We now want to briefly illustrate how the presented conducts can be used to create a service that is simple to use and takes full advantage of the mobile agent paradigm. Many security related protocols, in particular those that deal with non-repudiation, rely on the cooperation with a trusted third party (TTP) [MvOV97]. The role of this TTP can be to provide a well defined functionality (e.g., timestamping or logging) to create non-repudiable evidence that can later be used to resolve a dispute. The special character of this functionality requires that it is provided on a trustworthy environment, which can be guaranteed in the administrative domain of the TTP. Thus, a user of this functionality has to interact with a host of the TTP via a remote interaction, which suffers from the usual problems of limited bandwidth, high latency, and inflexibility of the interface (this also applies to the conduct for external logging described in Section 4.2.7). In order to get rid of this remote message exchange we propose to encapsulate the functionality of the TTP in a mobile agent that can be executed on a TPE, which is also a trustworthy environment. This allows us to gather all interacting parties on a single platform, where they can interact via local method invocations and, thus, take full advantage of the mobile agent paradigm (see Section 2.6.1).

In order to avoid confusion, we will refer to the host of the TTP (which is still required) as the TTP server (TS) and to the mobile agent that executes on the TPE on behalf of the TTP as the TTP agent (TA).

4.3.1 The Shopping Agent

Consider the shopping agent of Section 2.2.2. Once it has decided on a particular service provider, it will negotiate a contract that settles the exact details of the following purchase. This contract should be logged with some trusted logging server[10], before actually providing the payment information. This is necessary to prevent the service provider from intercepting and destroying the mobile agent after it has provided the payment information and subsequently offering a service that is not the one it negotiated with the shopping agent[11]. The described shopping agent can be implemented based on the conducts for basic protection, migration control, and external logging, which guarantee that it is protected from any external interference and can enforce the logging with a trusted logging server.

The interaction between the shopping agent and the service provider allows for an efficient negotiation exploiting all the performance advantages of the mobile agent paradigm. However, due to the non-repudiation requirements of the agent owner, the mobile agent has to interact with a TS via a remote interaction. This may cause considerable performance penalties and in particular the TS can become a bottleneck if its resources are consumed by a large number

[10]The agent could send the corresponding information directly to the agent owner, but since it needs an acknowledgement for the receipt of the log entry and since the agent owner might not have a permanent connection to the network, it is preferable to delegate this task to a TTP (which could also add a timestamp to the log entry).

[11]The shopping agent could also rely on the conduct for local logging and ensure that it will sign at most a single contract. This would allow the agent owner to request the contract signed by its mobile agent from the service provider. Since the service provider will only be capable to provide a single contract signed by the mobile agent, he is also prevented from offering a different service than the one negotiated with the mobile agent. Nevertheless, due to the rather high cost of non-volatile storage, we assume that the conduct for local logging will not always be available to all mobile agents. Furthermore, the service provider could simply refuse to provide the contract at all, if this is in its interest.

of clients. Therefore, we propose to encapsulate the functionality of the TTP in a mobile agent, the TA, that can be executed on the TPE.

4.3.2 The Logging Service

In the case of logging, the task of the TA is to act as a proxy for the actual TTP on the TPE. For this discussion we assume that the TA can rely on the following conducts:

- basic protection,
- migration control[12],
- protected invocation,
- synchronized time,
- local logging, and
- external logging.

The TA will accept log entries from mobile agents on the TPE, store them in a local cache, and respond with an acknowledgement in which it guarantees that it will forward the message to the TS[13] unless the TPE is destroyed (see below). Once a log arrives at the TS, it will be handled like a regular log request. In order for the TA to provide such a guarantee, it needs access to a sufficient amount of non-volatile storage on the TPE, in which it can safely store the log messages. Since this non-volatile storage is a limited resource of the TPE, the TA needs a special authorization to use it. Since the TA is a special agent that is assumed to manage the non-volatile storage efficiently and that also provides a service to the TPE owner, we assume that the TPE owner will grant this authorization; otherwise, the TA will abort.

The remaining problem is how the TA can guarantee that logged messages that are stored in the non-volatile storage of the TPE will eventually be forwarded to the TS. The TPE owner could simply intercept all the messages of the TA to the TS or, ultimately, request the TPE to terminate the TA. (This functionality has to be offered by the TPE to protect its owner from malicious or simply buggy agents that refuse to terminate.)

This problem can be solved with a supervision of the TA by the TTP and with the help of the synchronized clock provided by the TPE. The TTP has to keep track of all the TAs it sent to the various service providers and of the expiration date that is associated with each TA. A TA will accept log messages only until its expiration date. After this date it will refuse to accept and acknowledge any further messages. Thus, the TTP has to receive a final message from the TA after its expiration date (there should be an additional delay to accommodate for clock skew) indicating that no further messages for the TTP are stored in the non-volatile storage of the TPE. Provided that the TA only deletes messages from this non-volatile storage after sending them to the TS and obtaining an acknowledgement, the TTP knows that all the messages that the TA acknowledged have been forwarded to the TS. If this final message is not received within a given time, the TTP will request the TPE owner to restart the TA and to forward any messages it sends to the TS. Under the assumption that the TTP has a possibility to enforce the access to the TPE by legal means (we assume that the TTP has a

[12]The TA is a stationary agent that will not request a migration, so this is simply a safety assemption.

[13]Since the TA is configured by the TTP, the use of a symmetric key cryptosystem is sufficient.

special legal relationship with the TPE owner otherwise it would not send its agents to the TPE), the only possibility of the TPE owner to avoid the provision of missing messages is to destroy the TPE. Thus, the approach cannot guarantee that all logged messages will be delivered to the TS. But it can guarantee that a TPE owner can not cheat without being discovered. Furthermore, if it can be proven that the TPE owner intentionally destroyed the TPE, he can be punished with adequate fines.

The problem of destruction of storage media is not new and also applies to a regular TTP. However, it is assumed that the TTP operator implements adequate measures to avoid this problem.

4.3.3 Other Services

The concept of the TA is suitable for any TTP functionality that can be wrapped up in a reasonably small object and that does not have to rely on a large centrally managed database. Other interesting examples are timestamping or fair-exchange.

The former consists of a TA that adds a timestamp to a message and signs the resulting message with its signature key. The latter is a classical security problem, for which several solutions relying on a TTP have been proposed [BP90]. The problem of fair-exchange is that of principals A and B who want to exchange the data items D_A and D_B, but neither of them wants to provide its data item before being sure that it will eventually receive that of the other principal. A TA can facilitate the exchange by accepting the data items as well as a description of the data items expected by the designated receivers. It will verify if the descriptions match the actual data items (e.g., in the case of payment, it verifies if the paid amount corresponds with the amount expected by the receiver) and, if this is the case, deliver the data items to the designated receiver.

4.3.4 Discussion

The relocation of the TTP functionality from a remote host to the locally managed TPE allows us to prevent it from becoming a bottleneck that slows down other components. This is possible since the TPE owner can allocate as many resources as necessary to the TA without having to coordinate this with the TTP. Also, in the case of logging, since the TA can collect and merge several log entries from interactions of different agents with the service provider, it can forward them in a single remote interaction. Another major advantage of the described approach is that the remote interaction with the TS is taken off the critical communication path between a mobile agent and the service provider. They can continue their interaction as soon as the TA has stored the log message and sent the acknowledgement. Moreover, the entire interaction can exploit the locally available communication links with higher bandwidth and lower latency. This enables not only a better overall performance of the system, but allows interactions that were previously not possible due to an unreasonable overhead. For instance, a TA could be used by two interacting parties as an intermediate through which all messages are exchanged. Therefore, the TA can easily log the entire interaction and subsequently forward it as a single log entry to the TTP.

4.4 Other Application Areas

Apart from its usage for the protection of mobile agents in the mobile agent paradigm, that was discussed above, and for the protection of personal privacy, which is the topic of the following Chapter 5, the presented approach based on the TPE and the CryPO protocol, can also be used for other application areas. We want to briefly sketch two of these.

4.4.1 Copyright Protection

If we invert the basic architecture, by allocating the TPE not to the service provider but to the customer, the presented approach can be used to address the problem of protecting the copyright on digital works. This problem is concerned with the fact that digital works (e.g., news items, music, or computer programs) can currently be copied freely for a marginal cost. This situation provides no incentive for copyright holders to make their work available in digital form (apart from computer programs, which can only be distributed in this form).

In [Ste96], Stefik describes an architecture for the distribution, acquisition, and use of copyrighted digital works, which is based on a *trusted system* that is located at a customer and that enforces the copyright. It does this by preventing the creation of any unauthorized copies or by charging the copier a set fee for an authorized copy. In his discussion Stefik identifies the issue of integrity, which has three parts: physical integrity, communications integrity, and behavioural integrity. These could easily be enforced by the TPE, the CryPO protocol, and an appropriate policy for the TPE.

4.4.2 Secure Circuit Evaluation

A well-known problem in cryptographic research is that of secure circuit evaluation, which has been described by Abadi et al. in [AF90] as follows: Bob has an algorithm to compute a function f and is willing to compute $f(x)$ for Alice. Alice wants to compute f on her private input x but does not want to reveal x to Bob. Furthermore, Alice should not learn anything substantial about the algorithm of Bob for computing f[14].

The presented approach provides a pragmatic solution to this problem. Bob can encode the function f in a mobile agent A_f, that he sends to Alice, who operates a highly secure TPE. Alice installs A_f on her TPE and provides it with x. The mobile agent A_f calculates $f(x)$ and gives the result to Alice, but will provide no other information on f to Alice. If Bob wants to limit the usage of A_f to a certain duration or number of calculations, he can use the techniques described in Section 4.2.5 and 4.2.6 respectively. Since Alice can control all the communication to and from her TPE, she can ensure that A_f does not secretly communicate x to Bob. In an extreme case she could even destroy the TPE after the computation to guarantee that Bob will never be able to obtain any information on x.

The transformation in our pragmatic solution that requires Alice to perform the computation instead of Bob, is not necessarily a major limitation, since in [AF90] Alice also has to perform several computations

∞∞∞

[14]The problem of secure circuit evaluation can also be interpreted as a stronger version of the confinement problem identified by Lampson in [Lam73]. The confinement problem also requires Bob not to learn anything about x. However, it is not concerned with the protection of f from Alice.

Summary

In this chapter we explore the usage of the TPE for various goals that may be desirable for an agent owner, such as the assurance to interact with authentic code, a limitation on the number of times a mobile agent can be executed on a particular TPE, or the possibility to interact anonymously with other entities. This is described in the form of conducts, which identify the requirements, the policy, and the necessary collaboration among the different components to achieve the desired goals.

We then illustrate the usage of several conducts with the example of a TTP agent. The TTP agent acts as a proxy for a TTP on a TPE and allows us to take full advantage of the mobile agent paradigm. The chapter concludes with a brief outlook on two other application areas for the TPE and the CryPO protocol.

Chapter 5

A Framework for the Protection of Privacy

O negligence!... what cross devil made me put this main secret in the packet I sent the king? Is there no way to cure this? No new device to beat this from his brains?

– **W. Shakespeare**

5.1 Introduction

We will now return to the discussion of privacy protection in Chapter 1 and use the mechanisms that we have developed for the protection of mobile agents for the protection of privacy. The idea that we want to pursue in this chapter is to make the data active, in order to let it implement some self-defense. This is accomplished by encapsulating the data within a mobile agent, which acts as a vehicle for active data. We will refer to these mobile agents as *data agents*. Such a data agent encapsulates arbitrary data items, which can be very small (e.g., a single integer) or quite large (e.g., a complete medical record). These can then be given to a controller (which is the principal who uses the data) and used by the controller very much like a regular data item. The crucial difference, however, is that the encapsulation allows the data agent to protect its data with appropriate access control mechanisms, to perform additional external actions before an access to data is granted (e.g., log an access with an external TTP), or to simply expire in order to force the controller to replace the old data agent with a fresh data agent. Such a fresh data agent may hold updated data items or a new (and possibly more restrictive) access control policy.

This interpretation of mobile agents as a vehicle for active data allows us to ensure the basic principles of privacy protection that we identified in Section 1.4, which are the principles of:

- notification,
- control, and
- trust.

The goals that we want to achieve with our framework are thus threefold, first to provide data subjects with information on who uses what data related to them for what purposes. Second, we want to give them some limited control over the contents of this data (to correct erroneous data) and over who can access it. Third, the data subjects should have reason to believe that the principles of notification and control are actually observed.

The main idea that we pursue to realize these goals, is to require that all personal data on individuals is exclusively held within data agents. The first two goals can then easily be achieved with a straightforward realization of the naive implementation for notification and control within the data agent, which consists in querying the data subject for every access, change, or disclosure of data (see Sections 1.4.1 and 1.4.2). However, since the data agents can be freely programmed and can thus implement any access control policy and realize any additional behaviour, it becomes possible to considerably improve on the naive implementation for notification and control. For instance, a data agent may only log the first access to a particular data item or request a confirmation from the user only after well defined timeouts. Nevertheless, this still leaves us with the following two problems that need to be addressed to ensure the principle of trust:

DA1: how can the controller be prevented from simply bypassing the protection of the data agent (i.e., from accessing the data directly) and

DA2: how can the controller, once it has obtained some data via a regular access, be prevented from storing it in his own database for later use and thus circumventing any further protection by the data agent.

In order to resolve the former problem DA1, we rely on the approach discussed in Chapters 3 and 4 and require that a data agent is executed on a TPE located at the controller. We further assume that the data agent can rely on the following conducts to ensure its protection:

- basic protection: this allows the data agent to implement access control on its data;
- migration control: since data agents are usually stationary, this is mainly a safety assumption to prevent that a data agent is moved to an insecure TPE;
- synchronized clock: this allows data agents to implement a limited lifetime, which is one of the measures for control that we will discuss in Section 5.3.3; and
- external logging: this conduct allows the data agent to send notifications to the data subject or to some logging server designated by the data subject.

The latter problem DA2 does not have a convincing technical solution due to the basic problem of information disclosure (see Section 1.2.4). Our approach to address this problem is quite ordinary and consists of requiring laws that make any storage or communication of personal data on individuals outside of protected data agents illegal[1]. If at some point personal data on individuals is discovered on the storage media of a principal or communicated between principals, then these principals will be prosecuted and punished according to the appropriate

[1]The actual definition of such laws is in itself a very difficult problem and not within the scope of this dissertation. A particular problem is, for instance, the communication of the data subject's address to the network operator, in order to send a message to the data subject. This needs to be resolved with the creation and maintenance of an explicit catalog that identifies what kind of disclosures are permitted.

laws. Obviously this is very difficult to enforce (short of a complete surveillance of a controller) and does not lend itself to technical support. Nevertheless, we believe that if the presented approach based on protected data agents allows all interacting parties to conduct their regular business without significant complications, then there is little reason not to follow the rules and to risk punishment. We will briefly discuss this issue in Section 5.3.4.

5.2 An Example: Mobile Telecommunications

A simple example for the possibilities of privacy protection with the presented approach is the protection of location information in mobile telecommunication systems, such as GSM [MP92] (Global System for Mobile Communications). In such a system the serviced area is subdivided into *cells*, which correspond to the coverage area of a radio transceiver (in GSM this is called a Base Transceiver Station, *BTS*). In order for a user, say Alice, to remain reachable (i.e., available to receive incoming calls), she has to register the cell in which she currently resides with a well known location register (in GSM this is called Home Location Register, *HLR*). This location information is required by the telecommunication provider, in order to route an incoming call to Alice's current cell (see Figure 5.1). However, for the operation of the system it is not necessary that the telecommunication provider knows the location of Alice at all times, but it is sufficient to know her location when she has to be contacted for an incoming call. Since it is impossible to foresee when this will occur, Alice has to send continuous updates on her current location, which will be stored in the HLR and subsequently overwritten with more recent information whenever she moves to a new cell. This location information is confidential data that should not be available to anyone without a compelling reason (such as law enforcement personnel with an order issued by a judge). Furthermore, it could be abused by the telecommunication provider who could use this information to create a continuous trace of Alice's movements (a movement profile, see Section 1.2.1).

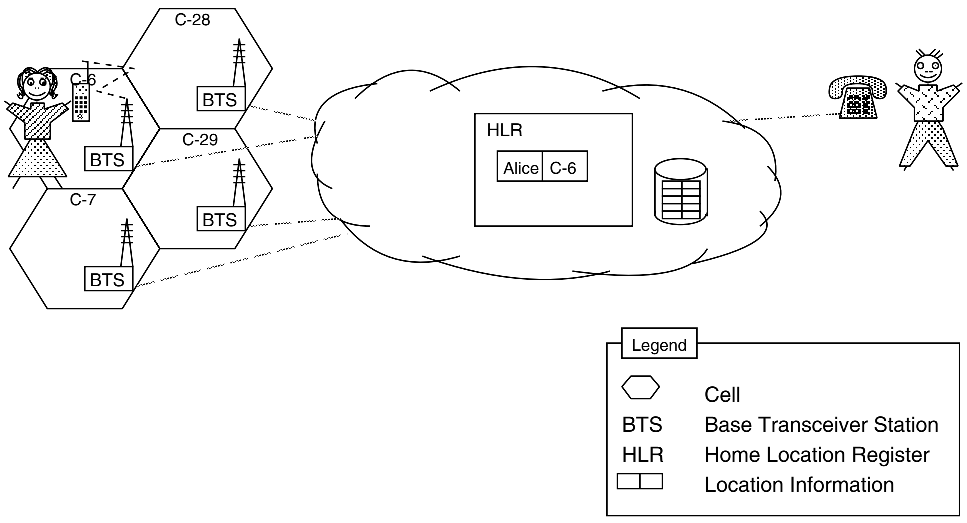

Figure 5.1: Location management in GSM

Currently this data is solely protected by legal means, which prohibit its use for anything other than the intended purpose (i.e., locating a user for an incoming call). Since this is not sufficient in the presence of malicious insiders, who can potentially access all the data without any constraints, we want to add a stronger protection based on technical means that allows only authorized access to this confidential information.

By storing the location information in a data agent that is executed on a TPE in the domain of the telecommunication provider, the location information can be protected more efficiently (see Figure 5.2). Since Alice can configure the data agent according to her personal preferences, she can instruct it on how to handle her location information. Moreover, she can establish a symmetric session key with her data agent that can be used to protect information communicated between her and her data agent. We assume that the location updates will be sent by Alice in encrypted form to her data agent, using this symmetric key[2]. Hence, Alice's location is exclusively known to her data agent, but not to the telecommunication provider. In the case of an incoming call, the telecommunication provider will query her data agent to obtain her current location. Depending on the configuration of the data agent, there are many possibilities how it can react to such a query. The simplest would be to always disclose the current location of the user. This would not really protect the location information, but if the data agent keeps a log of all the queries, the user can verify that his location was only disclosed for genuine communication requests[3]. On the other extreme, the data agent could act as reachability manager [DFRB97] by requesting each caller to authenticate himself and by accepting only communication requests from a set of users defined by the owner of the data agent (other callers can be deviated to a voice mail system).

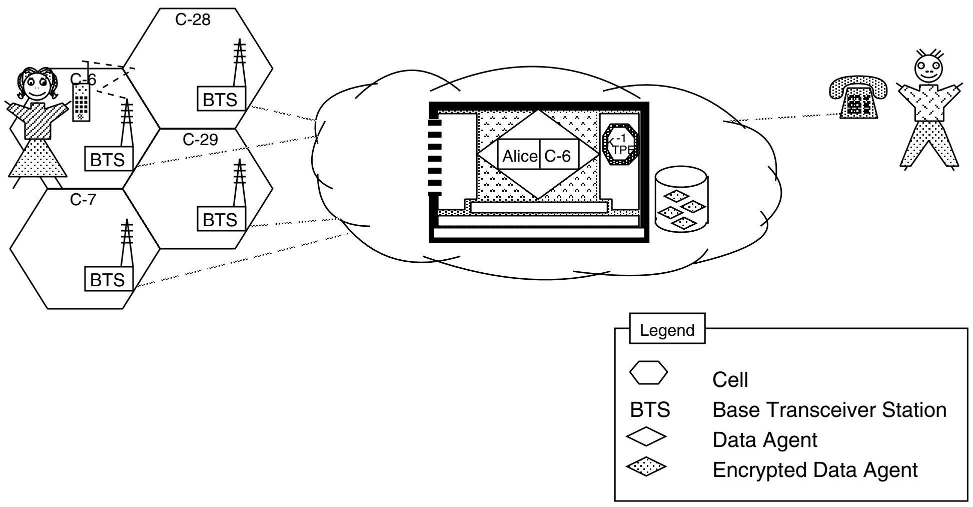

Figure 5.2: Location management with a data agent

Since Alice trusts her data agent to never disclose her current location without proper

[2]Obviously this would require a rather profound change in the operation of GSM, which currently puts the responsibility for location updates on the base transceiver station. In the presented approach this responsibility would be shifted to the mobile user. This is a typical example of a general problem, which is concerned with the tradeoff between moving the "intelligence" into the end devices or keeping it in the network. For security related problems it is often beneficial to move the "intelligence" closer to the user. However, this can often cause a considerable overhead since it prevents certain optimizations.

[3]The data agent can autonomously detect systematic attacks, such as continuous probing that could be conducted to obtain a trace of the user's locations.

authorization, the amount of confidential information that is available to the telecommunication provider is greatly reduced. Instead of having instant and permanent access to Alice's location, it only gains access to her location if she is called by an authorized user. Therefore, even without explicitly addressing the problem DA2, the goal of protecting the access to confidential information is partially achieved. A similar approach to address this problem, which is based on a fixed terminal within the network that is under the user's exclusive control is presented by Pfitzmann in [Pfi93].

5.3 A Simple Framework

We now want to develop the framework that we envision for the protection of personal data on individuals. We will only present a simple version of data agents, which can not be updated by the controller. An extension to updateable data agents is possible, but not immediately relevant for the protection of privacy.

A data agent is a simple mobile agent that implements a limited lifetime (for instance on the order of one month) and manages a list of entries. These entries contain the actual data items as values and can be accessed via a simple interface that allows another principal to query which entries are contained in this data agent and to query for the value of a particular entry. The latter query can be subject to an additional access control decision or to further actions, as explained below. A data agent has a unique identifier (*daID*) that is exclusively known to it and to the data subject[4]. This identifier can be used to create local pseudonyms for the data agent, which consist of the *daID* concatenated with some well-known information, encrypted with the public key of a TPE: $\{daID;\ \text{“}\textit{This is a daID}\text{”}\}_{K_{TPE}}$. This guarantees that any two pseudonyms for the data agent of a data subject at different controllers (with different TPEs) will be unlinkable but nevertheless consistent over many incarnations of the data agent at the same TPE. This pseudonym can also be obtained from the data agent via a method invocation on its interface. Each of the entries of the data agent is a four-tuple <*tag, value, access, log*>, where the fields are used as follows:

- The *tag* field contains the name of this entry and is used to select the corresponding data item.
- The *value* field holds the actual data item that is stored in this entry.
- The *access* field contains the access control information that defines the constraints under which the data item can be accessed. This can, for instance, be a list of public keys towards one of which an accessor needs to authenticate itself in order to get access to the data item.
- The *log* field defines whether and how accesses to the data item should be logged with a logging server trusted by the data subject (for performance reasons this logging server should be implemented by a local TTP agent on the TPE as described in Section 4.3). The log may be created immediately before disclosing the data item and can therefore contain information generated during the access control.

The described approach allows the data agent to implement and enforce the access control policy desired by the data subject and to notify the data subject of the actual use of its data.

[4]This identifier has to be sufficiently long to prevent guessing attacks.

Furthermore, since the TPE on which the data agent is executing supports migration control, the data agent can refuse to migrate to a different TPE, which prevents the controller from legally disclosing personal data on individuals to another principal.

5.3.1 The Use of Data Agents

The way a controller uses data agents is quite classical. We assume that the controller operates a regular database, in which the data agents are stored in encrypted form. Whenever it wants to access a particular value that is stored in a data agent, it will search the corresponding data agent in its database, load it on its TPE (where it is decrypted and launched), and query it for the desired value. In order to manage a potentially large number of data agents, the controller can use the local pseudonym of the data agent as a unique identifier to organize the storage of the data agents. This pseudonym can also be used to reference the data agent in various indices managed by the controller and furthermore it allows the controller to store additional information related to the data subject in regular database entries that are linked to the data agent via the local pseudonym (e.g., billing data). Since this local pseudonym is only meaningful in the context of a particular TPE, it is not considered as personal data. Therefore, it would be possible to communicate the local pseudonym to a third party, but since it is prohibited to communicate identifying personal data, this third party would not be able to link a particular data subject's local pseudonyms from different controllers.

Upon its instantiation on the TPE, the data agent will verify that its lifetime has not expired (see Section 4.2.5) and then wait for accesses to its data. When the controller provides the tag of a certain entry that it would like to access, the data agent will implement the access control policy that is specified in the *access* field of this entry and, if necessary, notify some logging server of the access. If both the authorization and the notification are successful, the data agent will provide the data item that is stored in the *value* field.

On the other hand, if the lifetime of the data agent has expired, then the data agent will grant no further access to the data it contains. This forces the controller to obtain a fresh data agent from a service provider that should be accessible via the network, to which we will refer to as the operator of the *repository*. The only information that the expired data agent will still provide after its expiration is what is needed by the controller to obtain the fresh data agent. This information must contain the address of the repository and some identification for the requested data agent. It would be possible that the data subject operates the repository for his data agents himself, which would give him direct control over the dissemination and the contents of his data agents. However, this has two serious disadvantages:

- the address of the data subject could be sufficient to identify him, which contradicts the requirement that the data agent provides no further useful information after its expiration and
- the server that is connected to the data subject's address is a private computer, which may be unreliable or too slow for the task (it is also likely that the data subject does not have sufficient expertise to administrate and operate such a server).

To overcome these problems, we decouple the data subject from its data agent and assume that the repository is operated by an independent third party. Since this principal provides its service with the help of a highly secure TPE, it does not necessarily need to be trusted with respect to the possibly confidential data it manipulates – nevertheless, it needs to be trusted for other reasons, for instance, to provide its service with a high reliability. The required protection can be ensured by specialized mobile agents, *data agent factories*, that execute on the TPE of the repository.

5.3.2 The Distribution of Data Agents

One of the goals of the more centralized distribution of data agents by the repository's operator is to create a sufficiently large anonymity group that prevents a particular individual from being identified via the address of his repository. However, it also creates the risk of a single point of failure for all the users of this repository, which needs to be addressed by more stringent security requirements towards the TPE of the repository.

We now want to briefly discuss the mechanism with which a controller can obtain a fresh data agent from a data agent factory executing on the repository's TPE. Figure 5.3 gives an overview of the interactions. When the data agent of a controller expires, it will still provide the address of the repository on which the data agent factory of its owner is located as well as a reference to the data agent factory that can only be used by the repository. It is important to ensure that this reference can not be used as a global identifier for the data subject. This can, for instance, be achieved by concatenating the secret identifier of the data agent factory (which may be the same as the identifier of the data agent, daID) together with a well-known information and a random number. The resulting bitstring is then encrypted with the public key of the repository's TPE (K_R): $\{$*daID*; *"This is a request for a data agent"*; $rnd\}_{K_R}$. Due to the random number, the reference created by any data agent will always be a completely different bitstring, which does not help the controllers to match data agents of the same data subject.

The controller will send a signed request for a fresh data agent together with this reference to the operator of the repository. The firmware on the repository's TPE will decrypt this reference and create the local pseudonym of the data agent factory[5] $\{$*daID*; *"This is a data agent factory"*$\}_{K_R}$, and provides this to the operator of the repository. The operator will then locate the encrypted data agent factory, load it on its TPE, and provide it with the request from the controller.

The data agent factory is a mobile agent that belongs to a data subject and is capable to create data agents for this principal. However, it does not implement the functionality of the data agent itself, which prevents the operator of the repository from accessing any of its data. The data agent factory is assumed to be in close contact with the data subject (they can share a symmetric encryption key for efficient interactions) and is configured by its owner with the entries that it should include in the data agents it creates. Moreover, it contains a list of the public keys of all the controllers who are entitled to receive a fresh data agent (this could also comprise an information on the entries a particular controller should obtain in a fresh data agent). When it receives the request for a fresh data agent from a particular controller it checks if this controller is on its list and verifies the signature on the request. If both checks are successful, the data agent factory will create a fresh data agent with a certain lifetime and send it to the controller.

The act of providing a data agent to a controller consists thus of notifying the data agent factory that this controller is entitled to a current copy of the data agent (this notification may also contain an information on the entries this particular controller should obtain) and

[5]This local pseudonym could be used as a global identifier for the data subject. To prevent this, the operator of the repository must not disclose this local pseudonym to any other principal.

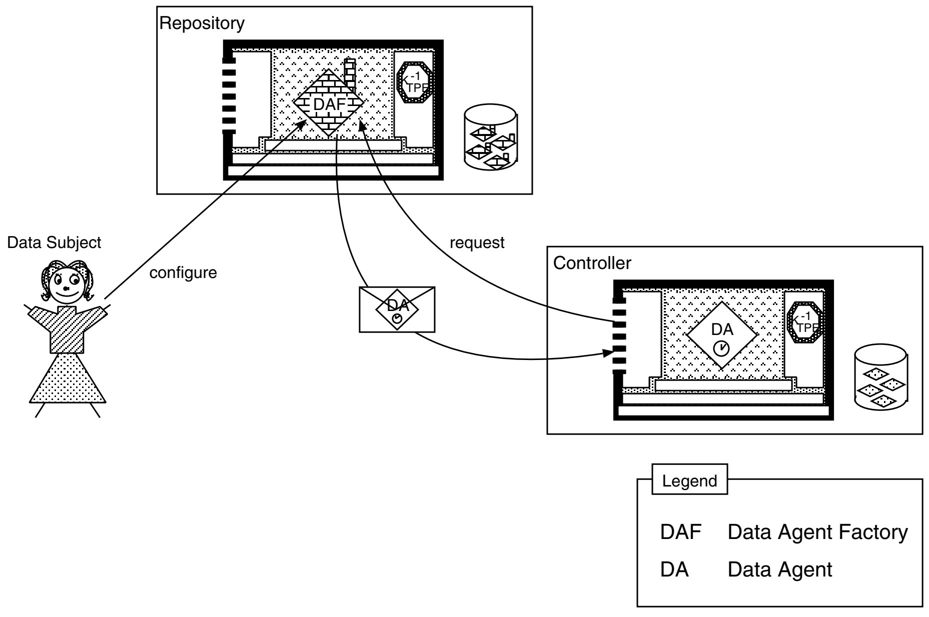

Figure 5.3: The distribution of data agents

of providing the controller either immediately with a valid data agent or with the necessary information with which it can obtain a fresh data agent from the repository.

A positive side effect for controllers is that the data in the data agents will almost always be accurate. For instance, when a user moves to a new address, then he only has to update his data agent factory in the repository and eventually all the controllers with whom he interacts, will receive a fresh data agent that reflects these changes.

5.3.3 Withdrawal of Data

Based on the use and distribution of data agents described above, it now becomes possible to “withdraw” data from a controller. This does not constitute a general solution to the basic problem of information disclosure, but within the presented framework it achieves a similar result.

Since we assume that information on individuals can exclusively be held in data agents, a controller will not keep a copy of the data stored in the data agent in a separate place (this would be illegal). Thus, if the data subject removes a controller from the list of those entitled to obtain a fresh data agent as defined in the data agent factory, the controller will not be able to obtain a fresh version of the data agent after its current copy has expired. The data has effectively become inaccessible to the controller who will therefore no longer be capable to interact with the data subject. What is left to the controller is some anonymous data that can not be linked with the data of any other controller and an encrypted data agent that refuses to provide any usable data. We assume that the controller, due to the large number of data agents with which it interacts, will not be capable to otherwise reconstruct the data that is stored in the data agent.

Nevertheless, there may be situations when the controller has a legitimate right to continue its interaction with the data subject. For instance, if the data subject has open bills with the controller. For such a case there must be mechanisms that permit a legal body (e.g., a judge after review of the evidence) to force the data agent to provide its data or the data agent factory to provide a fresh data agent. This can be accomplished with the installation of legal backdoors as discussed in Section 4.2.7. The verification that a data agent actually implements the legal backdoor can either be accomplished with the help of the TPE or implemented based on the conduct for authentic code (see Section 4.2.4).

The presented approach may obviously be extended by using more elaborate methods in the construction of the data agent by the data agent factory, which may allow the data subject to selectively withdraw access to particular entries in the data agent.

5.3.4 Enforcing the Use of Data Agents

As we have already noted in Section 5.1, the enforcement of the presented approach is extremely difficult. This is due to the fact that a controller can simply store all the data it obtains from a data agent in a private database, possibly in encrypted form to avoid punishment if this illegal data storage is discovered. The problem with the enforcement of the presented approach is similar to the enforcement of laws in other domains, such as the requirement for a proper technical condition of cars or the prohibition of insider trading on stock exchanges. However, despite the inherent problems of enforcing these laws, they are still largely adhered to, e.g., in Switzerland with respect to cars.

The reasoning behind the enforcement of such laws must be to make them easy to follow and to severely punish any violation. Therefore, our approach relies on the following three points:

- The laws require that the infrastructure to work with data agents is put in place (therefore, the money to build the infrastructure has to be spent in any case).
- The infrastructure allows all necessary operations on data (therefore, there is no compelling reason not to use data agents and the corresponding infrastructure).
- Non-compliance with laws will be punished (we assume that most people and companies simply prefer to stay within the laws).

The creation of a comprehensive database in a country that does not enforce a law that requires the use of data agents can not be prevented. Yet, since this data can not easily be sold to countries that enforce such a law, there is less incentive to create such a database in the first place. Consider, for instance, the effect of the EU-Directive on Data Protection [EC95] on the IRSG (Individual References Services Group) that was mentioned in Section 1.3.2. Even though there is no compelling law in the US that requires self-regulation, the desire to conduct business with individuals and companies of the European Community caused the members of the IRSG to adapt their business practices.

5.4 Making it Manageable

The major problem with the presented approach is complexity. The problem is that if individuals are provided with too much choice they may easily be confused and overwhelmed. The system should not require them to make choices about technical matters that are more suited to expert analysis. Furthermore, a regular user can probably not cope with the amount of information that is created in the system and that has to be managed. Therefore, there must be standard procedures and standard principals who organize the flow of data.

5.4.1 The Data Agent – Accessor Agent Paradigm

To overcome this complexity we propose to encapsulate most of it in a specialized mobile agent, to which we will refer to as the *accessor agent*. Consequently we refer to the entire approach as the data agent – accessor agent (da^3) paradigm. For the realization of this paradigm, we have to rely on the protected invocations conduct (see Section 4.2.3) that has to be supported by a TPE.

The basic idea of the da^3 paradigm as depicted in Figure 5.4 is to interpose the accessor agent between the data agent and the controller, where it can mediate accesses by the controller. All entries of the data agent are thus protected by an additional access control, which grants access only to accessor agents issued by a trusted principal (i.e., the accessor agent must hold a private key to authenticate itself to the data agent).

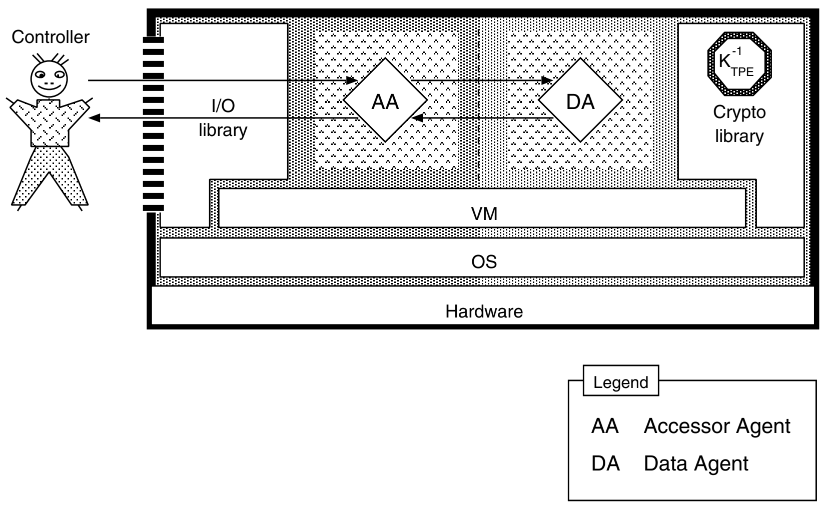

Figure 5.4: The data agent – accessor agent paradigm

We assume that the accessor agent is a highly specialized agent from an expert principal that acts as some form of lawyer for the data subject, which can make better decisions about the right of a controller to access a particular data item. However, the ultimate decision remains with the data subject who can decide whether he wants to provide his data agent and grant access to a particular accessor agent or not.

This means that most choices about technical matters should be dealt with by the implementor of the accessor agent, while the data subject only has to define some general guidelines on how he would like his personal data to be handled. The issuer of an accessor agent can, for instance, handle this with the definition of several categories of users who have different privacy preferences. Each accessor agent it issues will then request the category that was chosen by the owner of a data agent (this can be an entry in the data agent) and then implement a policy that respects the privacy preferences for this category of users.

If at some point the user remarks that there is a problem with the way data on him is used, then he can consult with an expert, who will suggest remedies to the data subject. Since all the data items and business relations of a data subject are encoded in his data agent, it is possible for the expert to analyze the problems based on this data. The remedies can then be put in place and will start to work after a reasonable period of time, say on the order of one month (when the data agents at the controllers expire and have to be refreshed with new data agents). This means that the data subject is not left alone and can conduct his daily business without having to trust in the good behaviour of a controller who has a direct interest in the data. Rather he can rely on an independent expert who has no direct interest in this data, who is more apt to decide if some data is actually needed by a controller, and who will contain the complexity of data interactions[6].

5.4.2 Using the da^3 Paradigm

Consider a person who applies for renting an apartment. Depending on the market situation it is often the landlord who can select among several applicants and who wants to chose an applicant who is capable to pay the rent for the apartment. Therefore, he will often request very detailed information on the financial situation of an applicant, even though he might legally only be entitled to a more limited information (e.g., whether the applicants monthly income is higher than three times the monthly rent). The applicant in turn knows that he will not be considered by the landlord if he does not provide comprehensive information on his financial situation and may often not know what information the landlord is entitled to obtain.

This dilemma can easily be resolved with the da^3 paradigm. The landlord needs to obtain an accessor agent from a trusted principal that will ensure that the landlord will only obtain the information he is legally entitled to. The applicant can then simply provide its entire data agent, which contains an entry that holds his monthly income, signed by the applicant's bank. This entry is protected by an additional access control, which grants access only to the agents of principals that are trusted by the data subject. Provided that the data subject trusts the issuer of the accessor agent used by the landlord, the data agent will provide it with the information on the data subject's monthly income. Subsequently the accessor agent will then use this information to compute exactly the information that the landlord is legally entitled to obtain.

∞∞∞

[6]On the other hand, this also makes users less responsible by moving responsibility to another entity, but this is simply the way our society works (e.g., government).

Summary

In this chapter we discuss a framework for the protection of personal privacy that relies on the use of mobile agents. These mobile agents are used as carriers of data that implement the principles of notification and control; they are referred to as data agents.

In order to protect the data agents we require that they are executed on a TPE and in order to ensure the protection of the data they contain, we prohibit any storage of personal data outside of the protected data agents. We then describe a simple example in the context of mobile telecommunication systems that allows a mobile user to protect his location information from the telecommunication provider. The chapter then continues with an overview on the use of data agents for the control of access to personal data, explains how this can be used to "withdraw" information from a principal and comments on the enforceability of the approach.

Finally, we sketch an extension to the basic framework that contains the complexity of access decisions in specialized agents, the accessor agents. These mediate every access to the data between the controller and the data agent and, hence, act as some form of lawyer for the data subject.

Conclusion

It's dangerous to be right when the government is wrong.

– **Voltaire**

The problem of protecting the personal privacy of individuals is a difficult issue that needs to be addressed in a convincing way in order for the information society to become a *desirable* reality. In this thesis we have analyzed some of the problems related to privacy in the information society and have developed a technical approach to privacy protection that is based on the principles of notification, control, and trust. The principle of trust is identified as a central issue and leads us to the distinction between an optimistic approach to trust and a pessimistic approach to trust. The former tries to control the actions of principals and to subsequently sanction malicious actions that are detected, while the latter tries to prevent malicious actions in the first place.

The technical approach to privacy protection is based on the mobile agent paradigm and on a trusted and tamper-resistant hardware device. It requires us to rely on an optimistic approach to trust with respect to the manufacturer of the trusted and tamper-resistant hardware device but enables us to rely on a pessimistic approach to trust with respect to the executor of the mobile agent.

The remaining results of this thesis are twofold. First we have developed an approach to protect a mobile agent from its executor that is based on the trusted and tamper-resistant hardware device which is referred to as TPE. This approach is then explored in great detail with the help of conducts, which describe how and under what assumptions an agent owner's protection goals can be realized. Second we have sketched a hypothetical framework to address the problem of privacy protection with the help of mobile agents. Furthermore we have briefly presented the data agent - accessor agent (da^3) paradigm, which allows us to make privacy protection more manageable.

Assessment of the Protection of Mobile Agents

We believe that our work on mobile agents and their protection within TPEs is interesting in its own right. Tamper-resistant environments become ever more common place, so that some of the conducts that we have described may soon be implemented in real systems.

On the other hand, it seems not clear that highly mobile agents, which visit many agent platforms in order to collect information or conduct business with the operators of these platforms, will be the actual use of mobile agents in the context of electronic commerce. (This might be confined to special application domains such as network management.) It is possible that mobile agents may be used in a much more restricted way. This would consist of mobile users who configure a mobile agent for a particular task and send it to the agent platform of a trusted executor. This trusted executor implements the execution of mobile agents as a service and offers the mobile agents a high speed, high bandwidth access to the network, advanced failure handling, as well as a protection similar to the basic protection of the TPE.

This is a much simpler model that can realize many of the discussed advantages of the mobile agent paradigm, while at the same time being more manageable. The issue of service customization may be addressed with sub-agents, which are created by the mobile agent and sent to unprotected agent platforms at some service provider to accomplish non-critical tasks. This model does not invalidate the results of the presented work (which is geared towards the protection of privacy), but the trusted executor can be interpreted as some form of TPE, which also has to implement a policy. This may be used to realize some of the goals described in the conducts. The main difference is that this model relies on rather different trust assumptions.

Assessment of the Framework for Privacy Protection

We believe that the problems of privacy will become ever more important and will require solutions that give the data subjects more information and control over the use of their data. This may not immediately be achieved in universally deployed system such as the one described. However, it may be applied to a particular application domain where the data that has to be protected can be well identified and isolated within a special subsystem. Within this subsystem it may then be possible to realize the presented approach to privacy protection.

An example for such an application domain is healthcare, in which large amounts of privacy relevant data have to be handled in a complex distributed system. The data that needs to be protected are patient records, which could be implemented as data agents that can only be accessed via accessor agents from some independent principal. This principal mediates between the different interests of data subjects, doctors, nurses, hospitals, insurance companies, and employers. In this setting it would be illegal to store identifiable (i.e., non-anonymized) health-related data outside of protected data agents.

Once such a pilot application is deployed, it might lead to the development of other domains. This may additionally result in a fragmentation of not interoperable systems, which can be an advantage for privacy protection since it prevents data from easily travelling into another application domain where it is not suitable.

Personal Conclusion

During my Diploma at the University of Kaiserslautern [Wil92], I designed and implemented the user interface for the MOSKITO operating system kernel [NG90]. The most fascinating problems that I encountered in the course of this work were those related to security, which led me to the decision to further pursue this topic.

The initial starting point of this thesis was a single unfocused idea: “I want to do something on privacy protection”. This idea has shown itself as quite difficult to address in a meaningful way and has led me to explore many different issues, such as:

- *fault-tolerance in distributed systems*, which was the major research area of the LSE when I started there. Before the funds for working on my primary field of interest were available, I got the chance to explore some problems of fault-tolerance in the context of Phoenix [MFSW95, WS95].
- *security in distributed systems* provides the basis for most of the presented ideas. Since privacy protection is concerned with problems in the real world, where actions are naturally distributed, this has to be modelled in the computer system. A useful tool for the design of secure protocols in distributed systems are logics [BSW98].
- *CORBA* is a very interesting approach for the design of distributed applications and still lacks convincing implementations for security. The research we conducted in this area was particularly influential for the development of this thesis [WSWZ97, SBC+99].
- *TINA* is a quite recent effort that tries to merge the fields of telecommunications and computer science. Since privacy protection is concerned with individuals, it is necessary to consider a system that deals with people, which is obviously the case in telecommunications systems. The CrySTINA project, which evolved concurrently with the work presented in this thesis, offered another setting for discussions and provided valuable input [SBH+97a, SW97a, SW97b, SW97c, SBW98, BSW99].
- *GSM* is the dominant standard for mobile communication systems in Europe. Due to the initial difficulty to obtain documentation on TINA, it was easier to investigate some ideas in the context of the well documented GSM architecture [Wil97].
- *UMTS* is a research effort of ETSI to define the third generation mobile communications system. I investigated into this topic as it could be the immediate application area for the ideas developed in the context of GSM. Due to our focussing on TINA this direction was soon abandoned.
- *Mixes* are a prominent and well developed technology for the protection of privacy in communication systems. It was interesting to examine this approach in order to investigate whether the ideas could be transferred to other application domains.
- *electronic cash* is also an application domain that already has many approaches for privacy protection and also served as an inspiration for the presented approach.

This exploration has resulted in several interesting results and some dead ends. The discovery of the mobile agent paradigm as a possible and viable solution to some of my problems was made in late 1996 and soon resulted in a first publication on this topic [WD97].

This thesis is basically an attempt to linearize many of the results in a comprehensible way and to explore them in an adequate depth. I hope it can serve as a small component in the design of socially acceptable information systems.

∞∞∞

Bibliography

[ABKL91] M. Abadi, M. Burrows, C. Kaufman, and B. Lampson. Authentication and delegation with smart cards. In T. Ito and A. R. Meyer, editors, *Theoretical Aspects of Computer Software*, volume 526 of *Lecture Notes in Computer Science*, pages 326–345. Springer-Verlag, September 1991.

[Ado85] Adobe Systems Inc. *PostScript Language Reference Manual.* Addison-Wesley, 1985.

[AF90] M. Abadi and J. Feigenbaum. Secure circuit evaluation. *Journal of Cryptology*, 2(1):1–12, 1990.

[AK96] 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.

[AK97] R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In *Security Protocols, 5th International Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 125–136, Paris, France, apr 1997. Springer-Verlag.

[And94] R. J. Anderson. Making smartcard systems robust. In *IFIP First Smartcard Research and Advanced Applications Conference*, pages 1–14, Lille, France, oct 1994.

[And96] R. J. Anderson. A security policy model for clinical information systems. In *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 30–43, Oakland, CA, May 1996.

[AR97] P. E. Agre and M. Rotenberg, editors. *Technology and Privacy: The New Landscape.* MIT Press, Cambridge, Massachusetts, 1997.

[Ast98] Ad Astra. Jumping beans. White paper, Ad Astra Engineering Inc., December 1998.

[ASW96] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. Research Report RZ 2858, IBM Research, September 1996.

[BBC$^+$94] J.-P. Boly, A. Bosselaers, R. Cramer, R. Michelsen, S. Mjolsnes, F. Muller, T. Pedersen, B. Pfitzmann, P. de Rooij, B. Schoenmakers, M. Schunter, L. Vallee, and M. Waidner. The ESPRIT project CAFE – high security digital payment systems. In *European Symposium on Research in Computer Security*, Lecture Notes in Computer Science. Springer-Verlag, 1994.

[BBI93] W. J. Barr, T. Boyd, and Y. Inoue. The TINA initiative. *IEEE Communications Magazine*, 31(3):70–76, March 1993.

[Bel97] V. Bellotti. Design for privacy in multimedia computing and communications environments. In Agre and Rotenberg [AR97], pages 63–97.

[BFL96] 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.

[BGP97] M. Baldi, S. Gai, and G. P. Picco. Exploiting code mobility in decentralized and flexible network management. In Rothermel and Popescu-Zeletin [RPZ97].

[Blu] Bluetooth Consortium. A global standard for wireless connectivity. `http://www.bluetooth.com/default.asp`.

[Bot] BotSpot. The spot for all bots & intelligent agents. `http://www.botspot.-com/main.html`.

[BP90] H. Bürk and A. Pfitzmann. Value exchange systems enabling security and unobservability. *Computers & Security*, 9(8):715–721, 1990.

[Bra88] G. Brassard. *Modern Cryptology – A Tutorial*, volume 325 of *Lecture Notes in Computer Science*. Springer Verlag, 1988.

[Bra94] S. Brands. Untraceable off-line cash in wallets with observers. In *Proceedings of Crypto 93*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318, Berlin, 1994. Springer-Verlag.

[Bro77] G. D. Brown. *System/360 job control language*. Wiley, New York, 1977.

[BSW98] L. Buttyàn, S. Staamann, and U. G. Wilhelm. A simple logic for authentication protocol design. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop*, Rockport, MA, June 1998.

[BSW99] L. Buttyàn, S. Staamann, and U. G. Wilhelm. Multilateral security in middleware based telecommunication architectures. In G. Müller and A. Pfitzmann, editors, *Multilateral Security for Global Communication*. Addison-Wesley, 1999.

[Car95] L. Cardelli. Obliq – a language with distributed scope. *Computing Systems*, 8(1):27–59, 1995. Also available as Digital Systems Research Center Research Report 122.

[CEC91] CEC. Information technology security evaluation criteria (ITSEC) – provisional harmonised criteria – version 1.2. Technical report, Commission of the European Communities, June 1991.

[CFN88] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Proceedings of Crypto*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327, Berlin, 1988. Springer-Verlag.

[CGH+95] D. M. Chess, B. Grosof, C. G. Harrison, D. Levine, C. Parris, and G. Tsudik. Itinerant agents for mobile computing. *IEEE Personal Communications*, 2(5):34–49, October 1995.

[CGPV97] G. Cugola, C. Ghezzi, G. P. Picco, and G. Vigna. Analyzing mobile code languages. In Vitek and Tschudin [VT97], pages 93–111.

[Cha81] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[Cha83] D. Chaum. Blind signatures for untraceble payments. In *Proceedings of Crypto 82*, pages 199–203, New York, 1983. Plenum Press.

[Cha85] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[Cha92] D. Chaum. Achieving electronic privacy. *Scientific American*, pages 96–101, August 1992.

[Cha96] D. Chappell. *Understanding ActiveX and OLE: a guide for developers and managers.* Microsoft Press, 1996.

[Cit] Citrix Inc. `http://www.digitivity.com/`.

[Cla89] R. A. Clarke. Information technology and dataveillance. *Communications of the ACM*, 31(5):498–512, May 1989.

[Com98] Common Criteria Editorial Board. *Common Criteria for Information Technology Security Evaluation, Version 2.0*, May 1998.

[Cot95] L. Cottrell. Mixmaster & remailer attacks. `http://www.obscura.-com/~loki/remailer/remailer-essay.html`, 1995.

[CPV97] 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.

[CTL98] C. Collberg, C. Thomborson, and D. Low. Manufacturing cheap, resilient, and stealthy opaque constructs. In *Principles of Programming Languages 1998, POPL'98*, San Diego, CA, January 1998. `http://www.cs.auckland.ac.nz/-∼collberg/Research/Publications/CollbergThomborsonLow97c/-index.html`.

[CZ95] D. B. Chapman and E. D. Zwicky. *Building Internet Firewalls.* O'Reilly & Associates, Inc., Cambridge, 1995.

[Dej] Deja news. `http://www.dejanews.com`.

[DFRB97] H. Damker, H. Federrath, M. Reichenbach, and A. Bertsch. Persönliches Erreichbarkeitsmanagement. In Müller and Pfitzmann [MP97], pages 207–217.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6), November 1976.

[DNI95] F. Dupuy, G. Nilsson, and Y. Inoue. The TINA Consortium: Toward networking telecommunications information services. *IEEE Communications Magazine*, November 1995.

[DO91] F. Douglis and J. Ousterhout. Transparent process migration: Design alternatives and the sprite implementation. *Software Practice and Experience*, 21(8):757–785, August 1991.

[DoD85] DoD. Trusted Computer System Evaluation Criteria (TCSEC). Technical Report DoD 5200.28-STD, Department of Defense, December 1985.

[DS98] P. T. Devanbu and S. G. Stubblebine. Stack and queue integrity on hostile platforms. In *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 198–207, Oakland, CA, May 1998.

[Dys97] E. Dyson. Labeling practices for privacy protection. In W. M. Daley and L. Irving, editors, *Privacy and Self-Regulation in the Information Age*. U.S. Department of Commerce, Washington, D.C., June 1997. `http://www.ntia.doc.gov/reports/privacy/privacy_rpt.htm`.

[EC95] EC. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. `http://www2.echo.lu/legal/en/dataprot/directiv/-directiv.html`, 1995.

[EFL$^+$98] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. Simple public key certificate. Technical Report <draft-ietf-spki-cert-structure-05.txt>, IETF, March 1998.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, June 1985.

[FGS96a] W. M. Farmer, J. D. Guttman, and V. Swarup. Security for mobile agents: Authentication and state appraisal. In *European Symposium on Research in Computer Security*, volume 1146 of *Lecture Notes in Computer Science*, pages 118–130. Springer-Verlag, Rome, Italy, September 1996.

[FGS96b] W. M. Farmer, J. D. Guttman, and V. Swarup. Security for mobile agents: Issues and requirements. In *National Information Systems Security Conference*, pages 591–597, Baltimore, MD, USA, October 1996. National Institute of Standards and Technology.

[FH98] S. Fischer-Hübner. Privacy and security at risk in the global information society. *Information, Communication & Society*, 1(4):419–441, Winter 1998.

[FIP97] FIPA. *FIPA 97 Specification*. Foundation for Intelligent Physical Agents, 1997. `http://www.fipa.org/`.

[FJK$^+$97] H. Federrath, A. Jerichow, D. Kesdogan, A. Pfitzmann, and O. Spaniol. Mobilkommunikation ohne bewegungsprofile. In Müller and Pfitzmann [MP97], pages 169–180.

[FKK96] A. O. Freier, P. Karlton, and P. C. Kocher. The SSL protocol version 3.0. Technical Report <draft-freier-ssl-version3-02.txt>, IETF, November 1996.

[Fla97] D. Flanagan. *Java in a Nutshell: A Desktop Quick Reference for Java Programmers*. O'Reilly, 1997.

[Flo97] Florida's sexual predator, sexual offender, and general felon registration and notification procedures as of October 1, 1997.

[FM96] J. S. Fritzinger and M. Mueller. Java security. White paper, Sun Microsystems, Inc., 1996.

[FM99] S. Fünfrocken and F. Mattern. Mobile agents as an architectural concept for internet-based distributed applications – the WASP project approach. In *Kommunikation in Verteilten Systemen (KiVS)*. Springer-Verlag, 1999.

[For94] W. Ford. *Computer Communications Security*. Prentice Hall, Englewood Cliffs, NJ, 1994.

[Fün98] S. Fünfrocken. Transparent migration of java-based mobile agents (capturing and reestablishing the state of java programs). In *Second International Workshop on Mobile Agents (MA'98)*, volume 1477 of *Lecture Notes in Computer Science*, pages 26–37, Stuttgart, Germany, September 1998. Springer Verlag.

[Gel97] R. Gellman. Does privacy law work. In Agre and Rotenberg [AR97], pages 193–218.

[GM96] J. Gosling and H. McGilton. The java language environment. White paper, Sun Microsystems, Inc., 1996.

[Goo] Google. `http://www.google.com`.

[Gra95] R. S. Gray. Agent Tcl: A transportable agent system. In *Proceedings of the CIKM Workshop on Intelligent Information Agents*, Baltimore, MD, December 1995.

[Gre96] G. Greenleaf. The 1995 EU-directive on data protection – an overview. *The International Privacy Bulletin published by Privacy International*, 3(2), 1996.

[GVU97] GVU. Graphics, Visualization, & Usability Center, GVU's 8th WWW user survey. `http://www.gvu.gatech.edu/user_surveys/`, October 1997.

[Ham98] K. H. Hammonds. Online insecurity (Harris poll). *Business Week*, March 1998.

[HCK97] C. G. Harrison, D. M. Chess, and A. Kershenbaum. Mobile agents: Are they a good idea? In Vitek and Tschudin [VT97], pages 25–47.

[Hoh98] F. Hohl. Time limited blackbox security: Protecting mobile agents from malicious hosts. In Vigna [Vig98b], pages 92–113.

[HP85] A. Herzberg and S. S. Pinter. Public protection of software. In *Advances in Cryptology: CRYPTO'85*, pages 158–179, Santa Barbara, California, August 1985.

[IBM97] IBM. *IBM 4758 PCI Cryptographic Coprocessor*. Secure Way Cryptographic Products, June 1997.

[IDM91] J. Ioannidis, D. Duchamp, and G. Q. Maguire. IP-based protocols for mobile internetworking. In *Proceedings SIGCOMM'91*, pages 235–245, Zuerich, Switzerland, 1991. ACM Press.

[IEE96] IEEE/ANSI Std 1003.1. *Information Technology – Portable Operating System Interface (POSIX) – Part 1: System Application Program Interface [C Language], ISO/IEC 9945-1:1990*, 1996. also Federal Information Processing Standards Publication 151-2.

[IRS98] IRSG. Individual reference services. Technical report, Individual Reference Services Group, 1998. `http://www.irsg.org/`.

[ITU93] ITU. *ITU-T Recommendation X.509: The Directory – Authentication Framework*. International Telecommunication Union, 1993.

[JLHB88] E. Jul, H. Levy, N. Hutchinson, and A. Black. Fine-grained mobility in the emerald system. *ACM Transactions on Computer Systems*, 6(1):109–133, February 1988.

[JMS$^+$98] D. Johansen, K. Marzullo, F. B. Schneider, K. Jacobsen, and D. Zagorodnov. NAP: Practical fault-tolerance for itinerant computations. Technical Report TR98-1716, Department of Computer Science, Cornell University, November 1998.

[KA98] M. G. Kuhn and R. J. Anderson. Soft tempest: Hidden data transmission using electromagnetic emanations. In D. Aucsmith, editor, *Information Hiding 1998*, volume 1525 of *Lecture Notes in Computer Science*, pages 124–142, Berlin, 1998. Springer-Verlag.

[KAG97] G. Karjoth, N. Asokan, and C. Gülcü. Protecting the computation results of free-roaming agents. In Rothermel and Popescu-Zeletin [RPZ97], pages 1–14.

[Kah67] D. Kahn. *The Codebreakers*. Macmillan Publishing Company, New York, 1967.

[Kah97] B. Kahle. Preserving the internet. *Scientific American*, March 1997.

[KJJ98] P. Kocher, J. Jaffe, and B. Jun. Cryptography research q&a on differential power analysis. `http://www.cryptography.com/dpa/qa/index.html`, 1998.

[KLO97] G. Karjoth, D. B. Lange, and M. Oshima. A security model for aglets. *IEEE Internet Computing*, 1(4):68–77, 1997.

[Kuh97] M. Kuhn. The trustno1 cryptoprocessor concept. Technical Report CS555, Purdue University, April 1997.

[Lam73] B. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, October 1973.

[Law98] G. Lawton. The internet's challenge to privacy. *IEEE Computer*, 31(6):16–18, June 1998.

[LC96] D. B. Lange and D. T. Chang. Ibm aglets workbench programming mobile agents in java. White paper, IBM Corporation, September 1996. `http://www.trl.ibm.co.jp/aglets/`.

[Lev95] N. G. Leveson. *Safeware: system safety and computers.* Addison-Wesley, Reading, 1995.

[LMKQ89] S. J. Leffler, M. K. McKusik, M. J. Karels, and J. S. Quarterman. *The Design and Implementation of the 4.3BSD UNIX Operating System.* Addison-Wesley, Reading, 1989.

[Lor93] R. W. Lorenz. Vergleich der digitalen mobilfunksysteme in europa (GSM) und japan (JDC) unter besonderer berücksichtigung der wirtschaftlichkeitsaspekte. *Der Fernmeldeingenieur Verlag für Wissenschaft und Leben G. Heidecker*, 17, 1-2:1–72, 1993.

[LS94] R. Lunheim and G. Sindre. Privacy and computing: a cultural perspective. In Sizer et al. [SYKFH94], pages 25–40.

[Lut96] M. Lutz. *Programming Python.* O'Reilly, 1996.

[Lyn90] E. Lynch. *Understanding SQL.* Macmillan, 1990.

[Mag] General Magic. Odyssey information. `http://www.genmagic.com/-technology/odyssey.html`.

[Mar77] S. T. Margulis. Conceptions of privacy: Current status and next steps. *Journal of Social Issues*, 33(3):5–21, 1977.

[McN] J. McNamara. The complete, unofficial tempest information page. `http://www.eskimo.com/~joelm/tempest.html`.

[MFSW95] C. P. Malloth, P. Felber, A. Schiper, and U. G. Wilhelm. Phoenix: A toolkit for building fault-tolerant distributed applications in large scale. In *Workshop on Parallel and Distributed Platforms in Industrial Products*, San Antonio, Texas, USA, October 1995. Workshop held during the 7th IEEE Symp. on Parallel and Distributed Processing, (SPDP-7).

[Mic97] Sun Microsystems. *Object Serialization Specification*, jdk online documentation edition, 1996,1997. docs/guide/serialization/.

[MP92] M. Mouly and M.-B. Pautet. *The GSM System for Mobile Communications.* Palaiseau, France, 1992.

[MP97] G. Müller and A. Pfitzmann, editors. *Mehrseitige Sicherheit in der Kommunikationstechnik: Verfahren, Komponenten, Integration.* Addison-Wesley, 1997.

[MRR98] D. Malkhi, M. K. Reiter, and A. D. Rubin. Secure execution of java applets using a remote playground. In *IEEE Symposium on Security and Privacy*, May 1998.

[MS97] V. Mayer-Schönberger. Generational development of data protection in Europe. In Agre and Rotenberg [AR97], pages 219–241.

[MvOV97] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography.* CRC Press, Inc., 1997.

[MvRS96] Y. Minsky, R. van Rennesse, and F. B. Schneider. Cryptographic support for fault-tolerant distributed computing. In *Seventh ACM SIGOPS European Workshop*, pages 109–114, Connemara, Ireland, sep 1996.

[MvRvRvS90] S. J. Mullender, G. van Rossum, R. van Renesse, and H. van Staveren. Amoeba – a distributed operating system for the 1990s. *IEEE Computer*, 23(5):44–53, May 1990.

[NAS] NASD. National Association of Securities Dealers, regulation public disclosure program. `http://www.nasd.com/`.

[NG90] J. Nehmer and T. Gauweiler. Design rationale for the MOSKITO kernel. In T. Härder, H. Wedekind, and G. Zimmermann, editors, *Entwurf und Betrieb verteilter Systeme*, volume 264 of *Informatik Fachberichte*. Springer Verlag, 1990.

[NIS94] NIST. *FIPS PUB 140-1, Security Requirements for Cryptographic Modules.* National Institute of Standards and Technology, January 1994.

[NL96] G. C. Necula and P. Lee. Safe kernel extensions without run-time checking. In *Second Symposium on Operating System Design and Implementation*, Berkeley, CA, oct 1996. Usenix Assoc. `http://www.usenix.org/publications/-library/proceedings/osdi96/necula.html`.

[Nut94] M. Nuttall. A brief survey of systems providing process or object migration facilities. *Operating Systems Review*, 28(4):64–80, October 1994.

[Obj] ObjectSpace. Voyagerpro. `http://www.objectspace.com/products/-voyager/index.html`.

[OLW97] J. K. Ousterhout, J. Y. Levy, and B. B. Welch. The safe-tcl security model. Technical report, Sun Microsystems Laboratory, mar 1997. `http://www.scriptics.com/people/john.ousterhout/safeTcl.ps`.

[OMG95] OMG. *CORBA: The Common Object Request Broker Architecture (Revision 2.0).* Object Management Group, July 1995.

[OMG98a] OMG. *CORBAServices: Common Object Services Specification.* Object Management Group, December 1998.

[OMG98b] OMG. *Mobile Agent System Interoperability Facilities Specification (MASIF).* Submission of GMD Fokus, IBM, Crystaliz, General Magic, and The Open Group, March 1998. `ftp://ftp.omg.org/pub/docs/orbos/98-03-09.pdf`.

[Ord96] J. Ordille. When agents roam, who can you trust? Technical Report Technical Report, Computing Science Research Center, Bell Labs, 1996.

[Orw48] G. Orwell. *1984.* Penguin Books, New York, 1972 (1948).

[Ous94] J. K. Ousterhout. *Tcl and the Tk toolkit.* Addison-Wesley, 1994.

[Pfi93] A. Pfitzmann. Technischer Datenschutz in öffentlichen Funknetzen. *Datenschutz und Datensicherung (DuD)*, 17(8):451–463, 1993.

[Pfl89] C. P. Pfleeger. *Security in Computing.* Prentice-Hall International, 1989.

[PK98] V. A. Pham and A. Karmouch. Mobile software agents: An overview. *IEEE Communications*, 36(7):26–37, July 1998.

[PPSW97] A. Pfitzmann, B. Pfitzmann, M. Schunter, and M. Waidner. Trusting mobile user devices and security modules. *IEEE Computer*, 30(2):61–68, February 1997.

[PS97] H. Peine and T. Stolpmann. The architecture of the ara platform for mobile agents. In Rothermel and Popescu-Zeletin [RPZ97], pages 50–61.

[Pur98] S. Purkis. Agent perl. `http://www.perl.com/`, December 1998. Agent/Agent-3.20.

[PWP90] B. Pfitzmann, M. Waidner, and A. Pfitzmann. Rechtssicherheit trotz anonymität in offenen digitalen systemen. *Datenschutz und Datensicherung DuD*, 14(5-6):243–253, 305–315, 1990.

[Qua] Qualcomm. Introducing the pdQ smartphone. `http://www.qualcomm.com/-pdQ/`.

[Ran94] K. Rannenberg. Recent development in information technology security evaluation – the need for evaluation criteria for multilateral security. In Sizer et al. [SYKFH94], pages 113–128.

[RC98] J. Reagle and L. F. Cranor. The platform for privacy preferences. Technical report, World Wide Web Consortium (W3C), November 1998. submitted for publication in the Communications of the ACM, `http://www.w3.org/P3P/`.

[RDEG98] A. D. Rubin and Jr. D. E. Geer. Mobile code security. *IEEE Internet Computing*, 2(6):30–34, 1998.

[RDFR97] M. Reichenbach, H. Damker, H. Federrath, and K. Rannenberg. Individual management of personal reachability in mobile communication. In *IFIP/SEC'97 13th International Information Security Conference*, Copenhagen, Denmark, May 1997.

[Red97] F.E. Redmond III. *DCOM: Microsoft Distributed Component Object Model.* IDG Books Worldwide, Foster City, CA, 1997.

[Ros92] R. Rosenberg. *The Social Impact of Computers.* Academic Press, 1992.

[Rot99] V. Roth. Mutual protection of co-operating agents. In Vitek and Jensen [VJ99], pages 277–287.

[RPM97] K. Rannenberg, A. Pfitzmann, and G. Müller. Sicherheit insbesondere mehrseitige IT-Sicherheit. In Müller and Pfitzmann [MP97], pages 21–28.

[RPZ97] K. Rothermel and R. Popescu-Zeletin, editors. *First International Workshop on Mobile Agents (MA'97)*, volume 1219 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 1997.

[RS98] K. Rothermel and M. Straßer. A fault-tolerant protocol for providing the exactly-once property of mobile agents. In *Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems*, pages 100–108, October 1998.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[Rue91] R. A. Rueppel. A formal approach to security architectures. In *EuroCrypt*, pages 387–398, Brighton, England, 1991.

[Rus98] J. Rush. Mobile agent (rover) workframe for python. `http://starship.skyport.net/crew/jrush/pyrover.html`, January 198.

[SBC+99] S. Staamann, L. Buttyàn, A. Coignet, E. Ruggiano, U. Wilhelm, and M. Zweiacker. Closed user groups in internet service centres. In *International Working Conference on Distributed Applications and Interoperable Systems (DAIS'99*, Helsinki, Finland, June 1999.

[SBH+97a] S. Staamann, L. Buttyàn, J-P. Hubaux, A. Schiper, and U. Wilhelm. Security in the telecommunications information networking architecture – the crystina approach. In *TINA'97 Conference*, Santiago, Chile, nov 1997. TINA-C, IEEE Publications.

[SBH97b] M. Straßer, J. Baumann, and F. Hohl. Mole - a java based mobile agent system. In M. Mühlhäuser, editor, *Special Issues in Object Oriented Programming*, pages 301–308. dpunkt Verlag, 1997.

[SBW98] S. Staamann, L. Buttyàn, and U. G. Wilhelm. Security in the telecommunications information networking architecture. In *IFIP/SEC'98 14th International Information Security Conference*, Vienna/Budapest, September 1998.

[Sch28] A. Scherbius. Ciphering machine. U.S. Patent # 1,657,411, 24 January 1928.

[Sch84a] F. D. Schoeman, editor. *Philosophical Dimensions of Privacy.* Cambridge University Press, 1984.

[Sch84b] F. D. Schoeman. Privacy: Philosophical dimensions of the literature. In *Philosophical Dimensions of Privacy* [Sch84a].

[Sch94] B. Schneier. *Applied cryptography.* Wiley, New York, 1994.

[Sch97a] F. B. Schneider. Towards fault-tolerant and secure agentry. In *11th International Workshop on Distributed Algorithms*, Saarbrücken, Germany, September 1997.

[Sch97b] H. Schulzrinne. Re-engineering the telephone system. In *IEEE Singapore International Conference on Networks (SICON)*, Singapore, April 1997.

[Sch98] F. B. Schneider. Enforceable security policies. Technical Report TR98-1664, Department of Computer Science, Cornell University, Ithaca, NY, January 1998.

[SG94] A. Silberschatz and P. B. Galvin. *Operating System Concepts.* Addison-Wesley, Fourth edition, 1994.

[SGB96] R. Simon, M. Gouker, and B. Barnes. *Windows 95 Win32 programming API Bible.* Waite Group Press, 1996.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[SNS88] 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.

[ST98] T. Sander and C. Tschudin. Towards mobile cryptography. In *IEEE Symposium on Security and Privacy*, May 1998.

[Ste94] R. Steele. The evolution of personal communications. *IEEE Personal Communications*, 1(2):6–11, 1994.

[Ste96] M. Stefik. Letting loose the light: Igniting commerce in electronic publishing. In M. Stefik, editor, *Internet Dreams: Archetypes, Myths, and Metaphors*, pages 219–253. MIT Press, 1996.

[Sun98] Sun Microsystems, Inc. *Java Remote Method Invocation - Distributed Computing for Java, White Paper*, September 1998. `http://www.java.sun.com/-marketing/collateral/javarmi.html`.

[SW97a] S. Staamann and U. G. Wilhelm. Corba as the core of the tina-dpe: A view from the security perspective. In *Object World: Distributed Object Computing in Telecommunications (DOCT'97)*, Frankfurt a.M., Germany, oct 1997. LogOn.

[SW97b] S. Staamann and U. G. Wilhelm. Cryptographic protection of connection integrity with interruption detection in tina. In *International Working Conference on Distributed Applications and Interoperable Systems (DAIS'97)*, Cottbus, Germany, October 1997. IFIP WG 6.1, Chapman & Hall.

[SW97c] S. Staamann and U. G. Wilhelm. Mehrseitige Sicherheit in der Telecommunications Information Networking Architecture. In Müller and Pfitzmann [MP97], pages 285–308.

[SYKFH94] R. Sizer, L. Yngström, H. Kaspersen, and S. Fischer-Hübner, editors. *Security and Control of Information Technology in Society, IFIP TC9/WG9.6*. Elsevier Science B.V. (North-Holland), St. Petersburg, Russia, August 1994.

[Tan88] A. S. Tanenbaum. *Computer Networks*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[Tan90] A. S. Tanenbaum. *Structured Computer Organization*. Prentice-Hall, Englewood Cliffs, NJ, 1990.

[Tim96] New York Times. U.S. workers stole data on 11,000, agency says, April 6, 1996.

[Tim99] New York Times. Creators of anti-abortion web site told to pay millions, February 3, 1999.

[TL95] T. Tannenbaum and M. Litzkow. Checkpointing and migration of UNIX processes in the condor distributed processing system. *Dr Dobbs Journal*, February 1995.

[TMMH96] C. F. Tschudin, G. Di Marzo, M. Muhugusa, and J. Harms. A distributed micro-kernel for communication messengers. Technical Report Technical Report No 110 (Cahier du CUI), University of Geneva, 1996.

[UN48] UN. United Nations, Universal Declaration of Human Rights. `http://www.unhchr.ch/`, December 1948.

[USC70] Fair credit reporting act of 1970. Administrative Procedure Act – 15 U.S.C. Sections 1681-1688t, 1970.

[USC74] Privacy act of 1974. Administrative Procedure Act – 5 U.S.C. Section 552a, 1974.

[USC88] Video privacy protection act of 1988. Administrative Procedure Act – 18 U.S.C. Section 2710, 1988.

[Vig98a] G. Vigna. Cryptographic traces for mobile agents. In *Mobile Agents and Security* [Vig98b], pages 137–153.

[Vig98b] G. Vigna, editor. *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.

[Vin98] S. Vinoski. New features for CORBA 3.0. *Communications of the ACM*, 41(10):44–52, October 1998.

[VJ99] J. Vitek and C. Jensen, editors. *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*. Lecture Notes in Computer Science. Springer-Verlag, New York, NY, USA, 1999.

[VST97] J. Vitek, M. Serrano, and D. Thanos. Security and communication in mobile object systems. In Vitek and Tschudin [VT97], pages 177–199.

[VT97] J. Vitek and C. Tschudin, editors. *Mobile Object Systems: Towards the Programmable Internet*, volume 1222 of *Lecture Notes in Computer Science.* Springer-Verlag, 1997.

[Was84] R. A. Wasserstrom. Privacy – Some arguments and assumptions. In Schoeman [Sch84a], pages 317–332.

[WB90] S. D. Warren and L. D. Brandeis. The right to privacy. *Harvard Law Rev. 4*, pages 193–220, 1890.

[WBS98a] U. G. Wilhelm, L. Buttyàn, and S. Staamann. On the problem of trust in mobile agent systems. In *Symposium on Network and Distributed System Security*, pages 114–124. Internet Society, March 1998.

[WBS98b] U. G. Wilhelm, L. Buttyàn, and S. Staamann. Protecting the itinerary of mobile agents. In *ECOOP Workshop on Mobile Object Systems: Secure Internet Mobile Communications*, Brussels, Belgium, June 1998.

[WCS96] L. Wall, T. Christiansen, and R. L. Schwartz. *Programming Perl.* O'Reilly, 1996.

[WD97] U. G. Wilhelm and X. Defago. Objets protégés cryptographiquement. In *Actes RenPar'9*, Lausanne, CH, May 1997. `http://lsewww.epfl.ch/~wilhelm/-CryPO.html`.

[Wei72] A. M. Weinberg. Science and trans-science. *Minerva*, 10:209–222, 1972.

[Wes67] A. F. Westin. *Privacy and Freedom.* Atheneum, New York, 1967.

[WHFG92] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.

[Whi94] J. E. White. Telescript technology: The foundation for the electronic market place. White paper, General Magic, Inc., 1994.

[Whi97] J. E. White. Mobile agents. In J. M. Bradshaw, editor, *Software Agents.* AAAI Press / The MIT Press, 1997.

[Wil92] U. G. Wilhelm. Music – eine interaktive und verteilte benutzerschnittstelle für den MOSKITO betriebssystemkern. Diplomarbeit, Fachbereich Informatik, Universität Kaiserslautern, August 1992.

[Wil95] U. G. Wilhelm. An architecture for more privacy in mobile communication systems. Technical Report 95/118, EPFL, Dept d'Informatique, April 1995.

[Wil97] U. G. Wilhelm. Increasing privacy in mobile communication systems using cryptographically protected objects. In *Verläßliche IT-Systeme 1997*, pages 319–334, Freiburg (Brsg.), November 1997.

[Win96] I. S. Winkler. The non-technical threat to computing systems. *Computing Systems, USENIX Association*, 9(1):3–14, Winter 1996.

[WL92] T. Y. C. Woo and S. S. Lam. Authentication for distributed systems. *IEEE Computer*, 25(1):39–52, January 1992.

[WLAG93] R. Wahbe, S. Lucco, T. E. Anderson, and S. L. Graham. Efficient software-based fault isolation. In *Proceedings of the 14th ACM Symposium on Operating System Principles*, pages 203–216, Asheville, NC, December 1993.

[WPW$^+$97] D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young, and B. Peet. Concordia: An infrastructure for collaborating mobile agents. In Rothermel and Popescu-Zeletin [RPZ97], pages 86–97.

[WS95] U. G. Wilhelm and A. Schiper. A hierarchy of totally ordered multicasts. In *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, September 1995.

[WSB99] U. G. Wilhelm, S. Staamann, and L. Buttyàn. Introducing trusted third parties to the mobile agent paradigm. In Vitek and Jensen [VJ99], pages 471–491.

[WSWZ97] U. G. Wilhelm, S. Staamann, G. Wolf, and J. Zöllner. Sicherheit in CORBA und TINA. In Müller and Pfitzmann [MP97], pages 267–284.

[Yao82] A. C. Yao. Protocols for secure computations (extended abstract). In *Proceedings of the 21st Annual IEEE Symposium on the Foundations of Computer Science*, pages 160–164, 1982.

[Yee94] B. S. Yee. *Using Secure Coprocessors.* PhD thesis, Carnegie Mellon University, 1994.

[Yee99] B. Yee. A sancturary for mobile agents. In Vitek and Jensen [VJ99], pages 263–275.

[YT95] 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.

[Zim94] P. Zimmermann. *PGP User's Guide.* MIT Press, Cambridge, 1994.

Appendix A

Enforcing an Itinerary

The TPE and the CryPO protocol guarantee the integrity of the agent platform to the agent owner and protect the code and state of a mobile agent against manipulation and disclosure, both in transit and during execution. These guarantees can be extended to formulate the requirements and the policy that have to be satisfied by a TPE in order to allow a mobile agent to follow a pre-defined itinerary.

This problem consists of ensuring that the agent does not visit any agent platform that is not on its itinerary, that it does visit all the agent platforms that are on its itinerary, and that it visits them in the correct order. Therefore, it suffices to make sure that the agent will migrate exactly to the TPE that is next on its itinerary. This resolves the problem of ensuring the safety part of this problem. The complementary problem of ensuring liveness (i.e., to make sure that the agent will actually continue its execution on the next TPE), is another difficult task that has to be addressed with the concepts discussed in Section 2.5.7. For reasons of simplicity, we assume that the agent owner provides the agent with a pre-defined itinerary upon its start and that the agent owner has also verified that all the TPEs on the itinerary enforce a sufficiently strong policy.

Goals:

- The mobile agent can autonomously enforce its itinerary.

Requirements:

- Basic protection
- The TPE allows mobile agents to perform and receive remote invocations.
- The operating system of the TPE implements standard process management functionality.

Policy:

a) The TPE allows the mobile agent to create new agents.

b) The TPE supports capabilities that allow an agent to perform process management operations on another agent.

c) the TPE provides a capability protected management operation to freeze[1] an active agent.

d) the TPE provides a capability protected management operation to marshal a frozen agent into a sequence of bytes.

e) the TPE provides a cryptographic interface to encrypt a sequence of bytes with a public key.

Discussion

The rules of the policy rely solely on regular operating system functionality that can be realized with memory protection and standard process management mechanisms. The capability that is required in rules *b)*, *c)*, and *d)* can be a simple random bit-string that is sufficiently long to prevent guessing attacks from other agents.

The use of this policy to guarantee that an agent will migrate from a TPE T_1 to a specific TPE T_2 is now straightforward. We assume that the original agent A_O carries the code for a send-agent A_S with it. A_O will now take advantage of rule *a)* and *b)* to instantiate A_S on the TPE (using a primitive similar to `fork` on the UNIX operating system) and provides it with a capability to access the management functionality on A_O as well as the certificate for T_2, which contains T_2's policy, the public key of T_2, and a network address for T_2.

A_S will now take the necessary actions to freeze A_O (rule *c)*) and to marshal A_O into a sequence of bytes (rule *d)*). For this, A_S needs the capability that it received from A_O. Then A_S will encrypt the marshaled A_O (rule *e)*) using the key in the certificate for T_2 and send it to the network address given in the certificate.

In order for A_S to ensure that A_O will actually continue its execution on T_2, it is necessary that A_S waits for a confirmation stating that A_O has successfully started on T_2. If A_S does not receive such a message after a certain timeout, it might simply re-instantiate A_O on T_1 and inform it about the unsuccessful migration attempt. A_O can then take appropriate actions to recover from this error.

Since no other agent on T_1 is capable to execute the operations that give access to A_O's code or data in any way and since T_1 will never disclose any information on any agent executing on it (basic protection), the only way that any information about an agent on a TPE can become available outside of this TPE is through proper actions of the agent itself (by sending regular messages or by using the presented approach for migration). Therefore, we conclude that the presented approach allows to guarantee that an agent will migrate exactly to the next TPE on its itinerary and will, thus, follow its pre-defined itinerary.

[1]The agent that will be frozen will be informed about this action, so that it can perform any necessary cleanup operations before being frozen.

Appendix B

Policy Issues

When an agent is at a certain agent platform and wants to move to another agent platform, it has to find out if the new agent platform will provide an adequate privacy protection. To assess this, the agent will obtain the policy of this agent platform P_{AP} (which is included in the reference to the agent executor, together with the address of the new site, the public key for confidential communication, the identity of the operator, a list of the available services, etc.). To make the assessment easier, we assume that the mobile agent itself has a minimal policy, say P_{MA}, which describes in some way the minimal protection requirements that the agent wants the new agent platform to provide.

The agent can now simply compare the two policies, which can lead to 4 different results:

- $P_{MA} = P_{AP}$ (very unlikely)
- $P_{MA} < P_{AP}$ (the new agent platform's policy is adequate)
- $P_{MA} > P_{AP}$ (the protection of the new agent platform is not good enough)
- $P_{MA} \sharp P_{AP}$ (the two policies are incomparable, which means that the protection is in some areas not good enough)

Based on this result, the agent can now decide that it will move to the new agent platform or search for a different agent platform which provides more adequate protection (for instance, the new agent platform might not provide a very good protection for locally executing agents, but it is willing to provide service to agents that are executing at a nearby special service provider (which may be more expensive).

Examples for this policy could be:

- the mobil agent requires a clock that is accurate to a precision of 10 seconds

  ```
  (clock-accuracy 10)
  ```

- the agent would like to be able to receive messages while executing on the agent platform

  ```
  (receive-messages TRUE)
  ```

A service provider would have a policy that says:

```
(NameOfServiceProvider ...
  (POLICY
    (clock-accuracy 3)
    (receive-messages TRUE)
    (send-messages TRUE)
  )
)
```

A comparison of the two policies would result in the conclusion that the policy is adequate, since it provides an accuracy that is better than the one we wanted and since it will allow us to receive messages. The fact that it also allows us to send messages is of no interest to us, since we don't intend to send any.

Another service provider might have the following policy:

```
(NameOfAnotherServiceProvider ...
  (POLICY
    (clock-accuracy 20)
    (receive-messages TRUE)
    (send-messages TRUE)
  )
)
```

Depending on the order that we have defined on policies, these two may be incomparable or my policy is larger than the provider's policy. In both cases it would be judged inadequate. The agent would decide not to move to this agent platform (without precisely having to know why it should not move to this agent platform).

Curriculum Vitae

Uwe Georg Wilhelm was born in Ludwigshafen am Rhein (Germany) in 1966. He attended primary school in Böhl-Iggelheim, from 1972 to 1976, then the Staatliches Gymnasium in Schifferstadt, and obtained the Abitur in 1985. From 1986 to 1992, he studied Computer Science, with a minor in Economics at the University of Kaiserslautern, obtaining a diploma in 1992. During this period, in the summer 1987 and 1989, he also worked on projects for Siemens AG in Speyer and München. In 1992 he went to the United States, and spent a year at Cornell University in Ithaca, NY working on research in distributed systems, security, and privacy, as guest of Professor Ken Birman. Then, for a short period, he worked as consultant for ISIS Distributed Systems. Since 1994 he has been working in the Operating Systems Laboratory at the Swiss Federal Institute of Technology in Lausanne (EPFL, Switzerland) as a reseach and teaching assistant, and Ph.D. student, under the supervision of Professor André Schiper.