Private Computing

and

Mobile Code Systems

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.dr.ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 21 november 2005 om 15:30 uur
door Kathy CARTRYSSE
elektrotechnisch ingenieur
geboren te Knokke-Heist (België).

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. R.L. Lagendijk

Toegevoegd promotor:
Dr.ir. J.C.A. van der Lubbe

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof.dr.ir. R.L. Lagendijk,	Technische Universiteit Delft, promotor
Dr.ir. J.C.A. van der Lubbe,	Technische Universiteit Delft, toegevoegd promotor
Prof.dr. R.W. Wagenaar,	Technische Universiteit Delft
Prof.dr. P.H. Hartel,	Universiteit Twente
Prof.dr.ir. H.C.A. van Tilborg,	Technische Universiteit Eindhoven
Prof.dr. A.A.C.M. Kalker,	Technische Universiteit Eindhoven
Dr. C. Witteveen,	Technische Universiteit Delft

ISBN-10: 90-90199-53-5
ISBN-13: 978-90-90199-53-5

Private Computing

and

Mobile Code Systems

Contents

Preface **v**

1 Introduction **1**
1.1 Mobile Code versus Software Agents . . . 2
1.2 Privacy . . . 3
1.3 Privacy and Mobile Code . . . 4
1.4 Problem Statement . . . 5
1.5 Thesis Outline . . . 6
1.6 Contribution . . . 7

2 Privacy Models **9**
2.1 Introduction . . . 9
2.2 Agent Privacy Model . . . 9
2.2.1 Model . . . 10
2.2.2 Trust and attackers . . . 11
2.2.3 Threats . . . 12
2.2.4 Privacy Requirements . . . 14
2.2.5 Problems Addressed in this Thesis . . . 16
2.2.6 Related Work . . . 17
2.2.7 Conclusions . . . 18
2.3 Mobile Code Privacy Model . . . 19
2.3.1 Model . . . 19
2.3.2 Assumptions . . . 23
2.3.3 Threats . . . 24
2.3.4 Problems Addressed and Approach . . . 25
2.3.5 Related Work . . . 25
2.3.6 Discussion . . . 26
2.4 Conclusions . . . 26

3 Agent Communication **29**
3.1 Introduction . . . 29
3.2 Problem Statement . . . 30
3.3 E-E-D: Private Agent Communication . . . 32
3.4 Example Applications . . . 34

3.4.1 Data-Collecting Agent Model . . . 35
3.4.2 Survey Model with Multi-Agents . . . 36
3.5 Conclusions . . . 38

4 Execution Privacy **39**
4.1 Introduction . . . 39
4.2 Problem Statement . . . 40
4.3 Execution Privacy Solution . . . 41
4.3.1 Function Protection . . . 41
4.3.2 Decision and Interpretation of Encrypted Data . . . 43
4.4 Evaluation . . . 45
4.5 Conclusions . . . 46

5 Agent Digital Signature **47**
5.1 Introduction . . . 47
5.2 Problem Statement . . . 48
5.3 Solution Outline . . . 49
5.4 Agent Digital Signature . . . 50
5.4.1 Introduction . . . 50
5.4.2 Agent Digital Signature . . . 51
5.4.3 Agent Digital Signature and Solutions to Double Signing Problem . . . 54
5.5 Conclusions and Discussion . . . 58

6 Secrecy Systems and Information Theory **61**
6.1 Shannon's Secrecy Model . . . 62
6.2 Information Theoretic Preliminaries . . . 63
6.3 Perfect Secrecy . . . 65
6.4 Unicity Distance . . . 71
6.4.1 Approach: Shannon . . . 71
6.4.2 Approach: Stinson . . . 74
6.4.3 Approach: van der Lubbe . . . 78
6.4.4 Conclusion . . . 80
6.5 Cryptographic Dilemma . . . 81
6.6 Conclusions . . . 82

7 Secrecy Systems and Plaintext Attacks **85**
7.1 Introduction . . . 85
7.2 Problem Statement and Assumptions . . . 86
7.3 Plaintext Attacks Based on Usage of Different Keys . . . 88
7.3.1 Perfect Secrecy . . . 89
7.3.2 Properties of Perfect Secrecy . . . 89
7.3.3 Unicity Distance . . . 91
7.3.4 Conclusions . . . 95
7.4 Plaintext Attacks Based on Usage of Identical Keys . . . 95
7.4.1 Definition of Maximum Secrecy . . . 96

7.4.2 Properties of Maximum Secrecy 97
7.4.3 Unicity Distance 104
7.4.4 Conclusions 105
7.5 Conclusions 106

8 Mobile Code Privacy **107**
8.1 Introduction 107
8.2 Ciphertext-only Protection 108
8.2.1 Perfect Secrecy 108
8.2.2 Example of Perfect Secrecy. 110
8.2.3 The Mobile Code Dilemma 111
8.2.4 Conclusions 112
8.3 Plaintext Attacks and Mobile Code 113
8.3.1 Maximum Secrecy 113
8.4 Conclusions 116

9 Unicity Distance in Mobile Code **117**
9.1 Introduction 117
9.2 Definition Unicity Distance for Mobile Code 117
9.3 Unicity distance for mobile code 118
9.4 Unicity Distance and Ciphertext-only attacks 119
9.5 Unicity Distance and plaintext attacks 122
9.6 Conclusions 124

10 Conclusions and Discussion **125**
10.1 Summary of the Results 125
10.1.1 Agent Privacy Model 125
10.1.2 Conclusions on the Theoretical Approach 126
10.1.3 Theory and Practice Combined 128
10.2 Discussion 128

A Notations **131**

B Perfect Secrecy **133**

Samenvatting **143**

Summary **147**

Acknowledgments **149**

Curriculum Vitae **151**

Preface

The research for this thesis was conducted within the PISA project. PISA (Privacy Incorporated Software Agent) was a European Union funded project within the fifth framework, which started in January 2001 and finished in January 2004. The objective of this project was to develop a software agent that was aware of the privacy risks and could respond such that privacy of the user is guaranteed according to the European directive on privacy.

In this multidisciplinary project the following parties participated: TNO, GlobalSign (Ubizen), National Research Council Canada, Finsa, Sentient Machine Research, Netherlands Data Protection Authority and Delft University of Technology.

The task of Delft University of Technology within the PISA-project was to provide cryptographic solutions to solve the privacy problems within a software agent environment. The results of this work are presented in this thesis.

K. Cartrysse, Delft, October 2005.

Chapter 1

Introduction

In today's society much of our communication involves mobile and wireless devices, such as mobile phones and laptops with a wireless connection to the Internet. Besides physical devices, (executable) code becomes mobile too. It is already common for code or programs to be transmitted over networks to be executed somewhere else, with java applets as the most common example. Agent technology may exploit this mobility even further as mobile agents can roam over the network and perform tasks on behalf of its user at a foreign host.

Being connected to the Internet also means that people are exposed to security and privacy risks. Many applications require registration of personal data, much of which is often not even necessary to run the application correctly. For example, a subscription for a digital newspaper requires often a registration where the user must provide his postal address, but this is not necessary to provide the service (an e-mail address would be sufficient). The result is that an average civilian does not know what kind of data about him/her is public and has no control over who can access his/her data. The European Union took the initiative to issue a directive on privacy, the data protection directive or directive 95/46 EC [34], which each member state implemented in a national law. This law defines when and how parties may access and collect personal data.

Most of these rules cover organizational aspects for collecting and processing personal data. However, if privacy can be guaranteed by technical solutions and tools, this may add to the protection level of privacy provided.

Privacy protection in an agent environment, where agents can be mobile, is especially challenging as the location of the agent is not always known beforehand, and therefore the level of trust may differ. The fact that an agent must sometimes execute its code in an untrustworthy environment results in many privacy threats.

Informally, intelligent agents can be seen as an example of mobile code. This thesis is about providing privacy in a mobile code environment, where not only eavesdroppers are present, but the execution environment of the mobile code itself may also be malicious.

1.1 Mobile Code versus Software Agents

The terms 'software agents' and 'mobile code' are often used equivalently and may cause confusion. Mobile code can be seen as code that is transported over the network and executed at some foreign location, for example a simple Java applet. Furthermore, a mobile software agent is also an example of mobile code.

Many definitions exist of 'software agents' and not one is standardized [43]. A definition of a software agent is given by Ted Selker [51] of IBM Almaden Research center:

> *"An agent is a software thing that knows how to do things that you could probably do yourself if you had the time"*

Although this definition is very general, it covers the principle that an agent helps to facilitate a certain task. A software agent may possess any combination of the following characteristics, but is not limited to them [45]:

Autonomous To be able to act without direct external intervention. It has some degree of control over its internal state and actions based on its own experiences.

Interactive To be able to communicate with the environment and other agents.

Adaptive The capability of responding to other agents and/or its environment to some degree.

Sociable Interaction that is marked by friendliness or pleasant social relations, that is, the agent is affable, companionable, or friendly.

Mobile To be able to transport itself from one environment to another.

Intelligent Its state is formalized by knowledge (i.e. beliefs, goals, plans, assumptions) and it interacts with other agents using symbolic language.

Cooperative The ability to coordinate with other agents to achieve a common purpose.

When a software agent has the mobility property, it can be seen as an example of mobile code. The term 'mobile code' will be used to denote the general concept of code that travels over a network and is executed at a remote location; the term 'mobile software agent' will be used to emphasize the objective of facilitating a certain task and all the corresponding consequences for this objective. For example, when mobile code is programmed to buy a flight ticket, specific tools like a digital signature are needed. In this case, the term 'mobile software agent' will be used to demonstrate its specific purpose. This means that the more practical solutions presented in this thesis will be specifically based on mobile software agent technology, and the more theoretical parts will be based on the general concept of mobile code.

1.2 Privacy

Privacy is defined by Westin [98] as:

> *"the claim of individuals ... to determine for themselves when, how and to what extent information about them is communicated to others."*

Privacy is one of the most important human rights issues of our evolving information age [4]. Informational privacy has two distinct characteristics:

1. The right to be left alone.

2. The right to decide for oneself what to reveal about oneself.

The Data Protection Directive of the European Union [34] is one of the directives of the EU that controls the informational privacy of an individual (others are the Directive of Telecommunications 2002/58/EC and the Digital Signature Directive and non-EU legislation 99/93/EC). Its purpose is twofold, namely providing a high level protection of personal data and enabling the free movement of data within the EU.

Legislation on and a definition of privacy provide a background for understanding the concept of providing privacy, but this knowledge does not provide insight into how to provide privacy to an individual by technical means. To be able to design privacy protection tools, one needs to define what type of data may provide a privacy threat. This may depend on the current environment of an individual. For example, privacy protection is not only about protecting the exchange of personal data (such as name, address, diplomas), but also about protecting information about actions an individual may take or orders he gives to other individuals. If an insurer can see that a client has dinner at a fast-food restaurant everyday, this may be a reason for him to increase this client's health insurance fee. Then the action of having dinner in a fast-food restaurant can be considered as private information.

The example above illustrates that privacy is more than just the protection of personal data. Privacy protection can be divided into four categories, such that in a system design, privacy protection tools can be added in a systematic manner. The first category is the individual's identity, e.g. their name. It is evident that providing someone's identity may be seen as a privacy threat.

The second category is protection of data about the individual. This data may be the type of degree he obtained, or which sports club he is a member of, etc...

The third type of privacy protection is protection of someone's actions. As the example above shows, when insurance companies start using access to data on a client's behaviour to ask a different fee, they invade that user's privacy. In real life it is impossible to follow everyone's actions everywhere as this would require a massive amount of manpower. However, in the digital world it has become relatively easy to track someone's actions and obtain private information about an individual in this way. Therefore, extra attention must be paid to this type of privacy protection.

The fourth is an extension of the third type of privacy, namely protection of actions taken by someone else on behalf of an individual. For example, when a user deploys his software agent, this agent has become an extension of the user and should protect

the user's privacy to some extent as the user would have done if he had performed the task himself.

In this thesis, the focus is on mobile code and agent technology, where these can be seen as concepts that are an extension of the user, i.e.. the fourth type of privacy is especially relevant and addressed in particular. Looking from the perspective of the code, it means that it must provide adequate privacy measures in the first three categories as the code must provide a level of privacy equal to that the user would have provided had he taken on the task himself.

1.3 Privacy and Mobile Code

The previous section gave a general explanation of the concept of privacy. As privacy protection involves more than just protection of personal data, this means that privacy protection in a mobile code environment is more than protecting the user's name and address.

Privacy protection for mobile code can best be explained by an example of mobile code. Consider a mobile software agent system, where a user instructs his personal agent to buy a flight ticket to New York if the price is less than $500,-. The agent is mobile and travels to a platform owned by an airline company where it issues the request to buy a flight ticket. It executes its program at the platform, where the input to the program is the offer from the airline; the output is the result of the decision. As the platform executes the code it is capable of observing it. It could therefore obtain the agent's strategy (e.g. buying a ticket less than $500,-) and change its offer for maximum profit. It may offer a flight ticket for the price of $499,- instead of $300,-, the price it would have offered if the strategy had not been visible. In this case, it is not necessary to protect the objective (acquiring a flight ticket to New York), but the strategy under which circumstances the purchase will be made requires protection. This type of protection is typically necessary for mobile code as the strategy and other personal information must be located in the code and this code will be executed at a possibly untrustworthy location.

From this example, two topics must be explained. The first is the difference between privacy and security for mobile code. The second is why privacy protection for mobile code differs from that for other systems.

The difference between privacy and security can be seen as follows. When security mechanisms are provided, this does not mean that the privacy is protected, too. For example, consider a database whose records contain personal information. The security of this database is guaranteed by access control mechanisms. The records may be encrypted such that only people with the correct key can have access. Privacy protection in such a database can mean that records are not linked to an individual or that different records are not linked to belong to one individual. Even the fact that it is known that data about a person is stored in the database can compromise his privacy. This example shows that when security mechanisms are provided, this does not mean that privacy is protected.

To be able to provide privacy in mobile code it involves protection of the code such

that hiding personal information is hidden during execution of this code and during transport over the network. In the flight ticket case, this means that the strategy and identity should be kept hidden from other parties.

The second issue that concerns privacy in mobile code is that what makes mobile code protection different from protection of other systems. The main difference is that the agent or mobile code is executed at an unknown location. The location may therefore be considered untrustworthy. As the execution environment is not trusted either, it means that the owner of the mobile code should take into account that the environment is capable of observing the entire code and its execution. In conventional IT systems, on the other hand, the execution environment can be considered trustworthy, and only during communication, one must provide privacy protection tools. Protecting mobile code means that the code itself must be protected, even during execution.

For example, consider again the flight ticket case within a mobile software agent system. When an agent purchases a ticket it may need to sign a document. The security of a digital signature depends on the secrecy of the private key. Furthermore, the private key can be considered personal information, as one can proof its identity by using the private key, therefore keeping the private key secret is also crucial for privacy. In a system where the execution environment can be trusted, this is not much of a problem as the access control mechanisms protect the secrecy of the private key. However, in the agent system, the execution environment cannot be considered trustworthy and therefore the private key must be protected such that even during execution, the private key is not accessible to the execution environment.

Concluding, the fact that privacy goes one step beyond security and, in case of mobile code, the execution environment cannot be trusted, makes privacy protection of mobile code a challenging research topic.

1.4 Problem Statement

The objective of this thesis is to develop tools and models to provide privacy to the owner of mobile code, when the code is executed in a foreign environment. The level of trustworthiness of the foreign environment is not known beforehand, and therefore it is considered to be untrustworthy.

The approach taken here is twofold. Two models are described, one for a practical approach and one for a theoretical analysis. In each of these models several research questions are defined and solutions and analyses are given to these questions. In the first model, cryptographic techniques are used to protect the privacy in a malicious environment. One of the conclusions of this approach is that it remains unclear what the limits of protection are in such an environment. To answer this question the second model is used, where information theoretic aspects are used to define the theoretical boundaries of protection in an untrustworthy computing environment.

1.5 Thesis Outline

The two different approaches of practical and theoretical work are both described in this thesis. In general, the outline is as follows. Chapter 2 describes two models that will be used. In chapters 3, 4, and 5, solutions are given for the first model and in chapters 6 up to 9, the second model is analyzed in detail. Conclusions are given in chapter 10.

In chapter 2, two models are presented that will be used throughout this thesis. The first model is the agent privacy model. It describes privacy protection for mobile software agents. The second model is the mobile code privacy model. This model is more general, so that it is is possible to derive theoretical boundaries for privacy protection of mobile code.

Chapter 3 provides a solution for secure data communication within the context of the agent privacy model. Not allowing the agent platform to view data it needs to exchange with other parties in the system is a complicated problem, as the data must be manipulated in such a way that only the receiving party can have access to it. This process of transforming the data must be done in such a way that at no point in time the agent platform has access to the clear text data.

In chapter 4, the problem of privacy in an agent's task is described. When agents perform tasks at a possibly untrustworthy host, the content of the task may need protection. This is called execution privacy. When the host knows the details of the agent's task, he may use these to his own advantage. A solution is presented in this chapter and the problem of making decisions when located at an untrustworthy host is investigated.

A third problem is providing an agent with mechanisms to obtain integrity and source authentication to an agent. Ordinary cryptographic tools like digital signatures do not fulfill the privacy requirements for an agent environment, and therefore a new method to have agents sign documents is proposed.

These three solutions to privacy threats form the more practical approach of the analysis of providing privacy in case of agent technology. The conclusions of this part of the thesis are given in the first part of chapter 10.

The second part of this thesis (chapters 6 - 9) provides an information theoretic approach to the concept of mobile code using the mobile code privacy model. The objective of chapter 6 is twofold; on the one hand it provides the introductory aspects of information theory that are necessary to understand the remaining chapters. On the other hand, it also shows the various approaches within information theory and points out the inconsistencies within these approaches by showing the differences.

Shannon provided a framework of information theoretic concepts for secrecy systems. One subject not taken into account is the concept of known-plaintext attacks. Chapter 7 extends Shannon's theory by using information theory to define levels of secrecy in case of plaintext attacks.

In chapter 8, the information theoretic concepts of Shannon's secrecy model are applied to mobile code. Especially in mobile code, plaintext attacks cannot be avoided and there the extension of Shannon's model can be applied. A new definition of perfect secrecy is given and a mobile code dilemma is derived.

Chapter 9 extends the information theoretic approach for mobile code by adding derivations for the unicity distance.

Finally, overall conclusions are given in chapter 10. In this chapter the practical approach and information theoretic approach come together and conclusions are given on the two models, but also on the consequences of results from one model to the other model.

1.6 Contribution

In this thesis several contributions have been made:

- In terms of providing secure communication, this thesis proposes a solution where the confidential data is not available to the agent platform at any time. It is especially useful for secure data exchange. This is an improvement to current solutions for providing confidentiality as in these solutions the platform is capable of observing the communication [31], [50].
- The second contribution of this thesis is the application of function encryption as provided by Sander and Tschudin [78]. They described how function encryption can be made possible and their solution is applied here to the practical case of task privacy. Furthermore, an analysis is done on how decisions can be made based on the outcome of encrypted functions.
- The final contribution within the agent privacy model is the development of an agent digital signature. This signature is based on an existing signature (ECDSA) [52], but by means of hiding the private key it provides a solution to the problem of private key exposure on an untrustworthy platform.
- The mobile code privacy model provides a framework to derive theoretical boundaries of mobile code protection. As far as the author knows this is the first approach of using information theory to derive theoretical limits for the protection of mobile code.
- Plaintext attacks cannot be prevented in the mobile code privacy model, therefore Shannon's theory on secrecy systems [83] must be extended towards plaintext attacks. This results in the notion of maximum secrecy, the minimum number of keys necessary to achieve this level of secrecy, and an expression for the unicity distance.
- The final contribution is to apply the results (by using information theory) to the mobile code privacy model. This provids theoretical limits of the level of protection for mobile code.

Chapter 2

Privacy Models

2.1 Introduction

This chapter describes two models used in this thesis. The first is called the agent privacy model and considers an example of mobile code (e.g. agent technology). In this model the mobile software agent is described and the other parties involved. It is a model that can be used in practice and it serves as a framework to develop practical solutions to privacy protection problems when making use of mobile software agents. Several problems will be derived in this model and practical solutions will be given in chapters 3, 4 and 5.

In the general case of mobile code a new model is developed that takes more theoretical concepts into account. Even if it is possible to obtain solutions in practical cases, it is of interest to analyze what level of privacy protection can be provided in theory. In this model attackers will be considered as attackers with infinite computation power and this approach will lead to the derivation of theoretical boundaries of providing privacy to mobile code. Providing privacy will be done by providing confidentiality. These theoretical boundaries are described in chapters 6 - 9.

Both models are explained in detail, followed by a list of threats and requirements. Based on this, several problems are described that will be addressed in this thesis. This is followed by related work on each model.

2.2 Agent Privacy Model

Currently, agent technology is seen as one of the technologies that will play a key role in future IT-applications. Mobile agents can be seen as a special case of mobile code. The objective of this section is to define clearly the important players in a mobile agent system. Each of these players can involve or produce risks and based on these risks we can define the requirements for the privacy enabled multi-agent system to be designed. Furthermore, this section explains why conventional solutions are not always solutions for agent technology.

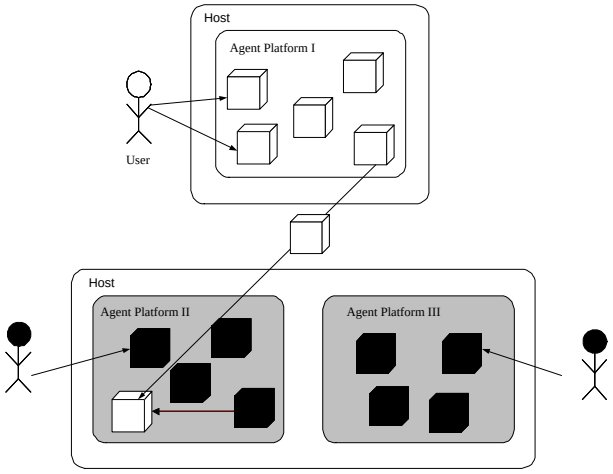

Figure 2.1: Overview of agent system

2.2.1 Model

Figure 2.1 gives a general overview of an agent system. The following players can be present in the system:

Mobile software agent: A piece of software that performs a certain task on behalf of its user. It is a software agent with the property of being able to travel over the network and being executed elsewhere. The agent may or may not have other possible properties.

Users: Users are the agent owners. One user can own multiple agents. In general, the agent performs a task on behalf of its user. The user can be in direct contact with its agent(s).

Agent creator: The agent creator is the person who or organization that designed and implemented the intelligent agent such that it can perform the tasks for which the user wants to use it (not shown in figure 2.1).

Agent platform: An agent platform provides generic agents and the physical infrastructure in which agents can be deployed. This can be the user's computer, but also a mobile phone or some server on the Internet

Host: A host is an entity that hosts one or multiple agent platforms. Whenever this does not cause confusion, the term 'agent platform' and 'host' may be used interchangeably.

Each of these entities has its own role in the system and its level of trust. The following scenario is used. At the user's location, the agent is instructed to perform a certain task. The user provides the agent with sufficient information to fulfil its job;

this may include personal information. The mobile agent leaves the user's computer and travels over the network to perform its task. After complete execution of its task, the agent returns to its user and gives the result to its user.

Note that in this model, one user may own many agents and these agents can be considered as the core elements in the model. All privacy protection tools developed are to provide privacy with respect to the agent, as the agent can be considered as an extension of the user.

2.2.2 Trust and attackers

Security and privacy solutions can only be designed with a good understanding of the entities in the system and their level of trust as seen by the other entities. The amount of trust is defined with respect to the agent and its owner. The amount of trust that a user has in another element of the system is determined by the type of attacker the user considers this element to be. Consider a system in which one user owns multiple mobile agents located on an agent platform.

If an element is considered trusted, it means that the entity executes all protocols correctly and does not attempt to attack other elements or eavesdrop on communications.

One type of attacker that is considered here is the passive attacker. This attacker is curious, but does perform its attacks only in a passive way. The attacker is considered to have polynomial computation power [67]. It executes protocols correctly but is curious in the sense that it tries to eavesdrop on communications.

A second attacker is actively trying to attack the system. It has polynomial computation power. It does not necessarily execute protocols correctly and whenever possible it will try to obtain knowledge of private information.

With respect to the user, the following assumptions are made about the type of attackers the various entities can represent.

Agent creator: The agent creator is fully trusted, as the agent is seen as a product one buys in full confidence. It is the agent creator who may add or provide privacy protection tools.

Agent: The agent owned by a user is fully trusted by its user. Agents owned by other parties than the user are considered to be untrustworthy. These agents are attackers with polynomial computation power. These agents do not conspire with the agent platforms.

Agent platform: The agent platform owned by the user can be considered to be fully trustworthy. All other platforms are seen as untrustworthy in the sense that they are curious. These platforms execute the agents correctly, but they are interested in the agent's content, its personal data, and the details of the task it is supposed to perform. This curious host is also called a 'passive attacker', as it does not actively change the agent's content but eavesdrops on everything it has access to. The agent platform has polynomial computation power. Furthermore, it is assumed that platforms do not conspire with each other.

Host: The trust level of a host corresponds to the trust level of the agent platforms that are located on the host.

User: All other agent owners except the user whose privacy is protected are considered to have an equal trust level as their corresponding agents.

In the next section a list of threats is given. It is a general list that does not take these assumptions on trust into account. The list of threats is followed by requirements where these trust levels are taken into account.

2.2.3 Threats

For each player, a list of threats is given. The threats are general and for each threat the list describes when this general threat can be considered a privacy threat. All these are threats to the user's privacy protection.

Threats towards agents

The threats towards agents can be divided in three different categories based on the state of the agent in which a threat may occur. The three categories are data storage, communication, and data processing. The first category, *data storage*, covers the storage and access of static data in the agent. Examples are storing data in the agent at initialization or the storage of computation results.

Threats in this category can be defined as:

Unauthorized access. If the agent is not well protected, other entities may have access to confidential data. When this data is personal data, this becomes a privacy threat.

Unauthorized altering of data. This threat occurs when other elements in the system are capable of altering or erasing data that is stored in the agent. The result may be that the agent cannot execute its tasks correctly anymore. For example changing its computation results may be in the interest of other parties. This general threat becomes a privacy threat when personal data is altered.

The second category is the category of *communication.* During its life cycle, the agent needs to communicate with other entities in the system. Different threats are present during communication.

Eavesdropping on communication. A third party may be capable of eavesdropping on the communication line. Note that in the case of agent technology, also the agent platform may eavesdrop on a communication. This makes this threat very serious as the platform can also observe encryptions and decryptions. Obviously, when personal data or actions are involved, this is a privacy threat.

Masquerading. Other agents may pretend to be an agent they are not. By pretending this they might learn more about the original agent. This can be done by talking to agents or sites the original agent is supposed to communicate with. Communicating with other agents about the concerned agent may also reveal information on the agent's identity. This is typically a privacy threat.

Altering data during communication. If this is not prevented, other parties may alter data during communication. This can be a privacy threat as it is important for personal data to be exchanged that these are correct.

Fraud with user's identity. This threat consists of a user falsely claiming to be an agent's owner. A consequence is that the agent will trust that user and may give all its information to that user or respond to orders given by the malicious user. This case poses a threat to the privacy of both user and agent.

The third category, *processing of data*, covers the actions of the agent. It involves similar threats as in the other two categories and several additional ones.

Unauthorized access to an agent's actions. When third parties are capable of obtaining knowledge of the agent's actions and strategies, this may be a serious threat as these attackers can change their own actions according to their obtained knowledge. This is the threat represented by the flight ticket example.

Duplication of the agent. It may be possible to clone the agent without permission. The cause of this threat can come from all other entities of the system. This threat may impact the network performance, but it may also cause serious problems for the user. If an agent is cloned, each clone can perform the same tasks as the agent and therefore the task may be performed multiple times, which could damage the user.

Duplicating agent's actions. When an agent is executed at an agent platform, this platform might gain control over the agent and send the agent to another platform to perform the same function. In this case the agent would perform the same function twice, which could have negative consequences for the user and if the function is protected, it could obtain more information on the content of the function. This can have privacy and security consequences.

Altering the agent's functionality. When the agent has a lack of security it may be possible for other entities to change one or multiple of its functions. A consequence can be that the user loses control over his agent, without knowing it. Two things should be considered: first it should not be possible for other entities to change the agent's functionality. Second, if the first condition cannot be guaranteed, somehow the user should be notified if his agent's functionality has been changed, as it is both a security and privacy threat. The change of functionalities is a security threat, but when these changes affect the level of privacy the agent provides, it also becomes a privacy threat.

This list is not an exhaustive overview of threats towards an agent, but the most important ones are covered. For an extensive list, the reader is directed to [26]. The list above is structured in categories and this will be helpful to set up requirements. Note that in each category, altering and unauthorized access are considered serious threats.

Threats towards the User

The threats described in the previous paragraph involve the privacy of the user, only indirectly as they are posed to the agent. However, there are also several threats that use the agent to directly attack the user.

Access to the user through the agent. When the agent is not adequately protected, it may be possible for other entities to gain access to the user via the agent. They can either do this by accessing user data in the agent or by setting up a communication link between agent and user. This threat has both security and privacy implications.

Agent is set up against user. If a malicious entity is able to change the agent in such a way that it is set up against the user, it may give information about the user to this malicious entity. But it may also order the agent to perform in another way, as it should have done if the original user were in control. The damage could be enormous, because the agent could start doing illegal things without the user knowing it. As the user still has trust in his own agent, privacy may easily be compromised.

Threat coming from agent provider. If an agent provider owns the agent instead of the user, the user should be aware of privacy threats caused by the provider.

Threats towards Agent Platforms

When agents operate on an agent platform, certain assumptions are made about the trust level of the platform. Threats or attacks towards the agent platform may result in false assumptions about the trust level and therefore privacy may be compromised.

Trust in the agent. When an agent platform trusts an agent while it is untrustworthy, the agent may damage the platform (e.g. it may contain a virus) or pretend to be an agent it is not.

Furthermore, the general threat exists that elements are not accessible (for example in case of a denial of service attack). Based on this list of threats, we can create a list of requirements.

2.2.4 Privacy Requirements

This thesis focusses on providing mechanisms to the software agent that guarantee the user's privacy when making use of agent technology. Hence from all possible privacy and security threats, we only study those related to the agent, e.g. the threats to the agent itself and its interaction with the various elements in the system.

To set up a list of requirements, we use the same categories as in the previous paragraph.

Data storage. For data storage, the following requirements can be listed.

- Protection of the data stored in the agent should be such that no eavesdropper is capable of obtaining this data.
- An integrity mechanism is required that prevents alteration or erasure of data.
- To perform its task, the agent may have to interact with other entities and this can influence the computation result (result after execution of the code). These computation results should be protected such that other entities cannot change or read them.

Communication. The following requirements are set when the privacy incorporated software agent communicates with other entities in the system.

- It should not be possible for a third party to eavesdrop on a communication between the agent and its communicating partner such that the content of the conversation can be observed.
- During communication, adequate integrity mechanisms must operate.
- During communication, an agent will receive private data from other entities and it should have the means to securely store this data.
- A mechanism should be present allowing the communicating parties to authenticate each other.

These requirements sound similar to the security requirements in an ordinary IT system. However, the impact of our requirements and therefore the solutions are different because the underlying computing platform may not be trustworthy.

For example, normally when two agents wish to set up a confidential communication, they will generate a session key using their public-private key pair. However, the operation of generating a session key involves decryptions, and whenever a decryption takes place, the agent platform is able to gain access to the decrypted data. This simple example shows that providing secure agent communication is not trivial.

Processing. For the category processing or actions, the following requirements can be set:

- Execution privacy. This means that the code must be kept confidential from the other parties. The agent must be able to execute functions and reason based on the outcomes of these functions. Both processes should be protected against eavesdroppers.
- It should not be possible to clone or duplicate an agent. For cases where this cannot be prevented, a detection mechanism should be developed.
- Some integrity mechanism must be provided to prevent altering of the agent's functionality.

2.2.5 Problems Addressed in this Thesis

Based on privacy threats and requirements, this thesis addresses three problems to protect privacy in an agent environment. In all categories where requirements are defined, the most important requirements are to provide

- confidentiality
- integrity
- authentication.

The problems addressed in this thesis are therefore problems with respect to these requirements. The following problems are addressed to protect privacy in an agent environment. With respect to confidentiality solutions are to be found in the categories communication and processing. The possibility to provide confidentiality is absolutely necessary to be able to fulfill the definition of privacy as was given in section 1.2. In order to obtain "the right to be left alone" one must be capable of hiding information from other parties, which means mechanisms to provide confidentiality should be present.

The first problem is the problem of secure communication between agents when located on a possibly untrustworthy platform. The objective is to design a method for communication that makes it impossible for the platform to eavesdrop on the communication. During communication the agents exchange data. Securing the agent's communication is only useful when the data that is to be exchanged is stored in a secure manner, otherwise the platform may have access to the data without having access to the communication. Therefore, also the problem of confidential data storage is addressed as secure communication is not provided in an agent system when confidential data storage is not guaranteed.

The second problem addressed in this thesis is in the area of providing privacy during processing of the agent. Each agent is programmed to execute a task, and this task should be kept private from a third party but also from the agent platform. Only the problem of confidentiality of a task is addressed here.

Finally, the problem of providing the agent with an integrity mechanism is addressed. Except from integrity the agent should be able to prove its identity, taking into account the privacy issues mentioned above. A new agent signature will be developed for this purpose.

Problems like prevention of cloning and prevention of altering the agent's functionality are not addressed in this thesis. These problems are important to be solved, but first the above three problems are addressed to provide solutions to the privacy protection problem.

Only these three problems are addressed, because it is chosen to develop solutions for the most important requirements with respect to privacy protection. Moreover, the results that will be presented in this model give rise to questions about the theoretical limits to providing privacy to mobile code in general. Therefore, the second part of this thesis covers this subject.

2.2.6 Related Work

Over the years, much research has been done in the area of privacy in conventional IT systems, and many adequate solutions have been presented. The term PET (Privacy Enhancing Technologies) is used to describe all types of technologies that provide privacy to a user [47]. Typical cryptographic techniques that can be called PET are blind signatures [27], [28], [17], partial blind signatures [2], and pseudonym systems [60]. Each of these techniques has its own applications but they are all based on the assumption that the computers where the computations are performed can be completely trusted, which is not the case in a mobile agent system. PET are mainly used in applications where the privacy aspects determine the success of the product. An application where privacy is of great importance is electronic voting. Several electronic voting schemes make use of blind signatures. A blind signature allows one to sign a message without being able to read the content of the message. For the electronic voting application, this means that the voting committee signs the vote to declare it is a legitimate vote, but it is not able to view who the voter voted for. Hence, by using blind signatures, anonymity can be provided. In electronic cash applications [28], [13], [29], a partial blind signature [2] can be used to provide anonymity, in the same way as in electronic voting, but here the amount of money should not be blinded, but only the user's name. A partial blind signature has this property: the amount of money can be public, but the name is kept hidden. A fair blind signature [87] makes it possible to reveal the connection between the message and signature in case of dispute.

Then there are other cryptographic techniques that may be of importance for providing privacy in a system. Zero-knowledge techniques [91], [90] allow one to prove the knowledge of something without actually revealing the secret. Using zero-knowledge techniques one can prove knowledge of a password without providing it. A second useful concept to provide PET is secret sharing schemes [90]. A (t,w)-threshold scheme is a method of sharing a message M among a set of w participants such that any subset consisting of t participants can reconstruct the message M, but no subset of smaller size can reconstruct M.

Privacy enhancing technologies can also be in the area of network privacy. Examples are the Mix network [30], onion routing [44], [69], and the crowds system [71], [70]. Measuring the level of anonymity is shown in [37] and [82].

Many solutions have been proposed to protect the user's privacy. However, these have several drawbacks. First, the solutions described above all have the assumption that the computations take place on a completely trustworthy computer. This is an assumption that cannot be made in agent systems if full benefit is to be taken from agent's characteristics. A second drawback is that all these techniques provide privacy to the user's identity (blind signatures) or to privacy-sensitive data (zero-knowledge techniques, secret sharing schemes), but they do not provide privacy about one's actions, which is necessary in the case of agent systems. Nevertheless, these privacy techniques will prove to be useful in the context of agent technology.

In addition to PET for conventional IT systems, many security techniques have been invented for mobile software agents to protect them from malicious hosts [68].

Several schemes have been described to provide integrity of partial results [88], [53]. In [53], a solution based on PRAC (Partial Result Authentication Code) is given. A PRAC provides forward integrity of the agent's partial results. Forward integrity means that the results obtained at the previous hosts cannot be modified. A second method to provide computation integrity is identifying a trusted host. Only if a host is trusted, computation integrity is ensured, [74], [80]. A different approach to provide integrity checks of code comes from the field of watermarking [32], [33]. A special data structure is embedded in the program such that even after the execution of the program the watermark can be detected by the user which makes it possible to detect any malicious modification of the program. Many schemes provide accountability in the sense that afterwards, the computations can be verified. In [95] and [9], execution traces are computed by the host which can be verified later to detect whether suspicious computations have taken place. Farmer et al. [40] describe a method to check the state of the mobile agent for inconsistencies.

Many articles have been written about confidentiality, and most of them define confidentiality for agents as protecting their code such that it is impossible to determine the agent's strategy. Hohl [48] describes a mechanism called "time limited black box security" where the idea is to obfuscate the source code such that it takes more time to understand the code than the programmed time limit. A more cryptographic method is presented in [76] where Sander and Tschudin encrypt functions that can be executed in its encrypted form. This method works for polynomials and rational functions [42]. Young et al. [79] extended the results to all functions computable by circuits of logarithmic depth and further generalized to arbitrary functions, provided they can be represented by a polynomial-size circuit. As far back as 1990, Abadi and Feigenbaum [1] described a method that provides confidentiality for circuit evaluation. A disadvantage of this method is that many interactions are required to provide confidentiality. Many other solutions have been published to provide secure circuit evaluation, but none of them is very practical and efficient. Loureiro et al. describe how functions can be hidden using coding theory [58]. Several more practical methods have been proposed, but they are all based on either trusted hardware located at the host [103] or on the presence of a trusted third party [3]. In addition protecting the agent's code, some data the agent receives must be protected against eavesdropping. A method is to use sliding encryption [105]. It provides encryption for small amounts of plaintext resulting in small amounts of ciphertext without loss of security.

2.2.7 Conclusions

This section described the context of the research done in this thesis. Assumptions were made to set the boundaries in which the solutions to be presented should provide an adequate level of security and privacy. The main assumption made here is that all agent platforms (except the one owned by the user) are considered to be curious. The agent's code is executed correctly and is not changed by the platform. However, the platform is interested in the agent's content and his task. If the platform succeeds, he can have access to the agent's strategy, which could influence the agent's actions. The fact that the agent considers the platform to be curious means that the agent has some level of confidence in the platform, but is not prepared to share personal secrets or

information. Compare this to the daily situation where we are sometimes prepared to show our passport, but we are not prepared to give it to another person. Furthermore, solutions will only be considered that are implemented in the agent. The agent platform is seen as a given environment, where no interference from outside is possible.

Based on the numerous threats to the privacy of the user in an agent environment, it was possible to set requirements. A conclusion from these requirements is that mainly three aspects will be addressed, namely secure communication, execution privacy and the problem of an agent digital signature.

The assumptions made in the agent privacy model are realistic for practical applications, but the model is not based on the strongest possible attacker. Therefore, to be able to define theoretical privacy protection limitations, a model is necessary that considers an attacker that is not limited by practical issues, although again he is considered to be curious. This is the case in the mobile code privacy model.

The following chapters (3, 4 and 5) propose a solution for each of these problems. Most of the solutions proposed were also published in [24].

2.3 Mobile Code Privacy Model

The second model, called the mobile code privacy model, covers the general concept of mobile code executed in an unknown environment. The model is generalized compared to the agent model as it is no longer specific for an agent application. However, it is also more simplified to be able to derive theoretical boundaries for privacy protection of mobile code.

2.3.1 Model

Figure 2.2 shows the different elements that play a role in the mobile code model. The mobile code is initiated in the user's environment and afterwards travels over the network to be executed at various hosts.

Host: The host provides the execution environment where the various mobile codes can be executed.

User: The user is the owner of the mobile code. The mobile code is programmed by the user and sent over the network to be executed elsewhere.

Trusted party: The trusted party is a party that can perform computations. It is trusted by all players in the system.

Mobile code: Mobile code is defined as code that may travel over the network and is executed at a foreign location. For this model, mobile code is seen as a set of functions that is executed in an environment unknown beforehand. For reasons of simplicity, the mobile code is modeled as one function, but the results can easily be generalized towards multiple functions.

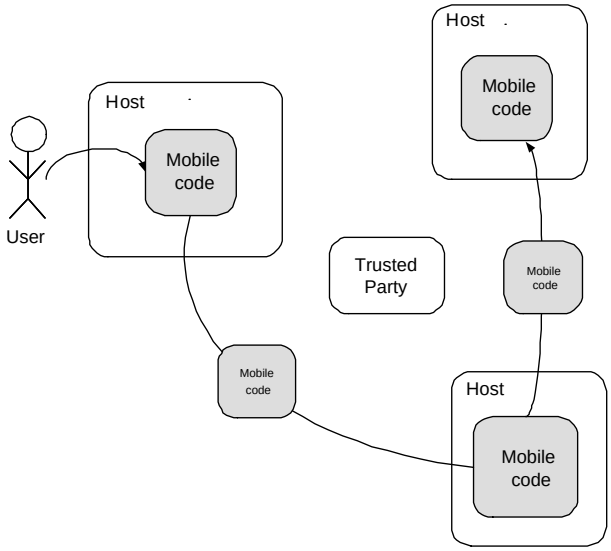

Figure 2.2: Elements in mobile code privacy model.

The objective is to derive theoretical boundaries of the privacy level that can be provided in this model. Privacy is protected by following the approach of providing confidentiality to the mobile code.

A more detailed view of the model is shown in figure 2.3. The following process takes place. A user wishes to execute function F on a foreign host, but F should be kept confidential. The user protects its code F by encrypting it. All the user's operations performed are done in a trusted environment as this environment is owned by the user. The user encrypts code F using a key K. The result of the encryption is G, which is again a function that can be executed but in an encrypted form. G is in this case equal to the protected mobile code (if the mobile code consists of multiple functions, the protected mobile code will consist of multiple G functions). Function G can be correctly decrypted by using key K. A third party (e.g. the host) may provide some input parameter X to the encrypted function G. The result is denoted by $U = G(X)$. When F is executed with input parameter X, the result is denoted by $Y = F(X)$. The set of all (X, Y) coordinates represents the function F as does the set of all (X, U) coordinates for function G. Because F and G are related to each other by key K, the sets of (X, Y) and (X, U) coordinates are related to each other by the same key K. Therefore, in theory it will always be possible to decrypt U to Y based on key K, although it does not mean an efficient algorithm exists to do so (table look up may sometimes be the most efficient algorithm).

In general, after execution of function G a decision must be made what the next action of the mobile code will be. In some cases it may be possible to make such a decision based on the outcome of U (as is demonstrated in chapter 4). In this case, the

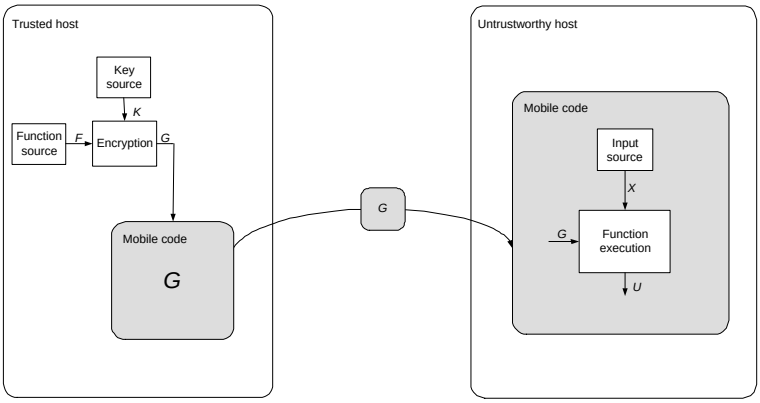

Figure 2.3: Mobile code model as it will be used in this thesis.

decision function will be stored as part of the mobile code and can be executed at an untrustworthy location.

However, it may not always be possible to make decisions based on the encrypted value U, such that U must first be decrypted to Y before a decision can be made. It is not possible to let the mobile code perform this decryption as this would mean that a key must be part of the code. This key is at least related to the key that can correctly decrypt G or is equal to that key. Therefore, this decryption cannot be part of the mobile code. The trusted party is added to perform this decryption. The user provides the trusted party with the correct key to decrypt U values towards the correct Y-values. These Y-values are then sent back to the mobile code, where a decision can be made (figure 2.4). It is important that the mobile code receives data on which it can base a decision. The decision is not made at the trusted party for a practical reason. If the decision would be made at the trusted party, the user must provide the decision criteria to the trusted party each time it uses mobile code. By just providing the decryption key, no interaction is necessary between the user and trusted party as long as the same key is used.

A second reason why it is an advantage to include a trusted third party is that when the mobile code receives Y from the TTP, it prevents the attacker from computing a benificial X value. Because the attacker does not know the relation between U and Y, he is not capable of computing an X value given Y. Note that a large advantage of using public key encryption in practice is that the attack can be prevented where the optimal input X can be computed from the output of the function. By encrypting X using the user's public key this attack is prevented, without the help of a trusted third party.

Protection of mobile code is here seen as providing confidentiality to the functions (F) stored in the mobile code. The decision criteria are outside of the scope of this thesis, but the reasoning that sometimes U must be decrypted and sometimes not, is important for the possible attacks in the mobile code privacy model.

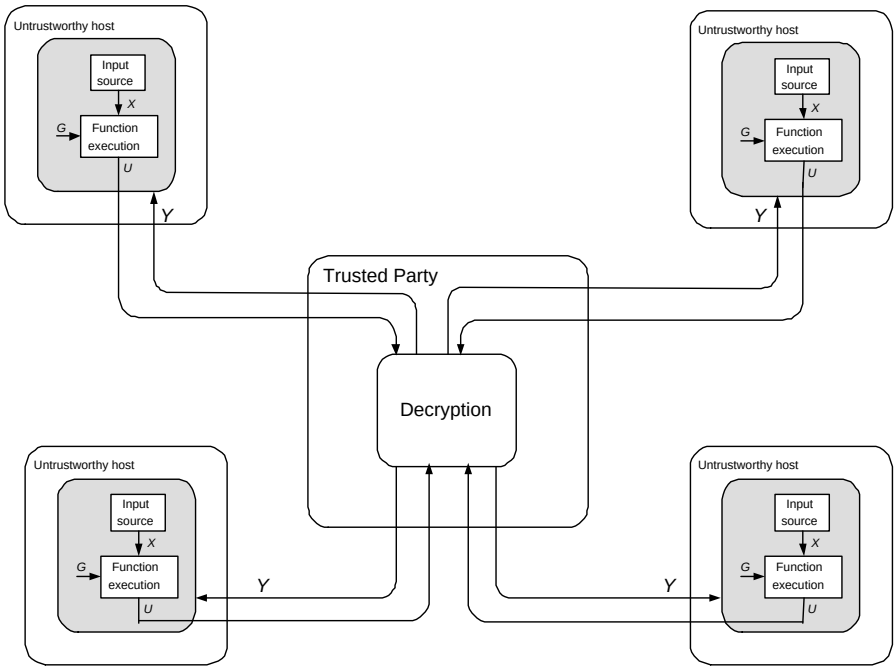

Figure 2.4: A trusted party is added to the model to perform decryptions of U values

2.3.2 Assumptions

The model of figure 2.3 contains several assumptions, which are listed in this paragraph.

Host: All hosts are considered to be untrustworthy, except the host owned by the owner of the mobile code (user). These untrustworthy hosts are curious in the sense that they observe what the mobile code does that is executed in its environment, but they do not alter the code. The code is executed correctly and only once. The hosts may conspire with other elements in the system. Furthermore, the hosts have unlimited capabilities for observing the codes, e.g. they have unlimited computation power and memory resources.

User: The user (owner of the mobile code) is considered to be completely trustworthy.

Trusted party: This party is trusted by all participants and does not conspire with any other element in the system.

Mobile code: Various assumptions are made in order to model mobile code.

Trust level: The mobile code itself is considered to be trusted by the user. It is assumed that the code's functionality is not altered during its lifetime. This assumption is connected to the trust assumption about the hosts.

Content of mobile code: As described in the mobile code model, mobile code is modeled as one mathematical function that can be executed elsewhere; other parties may provide an input X to the code. This mathematical function consists of an alphabet, which consists of numerical values and operators. By placing them in a correct order, a function is generated. For example an operator is followed by a number or variable. When the code F is encrypted, the result is an executable function G. Furthermore, mobile code exists of decision logic, but privacy is provided by adding confidentiality to the function F, therefore, the decision logic is not considered itself. It is important to know that a decision takes place as the method that is used determines whether or not U must be decrypted. Because privacy protection means in this case encryption of F, the term 'mobile code'is used here to denote F.

Encryption: The assumption is made that the encryption method used to encrypt the mobile code is based on a symmetric algorithm. This assumption is based on the objective of the model. The objective is to define the theoretical limits of mobile code protection. This means that not only practical attackers (i.e. those with limited resources) should be taken into account, but also attackers with unlimited resources (e.g. the host), as we saw in the assumption about the host. As this type of attacker is used in this model, information theory will be used to derive the theoretical boundaries. However, Maurer derived an upper bound on the uncertainty of a

key K generated by two parties Alice and Bob with a possible eavesdropper Eve [64]:

$$H(K) \leq min(I(A;B), I(A;B|E)), \tag{2.1}$$

where Alice, Bob and Eve know random variables A, B, and E, respectively, which are jointly distributed according to some probability distribution P_{ABE}. When the random variables A and B are chosen independently, no secrecy is possible for the key. From this it is clear that Alice and Bob cannot generate an information-theoretically secure secret when they do not share at least some partial secret information initially when they can only communicate over a public channel (accessible by Eve). It also means that there exists no unconditionally secure public key cryptosystem or public key distribution protocol [65]. In a public key cryptosystem no secret is shared initially by the sender and receiver, therefore it is theoretically impossible to achieve perfect secrecy for a message by public key encryption. By the same argument, it means that an information theoretic approach cannot be used to determine theoretical limits of privacy protection to mobile code when public key cryptography is used. Other approaches as complexity theory can be used to determine levels of secrecy [36] in practice. Therefore, it is assumed symmetric encryption is used.

Using these assumptions and the presented model, we can now set up a list of threats and describe the problems that will be addressed in this thesis within the context of the mobile code model.

2.3.3 Threats

However, the threats in the mobile code privacy model are less general as the model is such that the confidentiality of the code is protected and that protection takes the form of symmetric encryption. The threats of section 2.2.3 were with respect to the various elements, whereas here the solution to provide privacy (e.g. encryption of the function F) is part of the model and therefore the threats are with respect to this solution. Given this, the following threats can be defined.

The first threat is that the mobile code can be intercepted during transmission, while it is travelling over the network, such that an attacker is capable of understanding the content of the code. A similar threat exists to confidential messages that are sent over a network.

The second threat is that the mobile code may be observed while being executed. The host is capable of observing the code's execution. Because it is assumed that the host has unlimited resources, this is a serious threat.

These two threats can be considered the main threats when protecting the confidentiality of mobile code. In more detail, for these threats two types of attacks can be

considered.

Ciphertext-only attack. The first type of attack is the ciphertext-only attack. The attacker has access to a ciphertext, in this case protected mobile code, and the objective is to determine the plaintext or the key, or both.

Plaintext attack. In a plaintext attack, the attacker has access to a (part of the) plaintext and its corresponding ciphertext. This plaintext can be chosen (chosen-plaintext attack) or given (known plaintext attack). This distinction is not relevant to the mobile code privacy model. The analysis will not be on a specific algorithm, but on the general concept. The term plaintext attack will be used to denote an attack where plaintext and corresponding ciphertext are known to the attacker.

The information theoretic approach is used for secrecy systems based on symmetric encryption [83] and normally only ciphertext-only attacks are considered as in these systems plaintext attacks can be prevented. However, this is different for mobile code.

In the case that decisions can be made based on the value of U, plaintext attacks can be prevented by not performing a decryption at an untrustworthy host.

However, in the case that a decision cannot be made on U, this must be decrypted at a trusted location to value Y, where $Y = F(X)$. This value of Y is sent back to the mobile code (e.g. at an untrustworthy location). It means that the malicious host has obtained a (X, Y) pair. This pair provides information on F. When the mobile code is executed at several locations, these hosts may conspire and a number of (X, Y) pairs are available. In case of sufficient (X, Y) pairs, the function F can be reconstructed. This attack shows that in particular circumstances it is possible for the hosts to obtain F without breaking the encryption algorithm. It means that in the case of mobile code, plaintext attacks cannot be prevented, and therefore, in the analysis they must be taken into account. In order to be able to use Shannon's secrecy theory it must first be extended to plaintext attacks.

2.3.4 Problems Addressed and Approach

Based on the mobile code model, this thesis will address the problem of defining the maximum level of confidentiality, such that the privacy of mobile code can be protected. Essential is that attackers are considered to have unlimited resources and time. Several concepts as perfect secrecy, cryptographic dilemma, and unicity distance will provide theoretical boundaries. These characteristics will be derived for both ciphertext-only attacks and plaintext attacks.

2.3.5 Related Work

In this model, we will make use of information theory to derive theoretical boundaries. In 1949, Shannon published an article on secrecy systems [83]. He used results of his famous paper "A Mathematical Theory of Communication"[84] to describe a model for a secrecy system and provided some measures to define the level of secrecy of

such a system. This was the start of the use of information theoretic concepts in the area of cryptography. Shannon based his work on the assumptions that the adversary has unlimited time and manpower to attack the system.

Using the same assumptions, Hellman [46] extended Shannon's approach in 1975 by introducing the concepts of spurious message and key decipherment[1], as Shannon used these concepts but did not give names to them. Based on these concepts, Hellman derived several additional theoretical bounds. Beauchemin and Brassard [6] generalized Hellman's approach in the sense that Hellman's results hold with no restrictions on the distribution of keys and messages. These information theoretic concepts are also used to derive bounds on other techniques, such as authentication [85], [61] and secret sharing [19].

Additionally, much research has been done on using information theory to derive upper and lower bounds for secrecy systems, but the assumptions are slightly different from Shannon's assumptions; this does not make them less realistic, however. Most of these approaches assume that the information available to the sender and receiver is not equal to that to which the adversary has access. For example, the concept of a noisy channel assumes that the adversary can tap the communication channel, but only with some error probability [101], [62]. A second concept is the memory-bounded adversary [16], where it is assumed that the adversary has limited memory capacity. A third primitive is the usage of a quantum channel for applications like secret key agreement [8], [14]. The secrecy of the key is guaranteed by the uncertainty relation of quantum mechanics. The assumption that the information available to the adversary and sender/receiver is not equal cannot be applied to mobile code, as the host is a possible adversary and receiving the protected code is crucial for correct execution of the code. Therefore this situation is not taken into account.

2.3.6 Discussion

This section presented the mobile code model to protect privacy. The most important aspects of the model are the fact that the host is untrustworthy and has unlimited computation power, memory resources, and time. This model and an information theoretic approach will be used to derive levels of privacy (in this case confidentiality) for both ciphertext-only and plaintext attacks.

2.4 Conclusions

This chapter presented two models as they will be used in this thesis. The first one, the agent model, will be used to derive practical solutions and is therefore based on practical assumptions. The location of the agent execution is considered to be untrustworthy; however, the attackers' capabilities are limited by time and computation power. This model does not provide insight into the maximum level of privacy that can be provided. Therefore, the second model is introduced.

The second model, the mobile code model, considers the more general concept of mobile code and is used to derive theoretical limits on providing privacy in a mobile

[1] These concepts are important to understand the meaning of the unicity distance.

code environment. The main difference between these two models is the difference in capabilities of the attacker. In contrast to the agent model, in the mobile code model, the attacker has unlimited resources and time.

Chapters 3, 4 and 5 will provide solutions to the problems described to the agent model. The chapters 6 up to 9 will consider the mobile code model.

Chapter 3

Agent Communication

Mobile agents can travel to different hosts and it is often not known beforehand where the agent will migrate to. When located at an agent platform, the agent may need to communicate with other agents. The problem of confidential communication between agents is addressed in this chapter with respect to the agent privacy model that was described in section 2.2. In this model, confidential communication is especially difficult to achieve as the agent platform cannot be considered as trustworthy.

This chapter proposes a solution for agent communication, which was also published in [21], [24].

3.1 Introduction

Privacy during communication can be seen as preventing a third party from being able to eavesdrop on the communication. In the agent privacy model, this includes the agent platform.

Many solutions have been proposed in various papers on protecting mobile software agents against untrustworthy hosts by providing confidentiality. The proposed solutions can be divided into two categories, either they protect the data the agent owns or collects, or they provide a solution towards the protection of the agent's code. Data protection is, for example, necessary when the agent collects data during the execution of its task, as this data should only be accessible to authorized entities. In [53], Karjoth et al. define a number of security properties that define the protection of data collected by an agent against an attacker. The collected data exists of a chain of encapsulated small pieces of data and these security properties can be among others: data confidentiality, forward privacy, forward integrity, or non-repudiability. A number of solutions have been proposed that offer (a subset of) these properties, see [53], [59], [103], [105]. These solutions are based on public key encryption, digital signatures, and hash chaining. In [75], flaws in some of these protocols are identified.

The second part of protection of mobile agents against untrustworthy hosts is protection of the code itself. Protection of the code means providing the code with security properties such as integrity and confidentiality. When an agent's code is executed,

there should be a guarantee that the code is executed correctly. Tools like cryptographic traces [95] and proofs of correctness [103], [9] have been proposed. These solutions are generalized in [49] by using the concept of reference states. Confidentiality of code can be achieved by using tools as code obfuscation [48]. Although in theory this is impossible [5], it may provide an adequate level of protection for a limited amount of time. Furthermore, function hiding provides a cryptographic way to achieve confidentiality for agent's code [15], [78], [57].

These solutions provide various levels of protection to an agent against a malicious host, but the subject of secure communication is not addressed. Tools like protection of computation results or function hiding can be used to prevent the malicious host from accessing the agent's content or actions. However, when the communication between agents is not protected against the host, the host may still be capable of obtaining relevant information about the agent.

Although in today's agent systems such as Jade [50], it is possible to provide confidentiality during communication, it is assumed that the host can be trusted. Hence, communication is protected against all eavesdroppers but the host. Claessens et al. [31] have also proposed a solution to provide confidential communication based on SSL/TLS, but again the host is assumed to be trustworthy.

This chapter provides a solution to the problem of confidential communication, taking an untrustworthy platform into account. The solution proposed in this chapter was first published in [21], followed by [24]. First, the problem statement is given in section 3.2, followed by a proposed solution in section 3.3. Section 3.4 provides two examples of how the solution can be used in practice. The first example shows how agents can securely collect data to set up a confidential database. This may find application in the area of privacy-sensitive databases. The second example shows how an inquiry among agents can be carried out, such that it is, for example, possible to know how many other agents are willing to communicate to an agent without actually knowing the identity of the various agents. One must think of inquiries as to how many agents are interested in a certain topic without the investigator knowing who is interested, as he may abuse that information. Finally, conclusions are given in section 3.5.

3.2 Problem Statement

Within the context of the agent privacy model, the problem is defined as providing a confidential communication between agents, where the confidential data to be communicated is stored securely in the agent. Beforehand it is not known with whom the agent will communicate. Confidential communication means that at all times, no third parties can eavesdrop successfully on the communication, not even the agent platform.

The condition that confidential data is stored securely in the agent is added, because if that is not the case, the agent platform is capable of reading data even before communication starts and securing the communication would then be an empty gesture. The condition is added explicitly to the problem statement as it is the starting position of the problem and will be of great importance to the validity of proposed solutions.

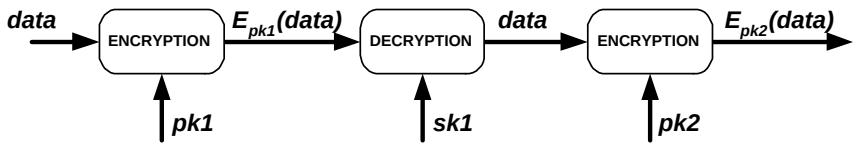

Figure 3.1: Conventional way of providing confidentiality in communication

Confidential data that is stored in a software agent is usually protected by means of encryption. Taking the untrustworthy host into account requires ensuring that at no point in time the data is available to the host in clear text. Note that within the agent privacy model, it is assumed that hosts do not conspire. Two solutions are given here that demonstrate the complexity of the problem.

A first naive approach is having the agent send the stored encrypted data directly to the communicating partner, and having it transfer in some way the key. The advantage is that at no time during communication the data appears in clear text. However, each piece of data must be encrypted separately to avoid one communicating partner from having access to more data than he is entitled to. Encrypting each block of data separately is very inefficient and impractical, as it is necessary to transmit a key for each piece of data. The key must be transmitted in a secure manner, which means that the problem has shifted from protecting the data to a key exchange problem where many keys (one for each type of data) must be transmitted confidentially. A simpler solution would be encrypting the data during the setup of the agent using the communicating partner's key. However, as in advance it is not known with whom the agent will communicate, this is not possible. Obviously, this approach to provide confidentiality during communication is not practical.

In the second approach, each piece of data is encrypted only once to keep the agent as small as possible in terms of stored data, and at the moment of communication the data is to be transformed such that only the receiving party is able to decrypt the message. Here, we describe this approach for the case where all encryptions are done using a public key algorithm. A similar solution can be described for a symmetric algorithm. The public key of the agent is used to encrypt the information, and its private key is used to decrypt it. Figure 3.1 shows the procedure for encrypting information for storage in the agent and making the data ready to be sent to the communicating partner. The first step is to encrypt the data that must be kept confidential using the agent's public key $pk1$: $E_{pk1}(data)$. This operation can be performed at the user's computer and the result can then be stored in the agent. The moment the agent needs to send this confidential data to another party, it decrypts the data using its private key $sk1$. The result of this operation is plaintext data. Then the data is again encrypted, but this time using the communicating partner's public key $pk2$: $E_{pk2}(data)$. This may also be a session key. At this point the data is ready to be sent to the communicating partner and this entity can decrypt the data, because he has access to his private key

(this last step is not shown in figure 3.1).

The advantage of this second approach is that it is very simple and efficient. All the data to be stored is encrypted with the same key, and only when needed it is transformed into an encryption with the appropriate key. It is also an advantage that beforehand it is not known to whom the agent will talk, because the encryption for the communicating partner occurs at the time of communication. A third advantage is that no complex key management scheme needs to be used, because at the moment the data is encrypted to be stored in the agent, only the agent's public key is used. Only the moment of data exchange with other parties it is necessary to obtain the communicating partner's key.

This solution would be sufficient and adequate in a scenario where the agent is in a trusted environment and where confidentiality is not a priority, but this is not the case in the agent privacy model. During the transformation from encryption with the agent's key to encryption with the communicating partner's key, the plaintext data is available to the host. Obviously, this situation should not occur. A second problem is that not only the data is readable to the host and possibly to other parties at a certain moment, but also the private key of the agent is accessible to the host during the decryption process. Consequently the host has access to all encrypted data stored in the agent. Concluding, this is only an adequate solution to provide confidential communication in a fully trusted environment.

These two approaches show why confidential data storage and communication is a complex problem. On the one hand the identity of the communicating partner is not known at initialization of the agent. On the other hand, the solution to the problem of not knowing in advance who the communicating partner will be gives the host access to the plaintext data, as the data encrypted with the agent's key is transformed at the time of communication into data encrypted with that of the communicating partner, creating a window of opportunity for the host. Therefore, a new method must be developed to provide this level of confidentiality.

We propose performing a double encryption such that at no moment in time the data is available in clear text to the host.

3.3 E-E-D: Private Agent Communication

The proposed solution to the problem of confidential data exchange is shown in figure 3.2. This solution entails that the data is first encrypted using the encryption key of the agent. At the moment that data must be exchanged to another party, the data is again encrypted, but this time with the encryption key of the communicating partner. A decryption process follows where the decryption key of the agent is used, such that the overall result is encrypted data which can only be deciphered by the communicating party. This solution will further be referred to as E-E-D, which indicates the order of encryption and decryption operations.

Usually it is necessary to first decrypt the final encryption in order to have a complete correct decryption, but here it is the opposite. This means that in our approach,

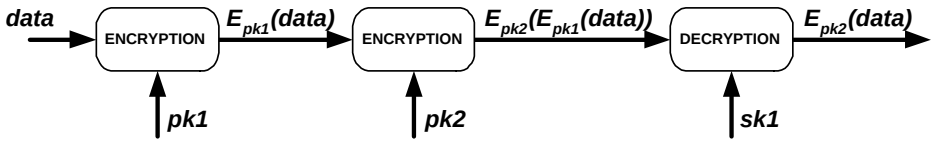

Figure 3.2: Confidentiality in agent communication.

the encryption algorithm should have the following property:

$$D_{sk1}\left(E_{pk2}\left(E_{pk1}\left(M\right)\right)\right) \;=\; E_{pk2}\left(M\right), \tag{3.1}$$

where $pk1$ and $pk2$ are the public keys of the agent and communicating party, respectively. The corresponding private keys are $sk1$ and $sk2$.

The algorithm defined by ElGamal [39] is suitable for this application. As two encryptions will be performed, two key pairs are generated. A large prime p is selected and a generator α of the multiplicative group $\mathbb{Z}_p^*$ of the integers modulo p. The private keys a_1 and a_2 are selected such that $1 \leq \; a_1, a_2 \leq p-2$. The public keys are computed as follows:

$$y_1 = \alpha^{a_1} \bmod p;\; y_2 = \alpha^{a_2} \bmod p. \tag{3.2}$$

The public keys are y_1 and y_2 with the corresponding private keys a_1 and a_2. Parameters α and p are system parameters and known by all participating parties (public parameters). The message is first encrypted using the ElGamal encryption algorithm.

First, a random integer k_1, $1 \leq k_1 \leq p-2$ is selected, and the ciphertext is computed as:

$$\gamma_1 = \alpha^{k_1} \bmod p;\; \delta_1 = m y_1^{k_1} \bmod p. \tag{3.3}$$

The ciphertext is $c_1 = (\gamma_1, \delta_1)$.

Then a second encryption is performed, using the second public key, and the message to be encrypted is now δ_1:

$$\gamma_2 = \alpha^{k_2} \bmod p;\; \delta_2 = \delta_1 y_2^{k_2} \bmod p. \tag{3.4}$$

The second ciphertext c_2 is then formed by the pair (γ_1, δ_2). Note that γ_1 is part of the ciphertext c_2, and not γ_2 as only then a correct decryption is possible. It is now possible to decrypt it once using the first private key:

$$m' = \left(\gamma_1^{-a_1}\right) \delta_2 \bmod p = m y_2^{k_2} \bmod p. \tag{3.5}$$

The result is an encryption of m based on the second public key, e.g. y_2. This can now be decrypted using the second private key:

$$m = \left(\gamma_2^{-a_2}\right) m' \bmod p. \tag{3.6}$$

Decryption of m' must be done at a different location than where the E-E-D operation took place, as decryption of m' results in plaintext. The advantage of using the E-E-D

algorithm is that it is possible to transform an encryption based on one key directly into an encryption based on a second key without decrypting it first into plaintext. This second key may be the public key of the communicating partner. Using E-E-D, one need not know when setting up the agent with whom it will communicate. Although this solution for confidential agent communication is promising, the way it is used determines the actual security. For example, if the first decryption takes place at the same location as the second encryption, the host may be capable of switching the order of operations such that the clear text can be obtained. This can also occur when hosts do conspire.

An additional property of E-E-D is the ability to add two different but similarly encrypted messages while preserving confidentiality. When it is possible to compute $E(m_1 + m_2)$ from $E(m_1)$ and $E(m_2)$ without decrypting any of these values, the encryption algorithm is denoted as being homomorphic in addition. This is possible in ElGamal given that the security parameter k is equal for $E(m_1)$ and $E(m_2)$. In that case, if two messages m_1 and m_2 are encrypted using an equal k, the ciphertexts will look as follows:

$$\gamma_1 = \alpha^k \bmod p \quad ; \quad \delta_1 = m_1 y^k \bmod p \tag{3.7}$$

$$\gamma_2 = \alpha^k \bmod p \quad ; \quad \delta_2 = m_2 y^k \bmod p, \tag{3.8}$$

where y is the public key. Note that here δ_2 is computed using the original ElGamal encryption scheme [39] and not by the E-E-D scheme. Adding δ_1 and δ_2 gives $\delta_1 + \delta_2 = (m_1 + m_2)y^k \bmod p$, which is equal to the direct encryption of m_3 where $m_3 = m_1 + m_2$, hence it is possible to add two numbers without having to decrypt one of the messages.

One of the conditions for using ElGamal is randomly choosing a new security parameter k for each encryption. Only in certain cases it is allowed to use one k twice. When one message is encrypted twice, separately, using an equal k will result in two equal ciphertexts. Furthermore, given two ciphertexts and equal values for k, it is possible to derive the ratio of the plaintexts ($\frac{\delta_1}{\delta_2} = \frac{m_1}{m_2} \bmod p$). Two parties that each encrypt a message use equal values for k and the public key. Then both parties can compute the other party's plaintext without knowing the corresponding private key. Taking these risks into account, ElGamal encryption can only be used with equal values for k when either the encrypting parties are one single entity or when the encrypting parties fully trust each other, e.g. when protection is only necessary towards a third party. The usage of E-E-D will be shown in the next section.

3.4 Example Applications

In the next two paragraphs, two examples are given of how the E-E-D algorithm can be applied. The first example is given for an agent using E-E-D and acquiring information located on untrustworthy hosts. The second example describes the usage of E-E-D with the homomorphic property in addition when an agent collects and performs additions within untrustworthy hosts. The presented examples are intended to describe possible implementation of E-E-D within a practical application. The examples do not

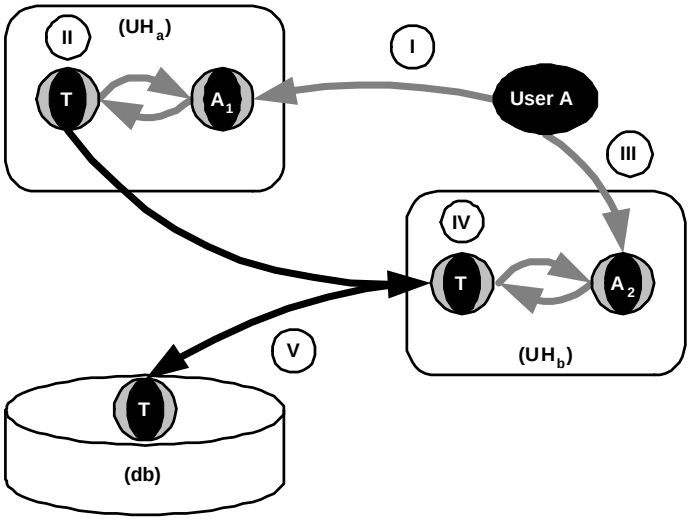

Figure 3.3: Data-collecting agent model using E-E-D

provide a full secure system and therefore should not be seen as such, e.g. they do not take into account authentication of the agents.

3.4.1 Data-Collecting Agent Model

Collecting data, in other words, acquiring information, is one of the applications mobile software agents are being used for. Figure 3.3 shows a model of an agent T acquiring information from the agents of user A located on untrustworthy hosts. The numbers in figure 3.3 indicate the order of operation.

For example, user A could be a police officer who has written a crime report on a public computer. Agent T is deployed by the police station and has the objective to collect information, like a crime report, to a secure database (db) while preserving confidentiality when transporting this information on untrustworthy hosts. This could be accomplished by E-E-D.

At number (I), user A deploys agent A_1 in possession of an encrypted message $E_{pka}(m_a)$ enciphered with the public key pka of user A. At a certain moment in time, agent T encounters agent A_1, who is willing to share his information m_a with agent T without the untrustworthy host (UH_a) acquiring any information about the content (II). Agent A_1 is not able to decipher $E_{pka}(m_a)$ because this would make the clear-text message m_a visible on the untrustworthy host. Therefore, agent T wants to transform this encrypted message m_a, and encrypts it with his public key pkt resulting in $E_{pkt}(E_{pka}(m_a))$.

Additionally, user A deploys a second agent, A_2, in possession of the private key ska, dispatching it to a different location (host) than agent A_1, namely, (III). It is presumed that various untrustworthy hosts do not conspire with one another, which

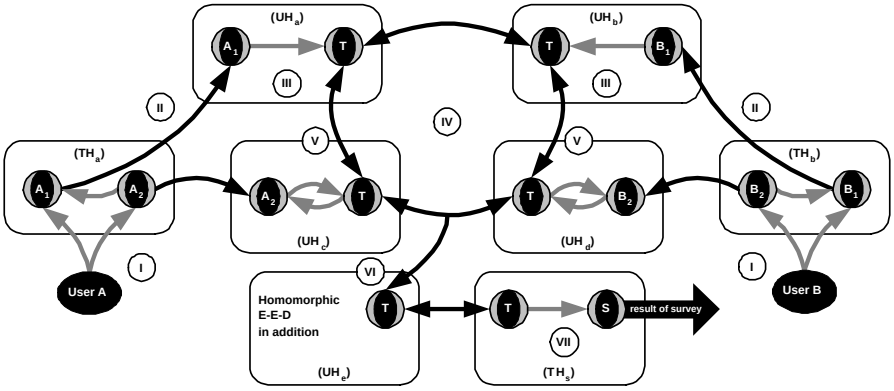

Figure 3.4: Survey model with multi-agents using homomorphic E-E-D in addition within untrusted environments

implies that the single encrypted data $E_{pka}(m_a)$ should never be on the same host at the same time as the private key ska. This is the reason why agent T does not move to another platform before protecting the data with his public key pkt, as was described above. Note that storing a private key in an agent must be done carefully, but this key ska is only used for decryption of data that is still encrypted after the first decryption. In order to have a secure system this ska must be changed regularly.

When agent T exchanges information with agent A_1, he also receives the location of agent A_2, who is in possession of the private key ska. Moving to the assigned location (IV), agent T sends his double-encrypted data to agent A_2, who is capable of deciphering the first encryption, $D_{ska}(E_{pkt}(E_{pka}(m_a)))$. This computation will result in a single encryption of the message m_a with the public key pkt, $E_{pkt}(m_a)$.

After collecting numerous pieces of data, agent T is able to store this data in a database (db) (V). Database (db) saves all the encrypted information using the public key of agent T. If access is needed to this information, agent T must only use his private key skt without first consulting other agents about their private key.

This example creates a uniform encrypted database, where all data is encrypted with the same public key of agent T even if this data was collected on untrustworthy hosts. For the police case, the police officer can be confident that no information has leaked to an untrustworthy host and that only the police station is able to read the content of the database (db) and therefore his crime report.

3.4.2 Survey Model with Multi-Agents

In figure 3.4, the survey model is given of an agent S sending his transport agent T around an untrustworthy network to survey for agents willing to communicate with him. The numbers indicate the order of operation. In this case, the requirements are to provide confidentiality when the data is collected and the ability to perform an operation on this collected data. Homomorphic E-E-D in addition, as was described

at the end of section 3.3, fulfills these demands.

This survey model can also be used as a building block for other applications like voting, e-commerce (market place) or polls. For example, a police officer deploys an agent S who looks for some information about a certain criminal. Agent S queries the network for agents willing to share information with him by deploying an transport agent T. To count the number of agents willing to communicate with agent T, the homomorphic property of E-E-D is used without compromising confidentiality.

At number (I), user A deploys an agent A_1 within a trusted host (TH_a) in possession of an encrypted message $E_{pka}(m_a)$ encrypted with the public key pka of user A. This message could contain a certain preference of the user and therefore it should only be accessible to authorized agents, for example agent S. Because user A does not know with whom he will communicate, he moves agent A_1 to a untrustworthy host (UH_a) (II) and waits for the appropriate agent to arrive. The user also deploys an agent A_2 which owns the users secret key ska and sends this agent to a different untrustworthy host (UH_c). Before both agents leave their trusted environments, agent A_2 gives his destination location to agent A_1.

The transport agent T moves around various platforms until he finds a suitable agent to communicate with. Agent T and agent A_1 meet each other at UH_a (III). After authentication, agent A_1 sends his encrypted message $E_{pka}(m_a)$ and the location of agent A_2 to agent T. Before moving to the location of agent A_1 (IV), agent T first encrypts the received data with his public key pkt, which results in $E_{pkt}(E_{pka}(m_a))$.

Agent A_2 and agent T meet each other at UH_c (V). After receiving the data of agent T, agent A_2 decrypts the first encryption using the private key of user A. It is not a problem that the secret key to the untrustworthy host is exposed because we consider the host to be curious and not conspiring with other untrustworthy hosts. The result of the decryption, $E_{pkt}(m_a)$, is stored at agent T.

When moving around the network, agent T identifies another agent with a similar interest, agent B_1 (owned by user B). Before moving to an arbitrary host, agent T carries out a computation on message m_b comparable to that on message m_a as mentioned above, where m_b is provided by user B.

Agent S is interested in the number of agents that want to communicate with him, so messages m_a and m_b represent the willingness to communicate. If the message is equal to $'1'$ it stands for *willing to communicate*; $'0'$ stands for *unwilling to communicate*. To compute the total amount of agents willing to communicate, agent T must simply add the collected messages together. Because the messages are uniformly encrypted, which means that they are encrypted with the same key of agent T, it is possible to use the homomorphic property of E-E-D, resulting in $E_{pkt}(m_a + m_b)$ (VI).

To obtain the result of the survey, agent T moves to his originator, agent S, who is in possession of the secret key skt (VII). Agent S is located on a trusted host (TH_s), where he is allowed to use this private key without compromising confidentiality of the obtained data.

In this example, agent T collects information and transforms the encrypted information into encrypted data based on T's public key. After this transformation, all data is encrypted based on the same key and security parameter k. Because no other parties are involved in this process, it is not a problem if the same value is used for k, as is

described in section 3.3.

This survey model gives an example of how E-E-D with the homomorphic property can be used to add confidential data together within untrustworthy environments. Using this model, the police officer is now able to communicate with users through agents without losing confidentiality.

3.5 Conclusions

This chapter addresses the problem of secure agent communication within the context of the agent privacy model. When agents need to communicate with each other and the host is not entitled to have access to this communication, then conventional mechanisms do not provide an adequate level of security. Therefore, the concept of double encryptions was proposed and it was shown how this concept can be applied by using agents.

First, the results of the E-E-D algorithm were analyzed, followed by the meaning of these results in the context of the agent privacy model.

The E-E-D algorithm provides a method to transform an encrypted message using one key into a message encrypted with another key, while during transformation, the message is never available in clear text. With E-E-D, data can be exchanged securely without the platform having access to the clear text data. Furthermore, the E-E-D algorithm removes the need to know beforehand with whom the agent will communicate as the key for communication must only be decided upon the moment of communication.

However, with respect to the original problem statement of providing confidential communication, the E-E-D algorithm does not fulfil all requirements. Using E-E-D, it is possible to provide secure data exchange. However, communication is in two directions. When receiving a message, a communicating partner should be able to respond to this message. This is only possible if it understands and can interpret the received message. When the E-E-D algorithm is used, the agent receives an encrypted message. Processing the data usually requires decryption, but that would again give the platform access to the clear text data. Therefore, a fully secure communication can only be achieved when this processing of received data occurs based on encrypted data. The additional advantage of the homomorphic property of E-E-D shows that some progress can be made in this direction, as was shown in the second example.

Concluding, for the agent privacy model it was shown that data can be exchanged securely can be achieved by using the E-E-D algorithm, and a start was made towards secure interaction to provide confidentiality for full communication. However, full communication requires privacy during processing of data, and this will be covered in the next chapter of execution privacy.

Chapter 4

Execution Privacy

The agent privacy model was used to describe to the problem of providing privacy during the execution of the agent. The host is capable of observing the agent's actions and by doing this may compromise the user's privacy. This chapter covers this problem and proposes a solution using the approach of encrypting an agent's task. The results have been described in [25].

4.1 Introduction

The objective of the agent privacy model is to provide privacy to the user by providing privacy to his mobile software agent. The possibility that the host may eavesdrop on all actions of the agent is a serious threat. During execution of its tasks, the host is able to observe the agent and therefore to obtain knowledge about its content. Hence it is necessary to provide some privacy protection mechanism for the agent's actions.

Furthermore, a result of the previous chapter on agent communication is that the privacy of full interactive communication can only be protected when the agent is capable of interpreting received protected data without performing a decryption. This interpretation is also a type of action the agent must perform and therefore a privacy protection tool is required to protect the privacy during an interactive communication. The privacy protection of agent's tasks or actions is here called execution privacy as it covers all types of functions, actions or tasks that must be executed. The main difference between data privacy and execution privacy is that data is static and does not change over time. However, in execution privacy, data must be protected while it is being processed.

More generally, the problem can be seen as protecting mobile software. Sander and Tschudin [78] described a method to provide confidentiality to functions in the form of a polynomial. In a later paper, they extended the results to all functions computable by circuits of logarithmic depth [76] and further generalized them to arbitrary functions, provided these can be represented by a polynomial-sized circuit [15]. Already in 1990, Abadi and Feigenbaum [1] described a method that provided confidentiality for circuit evaluation. A disadvantage of this method is that it takes many

interactions to provide the confidentiality. Many other solutions have been published to provide secure circuit evaluation, but none of them is very practical or efficient [1]. Several more practical methods have been proposed, but they are all based on either trusted hardware located at the host [103] or on the presence of a Trusted Third Party [99], [3].

Algesheimer et al. state in [3] that according for their model non-interactive mobile computing schemes do not exist. It is stated that any scheme in which some host is to learn information that depends on the agent's current state cannot be secure. What is meant is that it is possible to secure the evaluation of the agent's state but impossible to secure the output from the agent to the host if it depends on more than its own input. A TTP is suggested where computations are done, but in such a way that even the trusted party does not learn anything substantial about the task. However, the underlying assumption is that the hosts do not conspire with the trusted party. Obviously this is a strong assumption; it would be preferable to have a system where only the originator must be trusted and still the agent has guaranteed security and privacy. Summarizing, no satisfying solution has been found to the security and privacy problems that does not decrease the autonomy of the agent. Either the system does not provide protection against multiple malicious parties, or it is limited in autonomy such that decisions to determine the agent's next action should be made at a trusted site.

In this chapter, the approach described by Sander and Tschudin is adopted. First, the problem statement for this chapter is given. This is followed by a description of how polynomials can be encrypted using a simple encryption algorithm like ElGamal. Several constraints must be set for the usage of ElGamal; these are also explained in section 4.3.1. Conclusions are given in section 4.5.

4.2 Problem Statement

In the previous section, the term execution privacy was used in the context of providing privacy during the agent's actions. It is important that actions are dynamic; to provide confidentiality to these actions, they must be executed in an encrypted form.

The term 'execution privacy' consists of the two elements that define an agent's action, namely functions and decisions, i.e. the interpretation of the result of a function. When one agent needs to perform an action, one or more functions are executed. Based on the result, the agent must interpret the result and decide on the next action. A simple form of interpretation may be an if/else statement.

The problem statement for this chapter is then defined; how can execution privacy be guaranteed within the context of the agent privacy model.

The approach to provide execution privacy is based on the following requirements for execution. The first is that it should not be possible for the attacker (host) to obtain knowledge on the function that is executed, besides knowledge on one input and corresponding output. Knowledge on one input-output pair cannot be kept from the attacker as he provides the input and can observe the output.

The second requirement is that the attacker should not be capable of deriving an

optimal input given a certain end result. For example, the agent executes the task of purchasing a flight ticket at a foreign host (in the flight-ticket case, an airline company). The function in the task requires input from the host, such as price, departure time, available seats, etc... Obviously the host would be interested in the maximum price at which the agent will still buy the flight ticket. In this case, the host is not interested in obtaining the exact content of the agent's task, but only in determining what he should offer the agent such that the outcome is to his advantage.

A third requirement is that the decisions on the interpretation or the results should be hidden from or be not clear to the attacker. Based on the agent's decision criteria, the attacker could be capable of obtaining knowledge of its content or the optimal input.

Based on these requirements we chose the approach of function hiding. Function hiding provides confidentiality towards the function such as is required by the first requirement. This approach can be extended such that also the second and third requirement are fulfilled.

To develop a solution for the execution privacy problem, we confine ourself to the following case, because of mathematical tractability. For the function hiding approach, an agent's task consists of only one function. The results can easily be extended towards multiple functions. Furthermore, it is assumed that the function is written in the form of a polynomial. This assumption is made because of the homomorphic properties that are present in certain encryption algorithms. Finally, it is assumed that each function is only executed once at a particular host. This assumption is necessary to prevent an attack where the attacker executes the protected code many times and observes the agent's behavior, as this would allow him to determine the agent's content.

The next section describes a privacy solution to the execution privacy problem, given the extra assumptions with respect to the agent privacy model.

4.3 Execution Privacy Solution

4.3.1 Function Protection

This section shows how functions in the form of polynomials can be encrypted. The approach used is based on work in [78]. Polynomials only exist of two types of operands, namely addition and multiplication. When an encryption algorithm has the characteristic of being homomorphic in addition and multiplication, it can encrypt polynomials and execute the encrypted polynomial. The result can easily be decrypted.

An encryption algorithm is said to be homomorphic in addition when it is possible to compute $E(x + y)$ from $E(x)$ and $E(y)$ without decrypting $E(x)$ and $E(y)$. Similarly, an encryption algorithm is said to be homomorphic in multiplication if it is possible to compute $E(xy)$ from $E(x)$ and $E(y)$ without decryption of $E(x)$ and $E(y)$.

The public key encryption algorithm designed by ElGamal is suitable for this application as it is homomorphic in multiplication. However, in general, ElGamal is not homomorphic in addition; only if the security parameter k for each encryption of x and y is equal. In general, this is a bad idea; as knowledge of the security parameter makes it possible to decrypt all ciphertexts generated using this k-value (see section 3.3).

First, the algorithm will be given with which ElGamal can be used to encrypt functions. Then, section 4.4 will show how the security parameter, k, should be used.

Key generation. The generation of the key is equal to the original ElGamal encryption scheme and is performed by the agent's user at a trusted location. The public key is y and the private key is a, where the relation is $y = \alpha^a \bmod p$. Parameters p and α are system parameters and are public.

Encryption. Let the function to be encrypted be a polynomial of degree n: $f(x) = \sum_{i=0}^{n} c_i x^i \bmod p$. The user shall select at random a parameter k_1, $0 \leq k_1 \leq p-2$ and encrypt each of the coefficients separately:

$$E(c_i) = (c_i y^{k_1} \bmod p, \alpha^{k_1} \bmod p); \text{ and } \tilde{c}_i = c_i y^{k_1} \bmod p. \tag{4.1}$$

In the agent, the following function is stored:

$$\tilde{f}(\tilde{x}) = \sum_{i=0}^{n} \tilde{c}_i \tilde{x}^i \bmod p, \tag{4.2}$$

where the host or another agent delivers the encrypted input $\tilde{x}$ to the function $\tilde{f}(.)$.

Encrypted function execution. The host or other agent selects at random a parameter $k_2, 0 \leq k_2 \leq p-2$ and encrypts its input x and its powers:

$$E(x^i) = (x^i y^{k_2} \bmod p; \alpha^{k_2} \bmod p); \tag{4.3}$$

$$\tilde{x}^i = x^i y^{k_2} \bmod p \text{ for } i = 0, \ldots n. \tag{4.4}$$

The encrypted function is $\tilde{f}(\tilde{x}) = \sum_{i=0}^{n} \tilde{c}_i \tilde{x}^i \bmod p$ and is executed at the host. The result of $\tilde{f}(\tilde{x}), \tilde{y}$ is then used to make a decision on the consequence of the executed task.

Decryption The user is the only one able to decrypt function $\tilde{f}(\tilde{x})$ and $\tilde{y}$ as he only possesses the private key a. This is done by computing:

$$D(\tilde{f}(\tilde{x})) = \tilde{f}(\tilde{x}) \alpha^{-(k_1+k_2)a} \bmod p. \tag{4.5}$$

It is easy to show that decryption works:

$$D(\tilde{f}(\tilde{x})) = \left[\sum_{i=0}^{n} c_i x^i y^{k_1+k_2} \right] \alpha^{-(k_1+k_2)a} \bmod p$$

$$= y^{k_1+k_2}\alpha^{-(k_1+k_2)a}\sum_{i=0}^{n} c_i x^i \bmod p$$

$$= \sum_{i=0}^{n} c_i x^i \bmod p = f(x).$$

The advantage of using function encryption is that during execution none of the parameters are at one point in clear text.

Section 3.3 explains how the security parameter k should be used. Here the different ks are used according to these conditions. In the evaluation, it will be shown what information is revealed by using the security parameter in this way.

Task confidentiality is not only provided to hide the function, but also to prevent the host or agent platform from computing an optimal input. This is the reason why the input x must be encrypted. If x were not encrypted, it would be possible for the host to compute the optimal x value by inverting the function. In case x is encrypted with the user's public key, the host can compute an $\tilde{x}$ value that corresponds to the output value of the function that is optimal for him. Because the encrypted function takes an encrypted value as input, it is for the attacker infeasible to compute x from a preferred $\tilde{x}$ as that would mean he must decrypt $\tilde{x}$, but he has no access to the correct private key.

4.3.2 Decision and Interpretation of Encrypted Data

A task consists of one or multiple functions, and when these functions have been executed, typically an encrypted value is the outcome. The question is what the agent can do with this value. Based on this value, the agent has to know how to proceed to fulfill this objective. Therefore it must be able to reason based on the encrypted values.

Consider again the flight-ticket example. The agent asks the airline for a price offer (e.g. parameter x in the function $f(x)$) or an encrypted price offer (e.g. parameter $\tilde{x}$ in the function $\tilde{f}(\tilde{x})$). The function $f(.)$ consists of coefficients that provide weighting factors to the input. When the encrypted function $\tilde{f}(\tilde{x})$ is executed, the outcome is an encrypted value. Based on this value, the next action of the agent must be determined. If functions and parameters were not encrypted, the outcome would be a clear text value and correct interpretation would lead to the next action of the agent. For example, when the output is lower than a certain threshold value, the flight ticket will be purchased; but not otherwise.

Reasoning based on encrypted data is a rather complicated problem as it also contradicts the objective of encryption. Encryption is meant to be such that an adversary cannot obtain any information regarding the clear text, but reasoning requires some knowledge based on which a decision can be made. Here, we describe a conceptual but impractical solution.

A solution is storing all possible outcomes in an encrypted format linked to the decision. This is possible in some cases (when the output set has a limited number

of elements) but always inefficient, and if this solution were used, some of the nice properties of agent technology would be lost to us.

Given the following sequence of actions that forms a task when an agent is located at a host :

$$\begin{aligned} y &= f(x) \\ d &= B(y)\,; d \in \{0,1\}, \end{aligned}$$

where x is the input given by the host and $B(y)$ describes the decision process, the outcome of the decision is $d \in \{0,1\}$. The agent's owner encrypts $f(.)$ using his own public key pko:

$$\tilde{f}(.) = E_{pko}(f(.)) \tag{4.6}$$

Assume the following for the decision algorithm $B(y)$. The decision space is formed by a set S and its complement SN. For example, set S is defined as:

$$S : \{y_j | B(y_j) =' 0'\} \tag{4.7}$$

The protection of the decision algorithm $B(.)$ is defined as the encryption of each element of S or on the complement of S (depending on the length of the set):

$$\tilde{S} := \{\tilde{y_j} | \tilde{y_j} = E_{pko}(y_j)\}, \tag{4.8}$$

using the originators's public key, pko. Both $\tilde{f}(.)$ and $\tilde{S}$ together with the originator's public key pko are stored in the agent. When the agent is located at a host, the encrypted value of the host's input, x, is computed:

$$\tilde{x} = E_{pko}(x). \tag{4.9}$$

This is used as the input for $\tilde{f}(.)$:

$$\tilde{y} = \tilde{f}(\tilde{x}). \tag{4.10}$$

The decision algorithm $\tilde{B}(.)$ now consists of comparing the elements of $\tilde{S}$ to $\tilde{y}$. If a comparison is found, the decision is '0', which can be equal to the purchase of a flight ticket. This approach can easily be extended to multiple decision values.

It is easily seen that this solution combined with the function encryption of the previous section provides confidentiality to an agent's task under the assumption that the encryption algorithm used is secure. When the agent is located at the host, only one encryption takes place and no decryptions, which provides the privacy of the task. Because a decision is made based on comparison of the encrypted values, the host cannot gain any knowledge about the original intervals and therefore cannot determine its optimal offer.

The above solution does provide privacy and security, but it is not practical. The number of encryptions to determine the set $\tilde{S}$ is linear in the cardinality of set S. The maximum number of encryptions is therefore $|S|/2$. This solution is only practical

in case where only a strongly limited amount of elements leads to one decision. The time-consuming work may also be worthwhile if the task is highly confidential.

To decrease the number of encryptions needed to reason on encrypted data, one encryption computation could be used for multiple elements of the set S. For example, a subset of S is combined and encrypted in one go (e.g. $\tilde{s}_{1j} = E_{pko}(y_1||y_2||\ldots||y_j)$). This is more efficient as less encryptions must be computed beforehand, but a new problem is created. The question has now how become how we can check whether $\tilde{y}$ is an element of $\tilde{S}$. In general, the following problem must be solved. Given a set A and the encryption of a subset of A:

$$E(A) = E(a_0||a_1||\ldots||a_n), \tag{4.11}$$

and $E(b)$, find a method that makes it possible to check whether $b \in A$ without decrypting $E(A)$ or $E(b)$. This is a problem to which no answer has been found yet.

From the above description, it is clear that reasoning on encrypted data is difficult, as a simple check whether a value belongs to a certain set already requires too many encryption operations in advance.

Obviously, it would be possible to let the agent return to the user, where $\tilde{y}$ can be decrypted such that a decision can be made easily. However, this solution would restrict the desirable autonomy characteristic of an agent to be autonomous.

Especially this concept of reasoning on encrypted data is an interesting area that requires much more research.

4.4 Evaluation

The security of function encryption using ElGamal is largely based on the correct usage of security parameter k. Chapter 3 described how parameter k should be used. In the case of function protection, two different k- values are used, k_1 and k_2. Each of them is used by just one party. This was the condition given in chapter 3. However, it must be noted that encrypting two messages using the same k-value provides knowledge on the ratio between these messages. In the case of polynomial encryption, this has the following consequence. It is possible to write each encrypted coefficient in terms of c_0 and $\tilde{c}_0$, e.g. $c_i = \frac{c_0 \tilde{c}_i}{\tilde{c}_0}$. Hence, the security level depends on the security of one encrypted coefficient.

The solution proposed in this chapter has the following properties (based on the requirements). First, using the ElGamal encryption scheme, it is possible to store encrypted polynomials in an agent and these functions can be executed in encrypted form. The level of security that is provided depends on the security of the ElGamal encryption scheme. Using the approach of function encryption, the first requirement is fulfilled of providing confidentiality towards functions.

Second, by encrypting the input x, the method ensures that a host cannot determine an optimal input given an $\tilde{y}$. This is because x is encrypted using a public key, and the corresponding private key is not owned by the host. For an attacker it is possible to compute $\tilde{x}$ for a given $\tilde{y}$. When the attacker knows the best value for $\tilde{y}$, he can

compute the corresponding $\tilde{x}$, but because a decryption is need to compute x given $\tilde{x}$, it is not possible to compute an optimal x value.

With the solution proposed here, it is not possible to decrypt $\tilde{y}$ and determine the agent's next action based on this decrypted value, because then require the agent would have to carry a corresponding private key. This would allow the host to observe this private key, and as it can also be used to decrypt the encrypted function, the agent may not carry it. Therefore, it is necessary to make a decision based on the encrypted output $\tilde{y}$. A method was given, but it is inefficient in practice.

Providing privacy for an interactive communication requires some protected processing techniques. When the interpretation on processing or received messages can be described as polynomials encryption of functions can be used for this processing. In that case, privacy protection is provided for two-way communication.

4.5 Conclusions

Based on the evaluation of the proposed solution, we can conclude that function encryption in combination with decision making on encrypted data provides an adequate level of execution privacy. However, because only polynomials can be used and the decision making is inefficient, the practical possibilities are limited.

The limitations of this solution yield an interesting research question; to analyze the theoretical boundaries of execution privacy. With these theoretical boundaries, it may be possible to extend the approach that was discussed here of protecting functions. These theoretical boundaries are derived in chapters 6 - 9.

Chapter 5

Agent Digital Signature

This chapter addresses the problem of providing integrity and source authentication by introducing an agent digital signature. If an agent must sign a document at a foreign host, the host must not be able to obtain the agent's private key. The private key can be seen as the only information that can prove the identity of the agent; hence this information should not be revealed to anyone. This chapter gives several solutions to how an agent can sign a document without giving its private key to the host. Double signing is a problem here, and also for this, some solutions are given that makes it less attractive for the host to sign documents under the agent's name without permission. The solution can be seen as a new privacy enhancing technology for agent applications. The results of this chapter were published in [20].

5.1 Introduction

In the agent privacy model, one of the problems is to provide integrity and authentication to agents while providing privacy. Consider the flight ticket example, where the agent searches for a ticket based on conditions set by its user. When the agent decides to purchase the ticket, a digital signature is required. In the agent privacy model, the agent operates in a possibly untrustworthy environment. When the agent needs to sign a document, its private key is used, which means that the agent platform may obtain this key. At a later stage, the platform is capable of signing documents posing as the agent signed them. Obviously, this is a serious privacy threat.

This chapter describes a signature mechanism that can be applied within the context of the agent privacy model.

This chapter is organized as follows. Section 5.2 gives the problem statement and related work. In section 5.3 the solution outline is presented, followed by the detailed solution in section 5.4. Section 5.5 gives an evaluation on the proposed solution.

5.2 Problem Statement

The following problem is addressed in this chapter. How can a mobile software agent sign a document while located at a possibly untrustworthy host within the context of the agent privacy model? The solution to this problem should be such that both integrity and authentication can be provided.

The main problem requiring a solution is the fact that an ordinary signature requires an operation performed using the private key. Within the agent privacy model, it is essential to prevent the platform from having access to personal data and this includes the private key.

Therefore the requirements are to design a digital signature such that it has all properties of an ordinary signature and the untrustworthy host is not capable of pretending to be the agent. The latter is explicitly necessary because in contrast to conventional digital signatures, where it is possible to hide the private key for anyone, in the agent privacy model, this is not true.

Different approaches to provide a digital signature mechanism to mobile software agents have been proposed. In [73], Ramao proposes to use proxy certificates. The user has a private key for signing documents, but the key is not given to the agent. Instead, the agent receives a new key pair. This key pair is certified by the user. The lifetime of this certificate is short. It is based on the idea that it should be difficult for an attacker to discover the private key before the certificate expires. Because the key pair is certified by the user, the agent signature is linked to the user. This approach prevents malicious usage of the user's private key, but the host has access to the agent's private key and can therefore sign a document using this key and that compromises privacy.

In [104], Yi et al. present a digital signature scheme where each time a message must be signed, a different key pair is used. This key pair is certified by using a long-term key pair. The fact that the keys used to sign messages are message related makes this scheme less practical for the application of mobile software agents, where beforehand it is not always known what messages must be signed.

Furthermore, the concept of forward digital signatures can be used in an agent environment [7], [55]. In a forward signature scheme, the private key can be updated over time, while the public key remains the same. This update is done in such a way that once a private key is compromised, no signatures based on the previous private keys can be generated, therefore the signatures in the past remain valid. This approach has the advantage that once a malicious host has access to a private key, the past signatures still remain uncompromised. However, the host is capable of observing the private key used at that moment and of using it to sign documents in the agent's name; it may take time before it is noticed and detection afterwards may be too late.

In Sander and Tschudin's original paper on mobile cryptography [78], the concept of undetachable signatures is introduced. The idea is that the agent contains two functions: the function it must execute and a function that has the property that when it is executed, the result is a signature on the outcome of the first function. Hence, the signature is attached to the function to be executed. Kotzanikolaou et al. [54] presents

such an undetachable signature based on RSA. This is further extended towards undetachable threshold signatures by Borselius et al. [12], where multiple agents are required to generate one signature.

Around the same time that the results of this chapter were published, Ferreira and Dahab presented a solution to the agent signature problem in [41] based on the same solution as this chapter, namely the approach of blinding the private key. Although the approach is equal, the method of blinding is different, and in [41] the problem of double signing is addressed by using a notary that owns a policy that restricts the usage of the key.

5.3 Solution Outline

The approach to designing an agent digital signature is to hide the agent's private key such that the agent can sign a document with this hidden key. Knowledge of the hidden or blinded key should not lead to the original private key. The signature algorithms presented in the next section are based on the blind digital signature first introduced by Chaum [29] and later extended to different algorithms [18] or with additional properties [2] [87]. Blind digital signatures have the property that a message can be signed by a party without that party being able to read the message, e.g the message is blinded before signing. These signatures can be applied in applications like untraceable payments [29], [13] or electronic voting, where privacy of an individual is important.

The idea of blinding a parameter before signing is here applied to the private key of the agent.

This signature will then consist of the following steps:

1. Key generation
2. Blinding operation on private key
3. Signature operation
4. Activation of signature
5. Verification

Steps 1, 3 and 5 are necessary in any conventional digital signature algorithm. Step 2 is the transformation of a private key into a blinded version and in step 4, the signature is activated. This step is necessary because in step 3, a signature is set using a blinded private key. This signature cannot yet be verified by using the agent's public key, because the blinded private key and the agent's public key are not related as such. Hence an activation procedure must be added.

Steps 1 and 2 must be performed in a trusted environment, e.g. the user's computer. Step 3 and 4 are done at the foreign host. Finally the verification can be done anywhere in the system.

5.4 Agent Digital Signature

5.4.1 Introduction

The various digital signatures presented in this chapter are based on the elliptic curve digital signature algorithm [52][1]. In this section a small modification, similar to the one in [18] is done in order transform the digital signature function into one where a parameter is private. The following sections propose new signature algorithms.

The security in elliptic curve cryptographic systems is based on the hardness of the discrete logarithm problem for elliptic curves. This problem can be informally described as follows [81]: Given two points P and Q on an elliptic curve such that $Q = dP$, find the integer d. This problem is generally believed to be infeasible if the space in which d is chosen is large enough. Certain curves are believed not to be secure, such as supersingular curves [66]. To provide clarity in notation, for a point on the curve, a capital letter is used, and integers are shown in lower-case letters.

In a conventional system, the digital signature consists of three steps: key generation, signature operation and signature verification (steps 1,3 and 5 in section 2). The signer owns two kinds of keys: the private and public key. The private key, which is only known to the signer, is used to sign a digital message, and the public key is used to verify the validity of the signature [81]. In practice, the public key is certified by a Trusted Third Party (TTP), such that the public key is connected to an identity. Hence, the verification of the signature using the public key can prove who signed the document and this entity cannot deny its signature afterwards. This property provides non-repudiation.

1. **Key generation**
   The key generation starts with the signer choosing an elliptic curve E defined over Z_p. The number of points on $E(Z_p)$ should be divisible by a large prime n. The signer selects a point P on $E(Z_p)$ of order n and selects a random integer d in the interval $[1, n-1]$. Parameter d is the private key of the signer and must be kept secret. Using the private key, the signer can compute its public key:

   $$Q = dP \text{ over } E(Z_p). \tag{5.1}$$

   The public key is Q, and E, P, and n are system parameters.

2. **Signature generation**
   In order to sign a message m, the signer selects a random secret integer $k \in [1, n-1]$ and computes:

   $$R = kP = (x_0, y_0), \tag{5.2}$$
   $$r = x_0 \bmod n,$$
   $$s = km + rd \bmod n. \tag{5.3}$$

   The signature consists of the parameters (r, s) and is sent to the verifier in combination with the message m. Parameter k must be different for each signature

[1] Considering the content of this thesis, it would have been more consistent to base the proposed signature scheme on the ElGamal signature algorithm. This would be possible.

based on one private key d. In case k is equal for two different messages, d can easily be computed [39].

3. **Verification**
   By obtaining the right signature parameters, public key and digital signature, the verifier can check the validity of the signature by computing:

$$\begin{aligned} T &= (sP - rQ)m^{-1} = (x_1, y_1) \\ t &= x_1 \bmod n \end{aligned} \tag{5.4}$$

   The signature is valid if and only if:

$$t = r.$$

5.4.2 Agent Digital Signature

In case a software agent needs to sign a document, it cannot follow the above procedure, because there are some fundamental differences:

- The agent is the signer of the document but does not own the resources to be able to compute the signature.
- The signer does not know whether it is located in a trusted environment.

If the agent lets the host compute a signature for it, using the algorithm described in 3.1, the host will have access to the agent's private key d and allowing it to sign other messages in the agent's name or to pretend it is the agent (during authentication). In this case, the property of non-repudiation is no longer present, because multiple entities now have access to the agent's private key, hence there is no guarantee that it is the agent who signed the document.

As is described in the solution outline, the idea is to sign a document using the host's resources by using an agent's hidden private key. After the signature is computed, the host activates the signature. The user generates several blinding factors, which hide the private key. These factors are then needed to activate the signature. This can be accomplished by storing one part of the blinding factors in the agent and the other securely at the user's computer. How this can be achieved such that the activation can take place at the host and the private key cannot be computed by the host is shown in the description of the algorithm.

1. **Key generation**
   An elliptic curve is defined over Z_p, of which the number of points on $E(Z_p)$ is divisible by a large prime n. The user selects a random integer d in the interval $[1, n-1]$. Parameter d is the agent's private key and is securely stored at the user's computer. The user computes the agent's public key:

$$Q = dP \text{ over } E(Z_p). \tag{5.5}$$

The public key is Q, and E, P, and n are system parameters. These are stored in the agent and at the user's computer. Besides a regular public key, the user computes a "temporary"public key:

$$\delta = d\gamma \bmod n, \tag{5.6}$$
$$\Gamma = \delta P, \text{ over } E(Z_p). \tag{5.7}$$

and here, γ is a blinding factor and δ can be seen as a temporary private key. The parameters γ and d are stored securely at the user and are not given to any other element in the system. Parameter Γ is also stored in the agent for verification purposes.

Two extra parameters, α and λ, are chosen at random by the user in the interval $[1, n-1]$ and the following is computed:

$$c = \alpha\gamma \bmod n, \tag{5.8}$$
$$\Lambda = \lambda cP \text{ over } E(Z_p). \tag{5.9}$$

In the next steps, it will become clear why these parameters are necessary. The parameters Λ and c are stored in the agent, while α, γ and λ are kept secret at the user's computer.

2. **Blinding operation on private key**
This step, the hiding of the private key, is completed at the user's computer. In order to obtain a blinded private key, the user selects one other blinding factor β at random in the interval $[1, n-1]$. As in the digital signature algorithm, the user also selects a parameter k at random in $[1, n-1]$ and computes:

$$\tilde{R} = kP = (x_0, y_0), \tag{5.10}$$
$$\tilde{r} = x_0 \bmod n,$$
$$R = c\tilde{R} + \beta P = (x_1, y_1,) \tag{5.11}$$
$$r = x_1 \bmod n,$$
$$\tilde{d} = \alpha^{-1}d + \lambda \bmod n, \tag{5.12}$$

in which $\tilde{d}$ is the blinded private key. The blinding factors α, λ and γ must be kept secret at the user's computer, just like the agent's private key. Parameters $\tilde{d}, k, r, \beta$ and c are stored in the agent and therefore known to the host. Parameters $\tilde{R}$ and R are not necessary anymore for the remainder of the algorithm. These parameters in combination with the system parameters give the agent the opportunity to sign a document m while located at a foreign host and using its computational resources without giving this host access to its private key. Also, knowing these parameters, the host cannot compute d, as the parameters α and λ are not stored in the agent. Parameter α can only be obtained through c, but that requires knowledge of γ which is not available in the agent.

3. **Signature generation**
During the signature generation, the agent is located at the host. The signature

on message m is then computed by:

$$\tilde{s} = km + r\tilde{d} \bmod n. \tag{5.13}$$

It can be seen in equation (5.13) that the signature operation the host must execute for the agent is equal to that in equation (5.3). The only difference is that in (5.13), the hidden private key is used instead of the original private key.

In section 3.1 it was said that parameter k must be kept secret in order to prevent the disclosure of the private key. Here, this would mean that the host either may not have access to it or the user must trust the host not to abuse this knowledge. Fortunately, this does not matter here, because the private key is not used. By using the same k twice, the host would not gain any more knowledge about the private key.

4. **Activation of signature**
Because some of the blinding factors are stored in the agent, the host is able to transform the signature towards a valid signature. By valid we mean, that the signature must be verified using the agent's public key as registered at a Trusted Third Party (if a PKI is used). The signature can be activated by computing:

$$s = c\tilde{s} + \beta m \bmod n. \tag{5.14}$$

Parameters c and β are known by the host, but this is not sufficient to compute d or γ. Substitution of parameters gives the following signature:

$$s = (\alpha\gamma k + \beta)m + \gamma r d + \alpha\gamma r\lambda \bmod n. \tag{5.15}$$

From (5.15), it can be seen that this signature is of the same form as (5.3). From (5.15) it is seen that it is not important whether k is kept secret or not. If k is known and kept at the same value for multiple signatures, it still depends on the factor γ whether d can be calculated. Because the factor λ is not known by the host, it is impossible for the agent platform to calculate d or δ. Hence, neither the private key nor the temporary private key can be computed.

However, it is possible for the host to compute $\epsilon = \gamma d + c\lambda$, but during the verification process, it will be shown that this does not make it less secure. If parameter Λ were not used, e.g., if no λ occurred in (5.15), it would be possible for the host to calculate the temporary private key δ.

5. **Signature Verification**
The verification formulas are the same as in a conventional digital signature algorithm based on elliptic curves, only here the temporary public key must be used in combination with parameter Λ:

$$\begin{aligned} T &= (sP - r(\Lambda + \Gamma))m^{-1} = (x_1, y_1) \\ t &= x_1 \bmod n \end{aligned} \tag{5.16}$$

The signature is valid if and only if:

$$t = r.$$

For the verification process it is important that Γ and Λ are given to the verifier as two distinct parameters instead of $(\Lambda + \Gamma)$, because the host can calculate $\epsilon = \gamma d + c\lambda$ and hence $\epsilon P = \Lambda + \Gamma$, but the host cannot calculate γd and $c\lambda$ separately and therefore it cannot pretend to be the agent.

By introducing this temporary public and private key, we also achieve something extra besides making it possible to activate the signature at the host. This temporary key pair can be seen as a pseudo-identity of the agent. Parameter Γ must then be registered at the Trusted Third Party (TTP), just as Γ. Giving the agent multiple temporary key pairs means actually giving the agent more identities. Hence, this algorithm can be seen as a privacy enhancing technology [11] for agent-specific applications. Using these types of pseudonyms has an advantage over the simple solution of registering a temporary public key with the TTP and using the corresponding private key without blinding it, because the host cannot compute the temporary private key in the above proposed solution, and hence he or another agent cannot impersonate the agent. Depending on the amount of pseudonyms of an agent, extra overhead is added to the TTP for key distribution and revocation.

Using the parameters known by the host, it is impossible to calculate the private key, because of the hardness of the discrete logarithm problem for elliptic curves. Therefore, the identity of the agent, d, is protected. However, this algorithm does not give the user control over what the agent signs or how many times the host executes this algorithm. The host could repeat the algorithm with different messages and all the signatures would be valid. This drawback makes this algorithm only suitable for a trusted environment. However, it is preferred to the conventional digital signature in trusted environments, because the private key is not revealed at any time. The problem of multiple signing can be compared to the double spending problem in applications like digital anonymous cash [13]. Several solutions to this problem exist, two of which are presented in the next section.

5.4.3 Agent Digital Signature and Solutions to Double Signing Problem

Hosts may be prevented from using an agent's signature multiple time by including the host's identity in the verification of the signature. Each time a signature is verified, the verifier can see at what location the document was signed. This solution does not make the double signing operation impossible, but it will be an extra threshold to using it. Two algorithms are proposed in this chapter to accomplish the idea. The first is one without a signature from the host, only its identity is added to the verification formula. The second algorithm gives two signatures on the message. Both solutions require the agent to know beforehand which hosts it will visit. That knowledge may already be a privacy threat, but this could be solved by providing encapsulated encryption of the host's identities [97].

Agent Signature Combined with Host's Identity

The host's identity must be added in the verification formula. In order to obtain this, the public key is added to this formula. This means that the public key or the private key must also be added in the signature. This solution does not include a host's signature and therefore the host's public key, which represents its identity, is added during the blinding operation on the agent's private key. Adding the host's identity must occur at the user's platform, because this will make it impossible for the host to change its identity in the signature at a later stage.

1. **Key generation**
   As in the previous algorithms, the key generation starts with selecting an elliptic curve E over Z_p, of which the number of points on $E(Z_p)$ is divisible by a large prime n. The user selects a random integer d_a in the interval $[1, n-1]$. Parameter d_a is the agent's private key and is securely stored at the user's computer. The user computes the agent's public key:

   $$Q_a = d_a P \text{ over } E(Z_p). \tag{5.17}$$

   In addition to calculating a regular public key, the user selects the first blinding factor α and computes a "temporary"public key, and as in the previous section parameter Λ:

   $$\begin{aligned} \delta &= \gamma d_a \bmod n, && (5.18)\\ \Gamma_a &= \delta P \text{ over } E(Z_p), && (5.19)\\ c &= \alpha\gamma \bmod n, && (5.20)\\ \Lambda &= \lambda c P \text{ over } E(Z_p), && (5.21) \end{aligned}$$

   where γ is a blinding factor and the combination γd_a can be seen as a temporary private key. The parameters α, γ, λ and d_a are stored securely at the user and are not given to any other element in the system.

   In this algorithm, also the host generates a key pair:

   $$Q_h = d_h P \text{ over } E(Z_p), \tag{5.22}$$

   where d_h is the host's private key and Q_h is its corresponding public key.

2. **Blinding operation on private key**
   This step is equal to the blinding operation in the previous section. Only to the parameter R, the host's identity is added in the form of its public key Q_h:

   $$\begin{aligned} \tilde{R} &= kP = (x_0, y_0), && (5.23)\\ \tilde{r} &= x_0 \bmod n, \\ R' &= \alpha\gamma\tilde{R} + \beta P && (5.24)\\ R &= R' + Q_h = (x_1, y_1) && (5.25)\\ r &= x_1 \bmod n, \\ \tilde{d_a} &= \alpha^{-1} d_a + \lambda \bmod n. && (5.26) \end{aligned}$$

The parameters that are stored in the agent and therefore known to the host are $r, \tilde{d_a}, k, c$ and β and for verification purposes, Γ.

3. **Signature generation**
   Again the signature operation is equal to (5.3), with the exception that d is replaced by $\tilde{d_a}$:
   $$\tilde{s} = km + r\tilde{d_a} \bmod n. \tag{5.27}$$

4. **Activation of signature** The activation does not involve the host's identity, and therefore is equal to the activation in the previous algorithm:
   $$s = c\tilde{s} + \beta m \bmod n. \tag{5.28}$$
   Again parameters c and β are known by the host, but this is not sufficient to compute d_a or γ. Substitution of parameters gives the following signature:
   $$s = (\alpha\gamma k + \beta)m + \gamma r d_a + \alpha\gamma r\lambda \bmod n. \tag{5.29}$$
   Again the signature is of an equal form as in (5.3).

5. **Verification**
   The idea is to add the host's identity in the verification formula, such that it is always possible to know where the signature operation was executed. Adding the host's identity is possible, because the public key of the host is already used in the blinding operation:
   $$\begin{aligned} T &= (sP - r(\Lambda + \Gamma_a))m^{-1} + Q_h = (x_1, y_1) \\ t &= x_1 \bmod n \end{aligned} \tag{5.30}$$
   The signature is valid if and only if:
   $$t = r.$$

This algorithm has the advantage that the host's identity is attached to the agent's signature, which makes it less attractive for the host to sign documents in the agent's name without permission.

A disadvantage, however, is that the agent must know the identities of the hosts it plans to visit. An easy measure to overcome this is storing several R' parameters in the agent and before roaming to another platform, it has the current host add the next host's identity to form parameter R. The signature proposed here is only from the agent and not the host, because the host's private key is not used. In the next section it is shown how this can be achieved.

Combined Agent and Host Signature

This signature is similar to the previous one. In various stages, extra information is added about the host, such that also the host signs the document.

1. **Key generation**
   As in the previous algorithms, key generation starts with selecting an elliptic curve E over Z_p, of which the number of points on $E(Z_p)$ is divisible by a large prime n. The user selects a random integer d_a in the interval $[1, n-1]$. Parameter d_a is the agent's private key and is securely stored at the user's computer. The user computes the agent's public key:

   $$Q_a = d_a P \text{ over } E(Z_p). \tag{5.31}$$

   In addition to calculating a regular public key, the user also computes the "temporary"public key for the agent and the parameter Λ:

   $$\begin{aligned} \delta &= \gamma d_a \bmod n, && (5.32)\\ \Gamma_a &= \delta P \text{ over } E(Z_p), && (5.33)\\ c &= \alpha\gamma \bmod n, && (5.34)\\ \Lambda &= \lambda c P \text{ over } E(Z_p). && (5.35) \end{aligned}$$

   and here, $\gamma \in [1, n-1]$ is a blinding factor and δ can be seen as a temporary private key for the agent. Again, the parameters α, γ, d_a and λ are stored securely at the user and are not given to any other element in the system.

   In this algorithm, also the host generates a key pair:

   $$Q_h = d_h P \text{ over } E(Z_p), \tag{5.36}$$

   where d_h is the host's private key and Q_h its corresponding public key.

2. **Blinding operation on private key**
   This step is equal to the blinding operation in the previous section. Only to the parameter R, the host's identity is added:

   $$\begin{aligned} \tilde{R} &= kP = (x_0, y_0), && (5.37)\\ \tilde{r} &= x_0 \bmod n, && \\ R' &= \alpha\gamma\tilde{R} + \beta P && (5.38)\\ R &= R' + Q_h = (x_1, y_1) && (5.39)\\ r &= x_1 \bmod n, && \\ \tilde{d_a} &= \alpha^{-1} d_a + \lambda \bmod n, && (5.40)\\ c &= \alpha\gamma \bmod n. && (5.41) \end{aligned}$$

   The user also computes a temporary public key for the host:

   $$\Gamma_h = cQ_h \text{ over } E(Z_p). \tag{5.42}$$

   The host can check whether the right public key is used by performing the same operation as in (5.42). The parameters that are stored in the agent and therefore known to the host are $r, \tilde{d_a}, k, c, \Gamma_h$, and β, and for verification purposes, Γ_a.

3. **Signature generation**
   In this step, the signature operation is executed at the host and the signature should involve the private keys of the agent and the host. This can be accomplished by the following operation:

$$\tilde{s} = km + r\tilde{d_a} + rd_h \bmod n. \tag{5.43}$$

4. **Activation of signature**
   The activation does not involve the host's identity, and therefore it is equal to the activation in the previous algorithm:

$$s = c\tilde{s} + \beta m \bmod n. \tag{5.44}$$

   Parameters c and β are known by the host, but this is not sufficient to compute d_a or γ. Substitution of parameters gives the following signature:

$$s = (\alpha\gamma k + \beta)m + \gamma r d_a + \alpha\gamma r d_h + \alpha\gamma r\lambda \bmod n. \tag{5.45}$$

   Again the signature is of the same form as in (5.3), only now it has been signed by two parties.

5. **Verification** The verification formula here is a little different, because also the private key of the host is used to sign the message:

$$\begin{aligned} T &= (sP - r(\Gamma_a + \Lambda) - r\Gamma_h)m^{-1} + Q_h = (x_1, y_1) \\ t &= x_1 \bmod n \end{aligned} \tag{5.46}$$

   The signature is valid if and only if:

$$t = r.$$

This algorithm, like the previous one, makes it less attractive for a host to sign messages in the agent's name without have permission. Here both the agent and the host have signed the document and afterwards the host cannot deny having signed it.

5.5 Conclusions and Discussion

In this chapter, several solutions were presented to provide a digital signature mechanism to mobile software agents. An approach was used to blind the private key in the user's trusted environment. The agent carries the blinded key and when necessary it can perform a signature operation.

With this approach the host does not have access to the private key and cannot compute the private key from the blinded private key as that problem is equivalent to solving the discrete logarithm problem for elliptic curves.

The fact that calculation of the agent's private key is not possible does not mean that the host cannot forge the signature. It is capable of copying the algorithm and

execute it using a different message. This double signing problem could be solved by activating the signature at a trusted location, where the validity can be checked. However, this would limit the autonomy of the agent while we seek a solution that both provides privacy and profits from the advantages of agent technology.

The solutions proposed here to the double signing problem provide a threshold to the host for performing a double signing operation. When each agent's signature includes the host's identity, it is easy to detect malicious hosts.

Some solutions for a digital signature in agent technology are proposed in this chapter, but this is only one of the steps towards securing intelligent software agents. The next step in these digital signatures would be to provide non-repudiation not only for the host, but also for the agent. Non-repudiation for the agent is not yet provided, because if a malicious host intends to sell products, it can double sign an order and the agent is not capable of proving whether it gave permission for this order, or it cannot be proven that the agent gave its permission, because in this case it's in the host's advantage to have its name on the signature. A solution must be found to this problem in order to provide full functionality of a digital signature, which is equivalent to solving the double singing problem.

Chapter 6

Secrecy Systems and Information Theory

In the previous chapters several solutions were proposed to solve partial problems to provide an adequate level of privacy to mobile code. These solutions could be applied in practice, but they do not provide concrete limits for the maximum level of privacy that can be achieved. In the following chapters, an attempt is made to derive these theoretical limits to mobile code protection. The general mobile code privacy protection model as described in chapter 2 is used for this purpose. Note that the attackers in this model have unlimited computation power, memory resources and time. The concepts of information theory are used to derive these boundaries.

This chapter contains two objectives. First, it provides the necessary preliminaries of information theory. The definitions and results as proposed by Shannon [83] are given for the concepts of perfect secrecy, unicity distance, and the cryptographic dilemma [92]. In the conclusions an explanation will be given why these concepts are relevant to mobile code.

The second objective of this chapter is to point out different interpretations of and observations on Shannon's results, and where these differ we argue which interpretation we choose for mobile code protection.

First, Shannon's model of a secrecy system is given, followed by the preliminaries of measures of information. A secrecy system provides confidentiality to the data while it is in transmission. Based on the model, Shannon's definition of perfect secrecy is given (section 6.3), and different interpretations and results are discussed. Section 6.4 covers the unicity distance and 6.5 discusses one special interpretation, the so-called cryptographic dilemma. Finally, conclusions are given in section 6.6.

6.1 Shannon's Secrecy Model

In figure 6.1 the secrecy model is shown as Shannon defined it, only a different notation is used here[1]. The objective of the secrecy model is the possibility to derive a theory on the level of confidentiality towards a message that can be provided, when this message is exchanged between two entities. Secrecy is provided in terms of confidentiality. A message M is encrypted using a key K. The result of this enciphering is a cryptogram C. This operation is described by

$$E_K(M) = C. \tag{6.1}$$

The encryption function $E_K(M)$ gives a one to one relation between the message and ciphertext, such that a ciphertext can be uniquely decrypted based on the key K. Parameter M represents a random variable and can take on values from a finite set $\mathcal{M}$. Capitals are used to represent the random variable and calligraphic letters to represent the corresponding set. To denote the probability of a certain message $P_M(M = m_i)$ the notation $P_M(m_i)$ or $P_M(m)$; the latter we will use when no confusion is possible. All messages have a non-zero probability of occurring. A message m consists of a number of symbols. The parameter m^L denotes that message m consists of L symbols ($L \geq 1$). Superscript in combination with capital letters are used to denote the length of a message or ciphertext. The set of symbols that can be chosen for the message is denoted by $\mathcal{A}$ and exists of the elements $\{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$, where $|\mathcal{A}|$ represents the cardinality of $\mathcal{A}$. In appendix A, table A.1 and A.2, the notations are summarized as a reference. Equivalent notations also hold for the ciphertext and key.

This ciphertext C or cryptogram is transmitted over an unsecure channel to the recipient. The receiver can transform C into the original M only if the correct key K is used. The key is sent over a secure channel to the receiver, for example via courier service. An eavesdropper may listen in on the communication channel and intercept the ciphertext C. The objective of the eavesdropper is either to determine the key K or the corresponding plaintext, M. Furthermore, it is assumed that the attacker knows what type of transformation (e.g. encryption) is done but not with which key. In the information theoretic approach, the model of secrecy systems assumes that the attacker has unlimited computation power and memory resources[2]. This is an important assumption as many of today's cryptographic algorithms depend on the assumption that the attacker has limited resources and time.

The set of keys contains a limited number of elements, and each element has an associated probability. The same holds for the set of messages, and each of these messages has an a priori probability of occurring. Furthermore, it is important that decryption of a ciphertext results in a unique message.

[1] page 661 in [83]. Shannon describes equation (6.1) as a one parameter family of operations, such that $E = T_i M$, which means that transformation T_i applied to message M produces cryptogram E. The index i corresponds to the particular key being used. We will use the notation of equation (6.1).

[2] pp 656/ 659/ 662 in [83]. "The adversary has unlimited time and manpower available for the analysis of intercepted cryptograms." This can nowadays be translated into unlimited computation power and memory resources.

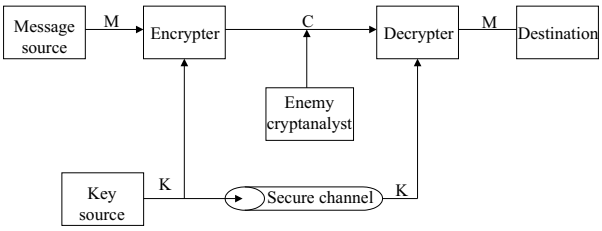

Figure 6.1: Shannon's secrecy model [61].

6.2 Information Theoretic Preliminaries

In [84], Shannon gives a measure of information that provides a level of uncertainty on some random variable. This measure is also called entropy. Using equivalent notations to those introduced in the previous section, entropy is defined as:

Definition 1 (Entropy) *Let X be a random variable, which takes on values from a finite set $\mathcal{X}$. To every possible value $x \in \mathcal{X}$ a probability of occurrence $P_X(x)$ is assigned, and $\sum_{x\in\mathcal{X}} P_X(x) = 1$. The entropy or the average amount of information is defined as:*

$$H(X) = -\sum_{x\in\mathcal{X}} P_X(x) \log P_X(x). \tag{6.2}$$

◇

In case $P_X(x) = 0$, $0 \log 0$ is defined to be 0. The logarithms are base 2. The entropy gives an average level of information or uncertainty for variable X. It is easy to see that the entropy satisfies

$$0 \leq H(X) \leq \log |\mathcal{X}|,$$

where $H(X) = \log |\mathcal{X}|$ is only possible if and only if the distribution on X is uniform. The entropy of X can only be zero if and only if for some i $P_X(x_i) = 1$ and $P_X(x_j) = 0$ for $j \neq i$, which means that there is no uncertainty on the outcome. It is also possible to provide an average level of uncertainty for a combination of two random variables.

Definition 2 (Joint entropy) *Let X and Y be random variables, who take on values from finite sets $\mathcal{X}$ and $\mathcal{Y}$, respectively. The joint entropy is given by:*

$$H(X,Y) = -\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} P_{XY}(x,y) \log P_{XY}(x,y), \tag{6.3}$$

where $P_{XY}(x,y)$ denotes the joint probability of x and y. ◇

Definition 3 (Conditional entropy) *Let X and Y be random variables, who take on values from finite sets $\mathcal{X}$ and $\mathcal{Y}$, respectively. The conditional entropy is given by:*

$$H(X|Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P_{XY}(x,y)\log P_{X|Y}(x|y), \tag{6.4}$$

where $P_{X|Y}(x|y)$ denotes the conditional probability of X taking value x given that Y has value y. ◇

$H(X|Y)$ gives the average level of uncertainty of X given Y. A useful property of conditional entropy is the relation:

$$H(X,Y) = H(X|Y) + H(Y) = H(Y|X) + H(X), \tag{6.5}$$

and the relation

$$H(X,Y|Z) = H(X|Y,Z) + H(Y|Z). \tag{6.6}$$

Furthermore, it always holds that $H(X) \geq H(X|Y)$ as adding information of a second variable cannot increase the uncertainty of the first variable.

Shannon also defined the rate of actual transmission. In the literature this is often called mutual information and is defined as:

Definition 4 (Mutual information) *Let X and Y be random variables, who take on values from finite sets $\mathcal{X}$ and $\mathcal{Y}$, respectively. To every possible value $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ probabilities of occurrence $P_X(x)$ and $P_Y(y)$ respectively, are assigned. The mutual information with regard to X and Y is given by:*

$$I(X;Y) = H(Y) - H(Y|X) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P_{XY}(x,y)\log\frac{P_{XY}(x,y)}{P_X(x)P_Y(y)}, \tag{6.7}$$

where $P_{XY}(x,y)$ denotes the joint probability of x and y. ◇

The quantity $I(X;Y)$ can be interpreted as the amount of information that Y provides about X.

Here, Jensen's inequality will be useful.

Corollary 1 (Jensen's inequality.) *Let f be a continuous strictly concave function on the interval I, and $\sum_{i=1}^{n} a_i = 1$ and $a_i \geq 0; 1 \leq i \leq n$, then*

$$\sum_{i=1}^{n} a_i f(x_i) \leq f\left(\sum_{i=1}^{n} a_i x_i\right),$$

where $x_i \in I$, $1 \leq i \leq n$. Equality occurs if and only if $x_1 = \ldots = x_n$. ◇

For this inequality no proof is given here, but it can be found in most text books on information theory [35]. What is important is that the function $\log_2 x$ is strictly concave on the interval $(0,\infty)$, hence Jensen's inequality will be useful when we work with entropies. It will be used for deriving the unicity distance.

Using these information measures we can determine theoretical boundaries for the level of security or confidentiality a system can provide. The entropy of a message M, $H(M)$, can be interpreted as the average level of uncertainty of this message. Shannon stated that "from the point of view of the cryptanalysis, a secrecy system is almost identical with a noisy communication system. The message (transmitted signal) is operated on by a statistical element, the enciphering system, with its statistically chosen key. The result of this operation is the cryptogram (analogues to the perturbed signal) which is available for analysis"[3].

The above explains why it makes sense to use identical measures to represent the elements in both systems (noisy communication systems and secrecy systems). According to Shannon, two conditional entropies are important: one of the message, $H(M|C)$, and one of the key, $H(K|C)$, where $H(M|C)$ and $H(K|C)$ are called the message and key equivocation, respectively. These entropies can be seen as the average level of secrecy provided by the system. If $H(M|C) = 0$, this requires that one probability $P_{M|C}(m_i|c_j) = 1$ and all others 0. In this case no secrecy is provided given the assumption that the adversary has unlimited computation power and memory resources. Equivalent reasoning holds for $H(K|C)$. Therefore, if the adversary has unlimited computation power and memory resources, one should design a secrecy system where it holds at least that $H(K|C) \neq 0$ and $H(M|C) \neq 0$ and preferably the value for these equivocations should be as high as possible. When it is assumed that the attacker has access to a limited amount of computation power, it may be the case that $H(K|C) = 0$ and still confidentiality is provided. The fact that $H(K|C) = 0$ may mean that on average, the key can be uniquely determined from the given ciphertext, but this does not mean that it will indeed be found as the attacker has limited resources to find the key. Today's public key cryptography is based on this assumption of limited computation power [38], [39], [72]. The level of secrecy is then expressed by using complexity theory.

6.3 Perfect Secrecy

An optimal level of confidentiality in a secrecy system is defined in [83][4]: "It is natural to define perfect secrecy by the condition that for all c the a posteriori probabilities are equal to the a priori probabilities independently of the values of these." Based on this, we give the following definition for perfect secrecy.

Definition 5 (Perfect secrecy) *Let C be the encryption of message M using key K. Perfect secrecy is provided if and only if*

$$P_{M|C}(m|c) = P_M(m), \tag{6.8}$$

for all m and c. ◇

[3] p. 685 in [83]

[4] pp. 679-680 in [83]. Some of the notations have been changed to correspond to the notations of this thesis

Using Bayes' theorem, this is equivalent to $P_{C|M}(c|m) = P_C(c)$ for all m and c ($P_M(m) \neq 0$). This definition is used to prove whether or not systems provide perfect secrecy.

Many textbooks and articles [100], [93], [62] define perfect secrecy in a system where it holds that the mutual information between message and ciphertext is zero, e.g. $I(M;C) = 0$. Based on probability theory, it is easy to show that these two definitions are equivalent [36], therefore perfect secrecy is provided if and only if $H(M|C) = H(M)$ (see appendix B). When defining perfect secrecy for mobile code, we will use the notation with entropies, but here in definition 5 probabilities are used to denote the original definition given by Shannon.

According to Shannon, the definition of perfect secrecy[5] can be interpreted as "the total probability of all keys that transform m_i into a given cryptogram c is equal to that of all keys transforming m_j into the same c for all m_i, m_j and c." The definition of perfect secrecy leads to a number of important observations.

First, looking at the definition for perfect secrecy where $I(M;C) = 0$, e.g. $H(M) = H(M|C)$, we can also interpret it as follows. Knowing the ciphertext will on average not contribute to knowledge of the message. Therefore, in a system where perfect secrecy is provided, on average knowing the ciphertext will not help the adversary to obtain the plaintext.

A second observation is that the definition of perfect secrecy only considers a ciphertext-only attack. In the definition, the adversary may have access to the ciphertext, but the case is not considered where he may have access to a part of the message and its corresponding ciphertext. This observation is important as in the mobile code privacy model plaintext attacks should be taken into account.

Furthermore, a third observation is that the definition of perfect secrecy provides information on the minimum number of keys required to provide perfect secrecy. Shannon shows that, because decryption with a specific key must always lead to a unique solution, the number of different keys must be at least as large as that of the messages to provide perfect secrecy[6]. Assume a fixed key, connecting each m to a cryptogram. Then in order to obtain a unique decryption, one needs at least as many cryptograms as plaintexts. Furthermore, it holds that in case of perfect secrecy $P_{C|M}(c_j|m_j) = P_C(c_j) \geq 0$ for any of these cryptograms and any message. For any other message m_u it must also hold that $P_{C|M}(c_j|m_u) = P_C(c_u) \geq 0$, therefore some other key must connect c_j and m_u. Obviously this must hold for any u; hence the number of keys must at least be equal to the number of messages.

The fourth observation is that using the definition of perfect secrecy a minimum average level of uncertainty for the keys can be derived in case perfect secrecy is provided. A lower bound on the uncertainty of the key can be set for systems that provide perfect secrecy. An informal reasoning of this lower bound is described by

[5] pp. 680-681 in [83]
[6] p. 681 in [83]

Shannon[7], where he states that the entropy of a message is at most $\log n$ ($n = |\mathcal{M}|$) if all messages are equiprobable. This information must be hidden completely, which can only be done if the key uncertainty is at least $\log n$. He concludes that there is a limit to what can be obtained given a uncertainty in the key, namely that the amount of uncertainty introduced into the solution cannot be greater than the key uncertainty[8].

In many textbooks and articles, this has been formalized as follows [100], [91], [62], [61].

Theorem 1 *When perfect secrecy is provided, then entropies of K and M are related as:*

$$H(K) \geq H(M). \tag{6.9}$$

$\diamond$

Proof. The conditional entropy $H(K, M, C)$ can be written as:

$$H(K|M, C) = H(K, M|C) - H(M|C), \tag{6.10}$$

and $H(K|M, C)$ can be written as

$$H(K, M|C) = H(M|K, C) + H(K|C). \tag{6.11}$$

Substituting (6.11) in (6.10) and using $H(M|K, C) = 0$ gives:

$$H(K|M, C) = H(K|C) - H(M|C). \tag{6.12}$$

A property of entropy is that it is always nonnegative, therefore $H(K|M, C) \geq 0$ and thus

$$H(K|C) \geq H(M|C).$$

Furthermore, it always holds that $H(K) \geq H(K|C)$, hence

$$H(K) \geq H(M|C).$$

In case of perfect secrecy, $H(M|C) = H(M)$, which concludes the theorem's proof. $\square$

Hence, when the uncertainty of the key is greater than the uncertainty of the plaintext, perfect secrecy can be provided. If all messages are equiprobable, the maximum uncertainty on M is achieved. This implies that the key space must be larger than or equal to the message space in order to be able to fulfil inequality (6.9). However, it must be noted that when $H(K) \geq H(M)$, this does not mean that perfect secrecy is provided. It can only be stated, from theorem 1 that when $H(K) < H(M)$, it is not possible to provide perfect secrecy.

However, when $H(K) \geq H(M)$, this again does not imply that perfect secrecy is provided.

When perfect secrecy is provided in the special case where $|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$, two characteristics are present:

[7] p. 682 in [83]. Shannon describes this result by using the example with maximum value of H(M) and from this the general principle follows.

[8] Where "solution" is the message in our case.

1. Each message is connected to each cryptogram by exactly one line.
2. All keys are equally likely.

Hence the matrix representation of the system is a Latin square.

Other textbooks and articles have extended the above result and provide formal proofs [89], [86]. They extend them by showing that not only perfect secrecy is characterized by these two properties but that the opposite holds as well. Both books claim this extended result to be Shannon's; however, Shannon's paper only proves the statement in one direction. This result will be important for an example of perfect secrecy for mobile code and therefore the proof is given here.

Theorem 2 *Let (M, C, K) denote a cryptosystem with $|\mathcal{M}| = |\mathcal{C}| = |\mathcal{K}|$, where $|\mathcal{M}|$ denotes the total number of possible values $m \in \mathcal{M}$ can have and $P_M(m) > 0$ for all m. Then the cryptosystem provides perfect secrecy if and only if*

- *every key is used with equal probability $\frac{1}{|\mathcal{K}|}$*
- *for each $m \in \mathcal{M}$ and $c \in \mathcal{C}$ there is a unique key $k \in \mathcal{K}$ such that $E_k(m) = c$.*

$\diamond$

Proof. Assume that the system provides perfect secrecy and that it is possible to find two keys such that

$$E_{k_1}(m) = E_{k_2}(m). \tag{6.13}$$

Furthermore, it is given that $|\mathcal{M}| = |\mathcal{C}| = |\mathcal{K}|$. This implies that given a particular ciphertext, the maximum number of correct corresponding plaintexts is $|\mathcal{K}|$. Otherwise a correct unique decryption is not guaranteed.

When (6.13) is true, this implies that a message and corresponding ciphertext can be found where it holds that $P_{MC}(E_k(m) = c) = 0$. In case of perfect secrecy, this is not allowed, as in that particular case $P_{M|C}(m|c) \neq P_M(m)$. This contradicts (6.13). Therefore it is not possible to find two different keys k_1, k_2 such that $E_{k_1}(m) = E_{k_2}(m)$ when perfect secrecy is provided. Hence, for all $m \in \mathcal{M}$ and $c \in \mathcal{C}$, there is exactly one $k \in \mathcal{K}$ such that $E_k(m) = c$. It needs to be shown that

$$P_K(k) = \frac{1}{|\mathcal{K}|} \text{ for all } k \in \mathcal{K}.$$

For a given $c \in \mathcal{C}$ and perfect secrecy, the probability of a message m_i can be expressed as:

$$\begin{aligned} P_M(m_i) &= P_{M|C}(m_i|c) && (6.14)\\ &= \frac{P_{C|M}(c|m_i)P_M(m_i)}{P_C(c)}. && (6.15) \end{aligned}$$

The previous shows that for each key m_i, there is a key k_i such that $E_{k_i}(m_i) = c$. The probability $P_{C|M}(c|m_i)$ is determined by the probability of the key, hence

$$P_M(m_i) = \frac{P_K(k_i)P_M(m_i)}{P_C(c)}. \tag{6.16}$$

Hence, $P_C(c) = P_K(k_i)$ for all i. This implies that the keys are used with equal probability, therefore $P_K(k_i) = \frac{1}{|\mathcal{K}|}$ for all i.

To prove the opposite direction, we must show that given the requirements to the system (every key is used with equal probability and for each $m \in \mathcal{M}$ and $c \in \mathcal{C}$ there is a unique key k such that $E_k(m) = c$), perfect secrecy will be provided. The joint probability $P_{MC}(m, c)$ is

$$P_{MC}(m, c) = P_{C|M}(c|m)P_M(m),$$

where $P_{C|M}(c|m) = P_K(k)$, as the ciphertext $c \in \mathcal{C}$ is uniquely determined by $k \in \mathcal{K}$ when $m \in \mathcal{M}$ is given, hence

$$P_{MC}(m, c) = P_K(k)P_M(m) = \frac{1}{|\mathcal{K}|}P_M(m).$$

Furthermore, the probability of a ciphertext c is the sum of the probabilities of all possible combinations of k and m that results in a particular c. As all keys are used with equal probability and for each m and c there is a unique key k such that $E_k(m) = c$, the probability of a ciphertext is given by

$$P_C(c) = \sum_{m \in \mathcal{M}} \frac{1}{|\mathcal{K}|}P_M(m). \tag{6.17}$$

The $\sum_{m \in \mathcal{M}} P_M(m) = 1$, hence $P_C(c) = \frac{1}{|\mathcal{K}|}$.

The conditional probability $P_{M|C}(m|c)$ can then be expressed as

$$P_{M|C}(m|c) = \frac{P_{CM}(c, m)}{P_C(c)} = \frac{\frac{1}{|\mathcal{K}|}P_M(m)}{\frac{1}{|\mathcal{K}|}} = P_M(m).$$

Hence, perfect secrecy is provided. □

An example of an encryption algorithm that fulfills theorem 2 (and therefore provides perfect secrecy) is the so-called one-time pad, patented by Vernam [94] in 1917. A message m is chosen and represented as digits corresponding to the alphabet used with cardinality n (if ordinary letters are used, 'A' is represented by 1, etc.). At random a key is selected of the same length as the message m. To compute the ciphertext, an addition between plaintext and the key takes place. Let message m consist of r symbols, $m = a_1 a_2 \dots a_r$, and key of r symbols $k = b_1 b_2 \dots b_r$, then the encryption is defined as:

$$E_k(m) = (a_1 + b_1 \bmod n, a_2 + b_2 \bmod n, \dots, a_r + b_r \bmod n). \tag{6.18}$$

Decryption is the inverse operation, e.g. subtraction of the ciphertext and key. Figure 6.2 shows a graphical representation of the one-time pad. Each message and cipher text is connected by exactly one line (the key).

It can easily be proven that the one-time pad provides perfect secrecy according to Shannon's definition of perfect secrecy. The definition of perfect secrecy is only

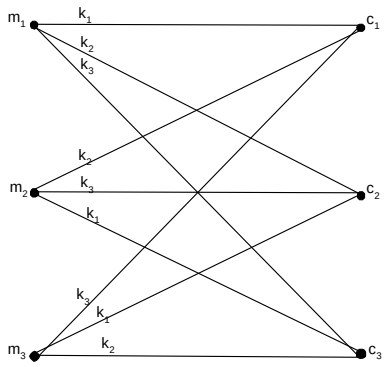

Figure 6.2: Visual representation of the one-time pad. Every message-ciphertext pair is connected by exactly one key

concerned with being resistant against ciphertext-only attacks, it does not provide information whether or not the one-time pad is resistant against plaintext attacks.

Moreover, the one-time pad is vulnerable to known-plaintext attacks. In case the attacker knows that multiple messages are encrypted using the same key, he needs only one message-ciphertext pair to compute all plaintexts. Therefore, the key must be chosen at random each time a message is to be encrypted.

The reasoning that in order to fulfill Shannon's definition of perfect secrecy, different keys must be used for each encryption because the one-time pad is vulnerable to plaintext attacks is not correct as is done in [89]. It is true that in order to fulfill Shannon's definition of perfect secrecy, each time an encryption must to be done, a new key needs to be generated at random, but the reason is not the vulnerability to plaintext attacks as plaintext attacks are not considered in the definition of perfect secrecy.

The reason why a new key must be selected for each encryption to fulfill the definition of perfect secrecy is the following. Consider two messages m_1 and m_2. Each of these messages of length r are encrypted using keys k_1 and k_2, respectively. If we concatenate these two messages (and therefore concatenate the keys k_1 and k_2), the result is a third message $m_3 = m_1||m_2$ (and $k_3 = k_1||k_2$) of length $2r$. The uncertainty of a message of length r is $\log r$ (given that each symbol in a message has an equal probability of occurring). The uncertainty of the concatenated message is then $2 \log r$, as m_1 and m_2 are chosen independently from each other. The same holds for the keys. If these two keys are chosen independently, the uncertainty of the concatenated key is $2 \log r$. However, when the keys for messages m_1 and m_2 are chosen to be equal on purpose, k_2 is completely dependent on k_1, and only k_1 contributes to the uncertainty of k_3; the entropy of the key is in this case only $\log r$. In this case $H(K) \leq H(M)$, and perfect secrecy cannot be provided. Therefore, k_1 and k_2 must be chosen independently from each other.

6.4 Unicity Distance

The definition of perfect secrecy does not consider the length of the ciphertext. When perfect secrecy is provided, it is not relevant whether the attacker has access to half of the ciphertext or to the full length; he is not capable of obtaining the corresponding message. In ciphers where perfect secrecy is not provided, it is easy to imagine that the cryptanalyst will be able to obtain more knowledge on the key when the intercepted ciphertext is longer. To be able to provide a measure of the secrecy level, we need to describe the uncertainty of the key depending on the length of the ciphertext.

The length of the ciphertext where the key equivocation, $H(K|C)$, becomes zero is called the unicity distance.

Definition 6 (Unicity distance (ciphertext-only attacks)) *Let (M^L, C^L, K) be a cryptosystem where $E_K(M^L) = C^L$ is the encryption of message M (of length L) based on key K resulting in ciphertext C^L (of length L). The unicity distance is defined as $UD = \min\{L \in \mathbb{N} | H(K|C^L) = 0\}$.* $\diamond$

The set $\{L \in \mathbb{N} | H(K|C^L) = 0\}$ consists of all lengths of ciphertext such that $H(K|C^L) = 0$. The length with the minimum value is the unicity distance.

When the ciphertext length is equal to or larger than the unicity distance, on average there is a unique solution to determine the key.

Note that the unicity distance is expressed in terms of length of ciphertext whether this length is expressed in bits or words (consisting of multiple bits) or another unit. If the encryption is performed on a word-to-word basis, the unicity distance is expressed in words. It is remarked here that for the definition of the unicity distance the underlying encryption algorithm is not relevant (it is only relevant in the case of computing the value of the unicity distance).

There are different approaches in the literature to determine the unicity distance. The original one (written by Shannon [83]) makes use of the random cipher model. Hellman [46] extends Shannon's approach as he determines the average number of spurious key decipherments (the number of possible keys that lead to a meaningful but incorrect message) and provides an upper bound on the probability that the number of actual spurious keys differs (in the sense that there are less spurious keys than average spurious keys) from the average number for spurious key decipherments. Stinson [89] and Smart [86] also follow this approach of spurious keys. In [96] and [91], the unicity distance is determined at the more conceptual level of entropies. First Shannon's approach is given, followed by the approach of Stinson [89], and finally the approach of [91] and [96] is described. These three approaches are then analyzed, and the one is chosen, which estimates the unicity distance most accurately.

6.4.1 Approach: Shannon

Shannon described the equivocation characteristic for the random cipher model and derived the unicity distance for this model. The random cipher model describes an ensemble of ciphers and is described as follows[9].

[9] p. 691 in [83]. Again here we use different notations that are conform to those in the rest of this thesis

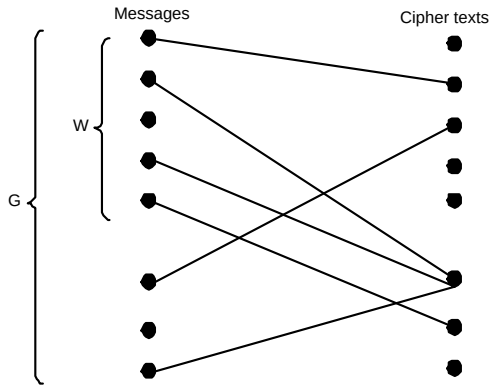

Figure 6.3: An example of the representation for a random cipher.

1. The number of possible messages of length L is $G = 2^{R_0 N}$, thus $R_0 = \log_2 |\mathcal{A}|$, where $|\mathcal{A}|$ is the number of symbols in the alphabet. The number of possible ciphertexts of length L is also assumed to be G.

2. The messages of length L can be divided into two groups. The first group is the group of likely messages (or meaningful messages) and can be approximated well by a uniform distribution. The second group contains messages, which are unlikely and the total probability of this group is negligibly small. The number of messages in the first group is $W = 2^{RL}$, where $R = \frac{H(M)}{L}$ (the entropy of the message source per symbol).

3. An easy way to imagine the cryptosystem is as a series of lines that connects the messages and ciphertexts. Each line represents a key. This is shown in figure 6.3. The total number of different keys is denoted by γ and all keys are equiprobable. Furthermore, the letter r is introduced, which represents the number of keys arriving at a particular ciphertext that have their corresponding plaintext in the set of meaningful messages, hence the number of spurious keys is $r - 1$. The random cipher is actually a whole ensemble of ciphers and the equivocation is the average equivocation for this cipher.

Theorem 3 (Unicity distance (Shannon)) *Let C be the encryption of message M using key K. Based on the random cipher model, the unicity distance is given by*

$$UD \approx \frac{H(K)}{D}, \tag{6.19}$$

where D is the redundancy of the original language per symbol; $D = \frac{\log G - \log W}{L}$, where G and W are the number of messages and the number of meaningful messages, respectively.

◇

Proof. Using equation (6.4), the key equivocation is given by

$$H(K|C) = -\sum_{k\in\mathcal{K}}\sum_{c\in\mathcal{C}} P_{K|C}(k|c)P_C(c)\log P_{K|C}(k|c). \tag{6.20}$$

The probability $p_C(c)$ must be determined.

The probability that at a specific ciphertext r lines come from the meaningful messages can be expressed as:

$$\binom{\gamma}{r}\left(\frac{W}{G}\right)^r\left(1-\frac{W}{G}\right)^{\gamma-r}. \tag{6.21}$$

Hence, the expected value of r becomes:

$$\sum_{r=1}^{\gamma}\binom{\gamma}{r}\left(\frac{W}{G}\right)^r\left(1-\frac{W}{G}\right)^{\gamma-r} r. \tag{6.22}$$

In case a ciphertext with r such lines is intercepted, the conditional entropy is $\log r$. The probability that such a ciphertext is intercepted is $\frac{rG}{W\gamma}$, because this ciphertext can be generated by r keys from the set of high probability messages each with probability $\frac{W}{G\gamma}$. Therefore, the conditional entropy is given by:

$$H(K|C) = \frac{G}{W\gamma}\sum_{r=1}^{\gamma}\binom{\gamma}{r}\left(\frac{W}{G}\right)^r\left(1-\frac{W}{G}\right)^{\gamma-r} r\log r. \tag{6.23}$$

Equation (6.23) can be simplified, based on some assumptions. Assume that γ is large. When the expected value of r is given by $\bar{r} = \frac{W\gamma}{G}$, the variation of $\log r$ over the range where the binomial distribution assumes large values will be small. Therefore, $\log r$ can be replaced by $\log \bar{r}$. Using this and substituting the expected value as expressed in equation (6.22) by $\frac{W\gamma}{G}$ in equation (6.23) leads to

$$H(K|C) \approx \log W - \log G + \log\gamma. \tag{6.24}$$

Let D be the redundancy per letter of the original language ($D = \frac{\log G - \log W}{L}$); then (6.24) evolves in

$$H(K|C) \approx H(K) - DL, \tag{6.25}$$

where L is the length of the ciphertext.

When it is assumed that $\bar{r}$ is small compared to γ, it is possible to approximate the binomial distribution by a Poisson distribution. Then

$$\binom{\gamma}{r}p^r q^{\gamma-r} \approx \frac{e^{-\lambda}\lambda^r}{r!},$$

where $\lambda = \frac{W\gamma}{G}$. Substituting this in (6.23) gives

$$H(K|C) \approx \frac{1}{\lambda}e^{-\lambda}\sum_{2}^{\infty}\frac{\lambda^r}{r!}r\log r.$$

To obtain a simpler expression, r can be replaced by $r+1$:

$$H(K|C) \approx e^{-\lambda} \sum_{1}^{\infty} \frac{\lambda^r}{r!} r \log(r+1),$$

which may be used in the region where λ is near unity. When $\lambda \ll 1$, $H(K|C)$ can be approximated by only taking the term $r = 1$ (as that is the main contributor to $H(K|C)$ into account; all other terms are negligible). Then,

$$\begin{aligned} H(K|C) &\approx e^{-\lambda} \lambda \log 2 \\ &\approx \lambda \log 2 \\ &\approx 2^{-LD} \log 2. \end{aligned} \tag{6.26}$$

When $L = 0$ $H(K|C) = H(K)$ and it decreases linearly according to (6.25), e.g. with a slope of $-D$, to the neighborhood of $L = \frac{H(K)}{D}$. Then a short transition region follows, after which $H(K|C)$ will act like equation (6.26).

Based on this, the unicity distance can be calculated. This is the distance where $H(K|C)$ approaches zero for the first time, hence for the random cipher this is roughly $\frac{H(K)}{D}$. □

6.4.2 Approach: Stinson

Hellman extended Shannon's result on the unicity distance [46] but first showed that a slightly different approach leads to the same approximation for the unicity distance. He introduced the concept of spurious keys. Spurious keys are keys that are candidates for the correct key, but are not the real one. When the number of spurious keys is equal to zero, only one key remains as a candidate and that is the correct one, hence the unicity distance is given by the minimum length of the ciphertext, where the number of spurious keys is equal to zero. This approach differs from Shannon's approach in the sense that instead of determining when $H(K|C) = 0$, it determines when the number of spurious keys is on average equal to zero. These two approaches are equivalent, but the accuracy of the result depends on the estimation of the redundancy. Stinson [89] also uses this concept to determine the unicity distance. The approach given by Stinson will be explained here as to determine the unicity distance we will refer to it elsewhere in this thesis.

In Stinson's derivation for the unicity distance, an assumption is made with respect to the message and the language model used. Here, Stinson's theorem is given without proof as it is followed by an improvement of this theorem where this assumption has not been made.

Definition 7 *Let M^L represent a message of L symbols. The entropy of a natural language is defined as:*

$$H_{lang} = \lim_{L \to \infty} \frac{H(M^L)}{L}, \tag{6.27}$$

and the redundancy of the language is defined as

$$R_{lang} = 1 - \frac{H_{lang}}{\log |\mathcal{A}|}, \tag{6.28}$$

where $\mathcal{A}$ represents the set of symbols in the language.

Corollary 2 (Unicity distance (Stinson)) *Let C be the encryption of message M using key K. The unicity distance is given by*

$$UD \approx \frac{H(K)}{R_{lang} \log |\mathcal{A}|}. \tag{6.29}$$

$\diamond$

In the following theorem the parameter ϵ is introduced that provides a measure until what extent a message corresponds to the language model, e.g. it's statistics. In the case that the characteristics of a message of length L correspond exactly to the language statistics, the entropy of that message will be $L \cdot H_{lang}$. However, because not all messages fulfil these characteristics accurately (such as names and short words), there will be a difference with respect to the ideal case, e.g. the entropy of a message M of length L can be expressed as:

$$LH_{lang} - \epsilon \leq H(M^L) \leq LH_{lang} - \epsilon. \tag{6.30}$$

This is incorporated in the following theorem.

Theorem 4 *Let C be the encryption of message M using key K. The unicity distance is given by*

$$\frac{\log\left(2^{H(K)} - \epsilon\right)}{R_{lang} \log |\mathcal{A}|} \leq UD \leq \frac{\log\left(2^{H(K)} + \epsilon\right)}{R_{lang} \log |\mathcal{A}|}, \tag{6.31}$$

where R_{lang} represents the redundancy of the language and $\mathcal{A}$ represents the set of symbols used in the original language.

Proof. The set of plaintext messages is divided in two subsets of meaningful messages and meaningless messages, where the elements of the set of meaningless messages occur with negligible probability. Let $\bar{s}_L$ be the average number of spurious keys over all ciphertexts of length L and $\mu(c)$ represents the number of keys for which c is the encryption of a meaningful string of plaintext of length L. As the meaningless messages have a negligible probability of occurring, the parameter $\mu(c)$ can also be interpreted as the number of candidate keys when a particular ciphertext is given. With these parameters the average number of spurious keys can be computed. Note that the number of spurious keys for a particular ciphertext is $\mu(c) - 1$, as of all the candidate keys, one will be correct and the rest incorrect. Then the average number of spurious keys over all possible ciphertexts of length L can be expressed as

$$\bar{s}_L = \sum_{c \in \mathcal{C}^L} P_C(c)(\mu(c) - 1), \tag{6.32}$$

where $\mathcal{C}^L$ represents the set of ciphertexts consisting of L symbols. Using the fact that $\sum_{c\in\mathcal{C}^L} P_C(c) = 1$, equation (6.32) can be written as:

$$\bar{s}_L = \sum_{c\in\mathcal{C}^L} [P_C(c)\mu(c)] - 1. \tag{6.33}$$

To determine the unicity distance we need an expression for $H(K|C^L)$, where C^L denotes a ciphertext of length L.

$$H(K, M^L, C^L) = H(C^L|K, M^L) + H(K, M^L) \tag{6.34}$$

Once the key and message are known, the ciphertext can be uniquely determined, e.g. $H(C^L|K, M^L) = 0$. Furthermore, the key is chosen independently of the message, hence equation (6.34) results in:

$$H(K, M^L, C^L) = H(K) + H(M^L). \tag{6.35}$$

Similarly, $H(K, M^L, C^L)$ can also be expressed as

$$H(K, M^L, C^L) = H(M^L|K, C^L) + H(K, C^L), \tag{6.36}$$

where $H(M^L|K, C^L) = 0$, hence $H(K, M^L, C^L) = H(K, C^L)$. Therefore equation (6.35) can be rewritten as

$$H(K, C^L) = H(M^L) + H(K). \tag{6.37}$$

It follows from (6.37) and (6.5) that:

$$\begin{aligned} H(K|C^L) &= H(K, C^L) - H(C^L) \\ &= H(K) + H(M^L) - H(C^L). \end{aligned} \tag{6.38}$$

The uncertainty of a message of length L, $H(M^L)$, depends on the language model, which is represented by H_{lang}. Parameter ϵ describes until what extend message M fulfills this model. The uncertainty of a message is therefore expressed as:

$$LH_{lang} - \epsilon \leq H(M^L) \leq LH_{lang} - \epsilon.$$

First, consider the case where $H(M^L) = LH_{lang} + \epsilon$. The maximum entropy of the ciphertext is $L \log |\mathcal{A}|$. Then it follows that equation (6.38) can be written as

$$H(K|C^L) \geq H(K) + L(1 - R_{lang}) \log |\mathcal{A}| + \epsilon - L \log |\mathcal{A}|,$$

which is

$$H(K|C^L) \geq H(K) - LR_{lang} \log |\mathcal{A}| + \epsilon. \tag{6.39}$$

The conditional entropy $H(K|C^L)$ can be related to the number of spurious keys as follows:

$$H(K|C^L) = \sum_{c\in\mathcal{C}^L} P_C(c)H(K|c). \tag{6.40}$$

The maximum uncertainty of a key given a particular ciphertext is determined by the number of candidate keys, e.g. $\mu(c)$. Hence $H(K|c) \leq \log \mu(c)$. Therefore equation (6.40) results in an inequality:

$$H(K|C^L) \leq \sum_{c \in \mathcal{C}^L} P_C(c) \log \mu(c). \tag{6.41}$$

Hence, Jensen's inequality can be used:

$$H(K|C^L) \leq \log \sum_{c \in \mathcal{C}^L} P_C(c)\mu(c). \tag{6.42}$$

Combining equations (6.33), (6.39) and (6.42) gives

$$\log(\bar{s}_L + 1) \geq H(K) - LR_{lang} \log |\mathcal{A}| + \epsilon.$$

This can be rewritten as

$$\log \bar{s}_L \geq 2^{H(K)} \cdot 2^{-LR_{lang} \log |\mathcal{A}| + \epsilon} - 1. \tag{6.43}$$

The unicity distance is the minimum L for which $\bar{S}_L = 0$. This is only possible if the right term of (6.43) is smaller than 0. Hence,

$$2^{H(K)} \cdot 2^{-LR_{lang} \log |\mathcal{A}| + \epsilon} \leq 1$$

Rewriting this gives

$$|\mathcal{A}|^{LR_{lang}} \geq 2^{H(K)+\epsilon},$$

which can be expressed as

$$L \geq \frac{\log\left(2^{H(K)} + \epsilon\right)}{R_{lang} \log |\mathcal{A}|}. \tag{6.44}$$

The unicity distance is the minimum value of L for which $\bar{s}_L$ can be zero. In this case the uncity distance would become $\frac{\log\left(2^{H(K)}+\epsilon\right)}{R_{lang} \log |\mathcal{A}|}$.

Because the uncertainty of the message has a lower and upper limit, the unicity distance must also be computed for the lower limit, which is analogues to the previous derivation. This results in a unicity distance of $\frac{\log\left(2^{H(K)}-\epsilon\right)}{R_{lang} \log |\mathcal{A}|}$, where $H(M^L) = LH_{lang} - \epsilon$ was used.

Now it must be proven that the unicity distance always takes a value between this upperbound an lowerbound for all ϵ, e.g.

$$\frac{\log\left(2^{H(K)} - \epsilon\right)}{R_{lang} \log |\mathcal{A}|} \leq UD \leq \frac{\log\left(2^{H(K)} + \epsilon\right)}{R_{lang} \log |\mathcal{A}|}.$$

Because the function $\log x$ is a monotonic increasing function, this is the case for all ϵ. □

In the above theorem an equivalent approach was given to Stinson's derivation of the unicity distance. In the case that $\epsilon = 0$, corollary 2 and theorem 4 are equal. This is the case where the message corresponds exactly to the language model. When the message does not correspond exactly to the language model, e.g. $\epsilon \neq 0$, this will influence the unicity distance in a positive or negative way. If $\epsilon < 0$, the unicity distance will be smaller than when $\epsilon = 0$. This is logical because when ϵ is negative, the uncertainty on the message is smaller and therefore the unicity distance is smaller.

In the remaining of this thesis, we will assume that all messages and functions will correspond to the language model made of that particular language.

The reason why Stinson's derivation is used throughout the rest of this thesis is that his approach to provide an expression of the unicity distance is very accurate as the complete statistics of a language are taken into account and not only those of a specific message length, (e.g. H_{lang}).

6.4.3 Approach: van der Lubbe

The third approach to determine the unicity distance is described in [91] and [96]. This approach can be explained using figure 6.4 [91], which shows how different uncertainties depend on the length of the ciphertext. For the $H(K|C^L)$, where L gives the length of the ciphertext, its maximum value is $H(K)$. This will be true when there is no ciphertext ($L = 0$). The larger the ciphertext, the smaller $H(K|C^L)$ will be. It is possible for this value to become zero when on average it is possible to uniquely determine the key from a ciphertext of length L.

A second equivocation that is of interest is $H(M^L|C^L)$. It may be the adversary's objective to determine the message given the ciphertext. This line in figure 6.4 is very similar to the one for $H(K|C^L)$, with one exception; when the ciphertext is small, there can only be a limited number of messages, hence the uncertainty of $H(M^L|C^L)$ for small L is small. The larger the ciphertext becomes, the more messages become possible solutions and $H(M^L|C^L)$ will increase until L is sufficiently large for the ciphertext to contain enough information and $H(M^L|C^L)$ will no longer increase. This curve will then coincide with that of the key equivocation, when the ciphertext will contain sufficient information to retrieve the key from the found plaintext with the same uncertainty and the other way around.

The third equivocation, $H(K|M^L, C^L)$, is similar to $H(K|C^L)$ with a maximum uncertainty of $H(K)$. This equivocation will in general be less than $H(K|C^L)$ but has the same shape.

The length of the ciphertext where $H(K|C^L) = 0$ gives the unicity distance. As said before, the unicity distance of a cryptosystem is defined as the value of L which represents the average amount of ciphertext that is required for an opponent to be able to uniquely compute the key given enough computing time [89]. This value L can be computed in the following way.

Theorem 5 (Unicity distance (van der Lubbe)) *Let C be the encryption of message*

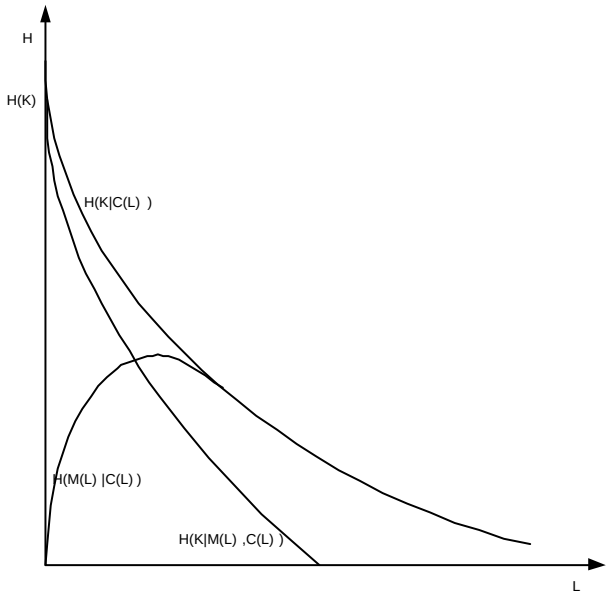

Figure 6.4: Equivocations with respect to the length of the ciphertext.

M *using key* K*. The unicity distance is given by*

$$UD \approx \frac{H(K)}{\log|\mathcal{A}| - H(M)}, \tag{6.45}$$

where $|\mathcal{A}|$ *represents the number of symbols in the original alphabet and* $H(M)$ *the uncertainty of a message of length 1.* ◇

Proof. Because of the one-to-one relation between ciphertext and plaintext, $H(K, C^L)$ will always be equal to $H(K, M^L)$. The conditional entropy $H(K|C^L)$ can be written as:

$$\begin{aligned} H(K|C^L) &= H(K, C^L) - H(C^L), \\ &= H(K, M^L) - H(C^L). \end{aligned} \tag{6.46}$$

It holds that $H(K, M^L) = H(K) + H(M^L)$ because the keys and messages are chosen independently from each other. Let $|\mathcal{A}|$ be the number of symbols in the alphabet; then an upper bound for the uncertainty of the ciphertext is

$$H(C^L) \leq L \log|\mathcal{A}|. \tag{6.47}$$

Combining (6.46) and (6.47) gives

$$H(K|C^L) \geq H(K) + LH(M) - L\log|\mathcal{A}|,$$

where $H(M^L)$ is written as $LH(M)$ because it is assumed that the message source generates the symbols independently. The uncertainty $H(K|C^L)$ can only become zero if

$$L \geq \frac{H(K)}{\log|\mathcal{A}| - H(M)}. \tag{6.48}$$

The unicity distance is the minimum length of ciphertext where $H(K|C^L)$ can become zero. □

6.4.4 Conclusion

The three described approaches lead to an approximation of the unicity distance. The approaches by Shannon and van der Lubbe determine the unicity distance by deriving when $H(K|C^L)$ becomes zero. Stinson's approach determines the average number of spurious keys. These approximations are equivalent. When they are zero on average the key can be uniquely determined.

The three approaches all have results of the form $UD \approx \frac{H(K)}{\Delta}$, where Δ is a measure for the redundancy. The approximation of Δ determines the accuracy of the unicity distance. Therefore, all three approaches are equivalent, but for each the redundancy is determined differently. This redundancy therefore determines which approach will be chosen to model mobile code.

The approach by van der Lubbe uses $H(M)$ to model the language statistics. As $H(M)$ represents the uncertainty of a message of length 1, the approach does not take into account that certain combinations of symbols occur more often than others. For example, in English, the letter 'q' is usually followed by a 'u'; letters are not chosen independently from each other. Therefore, $H(M)$ is a good start to model a language, but it does not provide a high level of accuracy.

Shannon's approach is based on the random cipher model. It is this model that makes the derivation of the redundancy not very accurate. In the random cipher model, the messages are divided into two sets, a set of meaningful messages and a set of meaningless messages. All the elements in the set of meaningful messages have an equal probability of occurring. This can be seen from the expression for the redundancy ($D = \frac{\log G - \log W}{L}$), and in equation (6.21) this can be seen as a binomial distribution is used. Obviously, in practice this is once again not true as certain words have a higher probability of occurring than others. Hence Shannon's approach does not model the language very accurately either.

The approach taken by Stinson is more accurate than the other two. The language is modeled by H_{lang}, which takes into account that certain combinations of symbols occur more frequently than others. Moreover, for the derivation of the unicity distance, no choice is made about the distribution of the messages. All the language characteristics are taken into account by H_{lang} and when this is estimated accurately, Stinson's approach provides an accurate estimate of the unicity distance. Stinson assumes that the plaintext fulfils this language model and states therefore that the entropy of a message of length L can be approximated by $L \cdot H_{lang}$. If the message is sufficiently large, it is highly probable that the plaintext will follow the language model. However, if the

plaintext does not consist of many symbols, it may not follow the language model accurately and this approximation of entropy is not accurate anymore. Therefore, we propose to use the unicity distance based on Stinson's approach under the assumption that the message will be according to the language model.

Because Stinson's approach uses an accurate language model, this approach will be used throughout of this thesis. It will be assumed that all messages consist of sufficiently large number of symbols, such that $H(M^L)$ can be approximated by $L \cdot H_{lang}$.

6.5 Cryptographic Dilemma

Shannon showed that conditional entropy provides a good measure of the secrecy of a system, as was also described in section 6.2. Especially the key equivocation, $H(K|C)$, and message equivocation, $H(M|C)$, are important quantities. It is crucial to have a certain level of uncertainty for the key when the ciphertext is known and equivalent for the message; otherwise no secrecy is provided. These are the two equivocations that Shannon finds significant, however, a third equivocation appears to be important, namely $H(K|M,C)$. This equivocation is especially relevant when a secrecy level must be provided for a system that must be resistant against a plaintext attack, because it describes the uncertainty of the key when a message and ciphertext are given. This is exactly the situation for plaintext attacks. The relation between the three equivocations is easy by making use of standard properties of entropy as was described in section 6.2. The relation between the three equivocations is shown in the following theorem.

Theorem 6 *Let C be the encryption of message M using key K. Then the following is true:*

$$H(K|M,C) = H(K|C) - H(M|C). \tag{6.49}$$

◇

Proof. The joint entropy of the message, ciphertext and key can be written as

$$\begin{aligned} H(M,C,K) &= H(C,K) + H(M|C,K), &(6.50)\\ &= H(M,C) + H(K|M,C). &(6.51)\end{aligned}$$

Rewriting the joint entropies of the ciphertext and key, and of the message and ciphertext gives:

$$\begin{aligned} H(C,K) &= H(C) + H(K|C), &(6.52)\\ H(M,C) &= H(C) + H(M|C). &(6.53)\end{aligned}$$

Using the fact that $H(M|C,K) = 0$, followed by substitution of (6.52) and (6.53) in (6.50) and (6.51) leads to equation (6.49). $H(M|C,K) = 0$ because once the key and ciphertext are known, the plaintext can be uniquely determined. If this were not the case, the owner of the ciphertext and corresponding key would not be guaranteed to obtain the correct plaintext. □

However, this relation was first explicitly stated in [96][10], followed by [91][11]. In the latter, it is interpreted for the first time as a dilemma, called the cryptographic dilemma.

Equation (6.49) can be explained as follows. If an attacker has obtained knowledge of a ciphertext and the goal is to provide confidentiality to the message, it is required that the uncertainty of the message is high (given the ciphertext). If a high value for $H(M|C)$ is obtained equation (6.49) states that $H(K|M,C)$ will be low given that $H(K|C)$ remains constant. In general if one value of equation (6.49) remains constant, an increase of the second value will result in a decrease of the third value. In practice it means that when a system is protected against a ciphertext-only attack additional (physical) measures must be taken to prevent a plaintext attack.

6.6 Conclusions

This chapter gave several concepts for secrecy systems that make use of information theory. The information theoretical approach takes into account an attacker with unlimited resources. For that reason this approach can be used to derive theoretical boundaries of the level of secrecy a system can provide. The definition of perfect secrecy provides this maximum level. If this maximum level cannot be achieved in practice, the unicity distance provides a good measure of the level of secrecy that can be provided, depending on the length of the ciphertext that is known by the attacker.

Furthermore, the cryptographic dilemma provides insight into the relation between protection against ciphertext-only attacks and against plaintext attacks.

These three concepts of perfect secrecy, unicity distance and the cryptographic dilemma will be used to derive theoretical boundaries for mobile code privacy. As said in the description of the mobile code privacy model (section 2.3), privacy is seen as providing confidentiality to mobile code. Because confidentiality is providing secrecy, concepts like perfect secrecy and the unicity distance provide a good measure of the level of privacy that can be achieved.

In this chapter, three approaches for determining the unicity distance were given. Furthermore, an extension to Shannon's approach was given to take small messages into account. The approach of Stinson appears to be the most accurate one and will be used in the remainder of this thesis.

Based on Shannon's work and the description given in this chapter, it can be concluded that Shannon only considered ciphertext-only attacks. This can be seen in the different concepts. The definition of perfect secrecy only considers the relation between knowing a ciphertext or knowing no ciphertext. Only when there is no difference in the probability to find the correct plaintext, the maximum level of secrecy is provided. There is no equivalent expression for the maximum level of secrecy that can be achieved when some plaintext is known to the attacker.

[10] p. 112. In [96] this equation is given to show the relation between message and key equivocation, but no interpretation is provided.

[11] pp. 39-40

Furthermore, the unicity distance is defined to be the minimum length of the ciphertext for which $H(K|C^L) = 0$. Again, only knowledge of the ciphertext is taken into account, but no knowledge of (parts of) the plaintext.

Finally, Shannon considers the equivocations $H(K|C)$ and $H(M|C)$ as being very important, but when plaintext attacks are taken into account $H(K|M, C)$ becomes also relevant.

Until the development of mobile code, it was not necessary to extend Shannon's work to plaintext attacks as the attacker could be prevented from gaining access to plaintext by practical means. However, the mobile code privacy model stated that plaintext attacks cannot always be prevented. Therefore, to derive correct boundaries for the mobile code privacy model, extending Shannon's approach towards plaintext attacks is necessary. The next chapter describes this extension and in the chapters 8 and 9, these results will be applied to mobile code.

Chapter 7

Information Theoretic Approach for Secrecy Systems Taking Into Account Plaintext Attacks

Shannon's results on perfect secrecy and the unicity distance applied to ciphertext-only attacks. In the mobile code privacy model, it is stated that plaintext attacks cannot be completely prevented. To apply Shannon's results to mobile code, his theory should first be extended to plaintext attacks. This chapter presents the results of taking into account plaintext attacks for the concepts of perfect secrecy and the unicity distance.

7.1 Introduction

The mobile code privacy model aims to derive theoretical boundaries of the maximum level of privacy that can be provided to mobile code. Privacy is provided here by means of confidentiality to the mobile code. Shannon's results as discussed in the previous chapter provide measures to express maximum boundaries of secrecy for conventional secrecy systems.

Important is the observation in chapter 6 that Shannon's definitions and results are generally applicable to ciphertext-only attacks.

For conventional data protection, plaintext attacks were not that relevant as in practice the attacker could be prevented from access to message-ciphertext pairs, although the implications can be great. In the mobile code privacy model of section 2.3, it was stated that plaintext attacks cannot be prevented at all times; therefore they should be taken into account. It is not true that when encryption algorithms can protect data against ciphertext-only attacks, such data will also be resistant to plaintext attacks. For example, when the one-time pad (OTP) is used correctly, for each encryption a new key is chosen at random, and an attacker is not capable of obtaining a message

ciphertext pair. However, if the key were the same for a number of encryptions, no secrecy would be provided if the attacker had access to a part of the plaintext.

If public key encryption schemes are used, plaintext attacks cannot be prevented. However, the level of secrecy is then given by using complexity theory as information theory cannot be applied for public key cryptography (see section 2.3.2)[63].

In this chapter, Shannon's results and concepts are used to derive theoretical bounds to secrecy and privacy when plaintext attacks are taken into account, such that the attacker has access to a number of ciphertexts and corresponding plaintexts.

Section 7.2 provides detailed information on the type of attack considered combined with overall assumptions. It will be shown that two types must be distinguished; and each of these will be treated separately in section 7.3 and 7.4, respectively. For both types, perfect secrecy is defined differently and it is shown under which condition such secrecy can be provided. It will be shown that the usage of the OTP and the definitions of a plaintext attack and perfect secrecy will determine whether the OTP is resistant against a plaintext attack. Also new expressions for the unicity distance are given. Section 7.5 provides conclusions for this chapter.

7.2 Problem Statement and Assumptions

The objective of this chapter is to derive measures that give the limits of the maximum level of secrecy that can be provided in a secrecy system which takes into account plaintext attacks.

The results described in this chapter are based on Shannon's model described in section 6.1. A message M is encrypted using key K. This results in the cryptogram C, which can only be correctly decrypted with the correct key K. An attacker has unlimited computation power and resources available to break the system.

The following attack is considered. The attacker has obtained l pairs of messages and ciphertexts, and one ciphertext without knowing its corresponding message. These messages and ciphertexts are all of equal length. The objective of the attacker is to either determine the key used to obtain the ciphertext or the corresponding message. This is called a plaintext attack.

No difference is made between a known plaintext and chosen-plaintext attack as the results do not differ if this distinction is made; therefore just the general problem of a plaintext attack is addressed. In case of chosen-plaintext attacks, the attacker can decide what plaintexts are encrypted and he obtains the corresponding ciphertexts. For a known plaintext attack, the attacker has access to l pairs of (m, c), but the plaintexts of these pairs are not chosen by the attacker. We will explain why results for known and chosen-plaintext attacks are equal when defining perfect secrecy is defined that takes these attacks into account.

The following notation will be used to denote that l message- ciphertext-pairs are known to the attacker. $\mathcal{T}$ represents the set of all possible message-ciphertext pairs. Parameter T is a random variable that can take on values from the finite set

$\mathcal{T}$. Each element in $\mathcal{T}$ consists of a (m, c)-pair. The probability $P_T(T = t_i)$ denotes the probability that T takes the value t_i, where t_i consist of a (m, c)-pair. $\mathcal{T}^l$ is a subset of $\mathcal{T}$ and consists of l elements. The notation T^l is short for $T_1, \ldots, T_l$, e.g. l elements of $\mathcal{T}$ or all elements of $\mathcal{T}^l$. The notation $P_{T^l}(t_1, \ldots, t_l)$ denotes the joint probability of l (m, c) pairs. How these elements (based on which key) are generated is dependent on the cryptosystem and will be explained in section 7.3 and 7.4. Note that capital superscripts denote the number of symbols a message exists of (M^L denotes a message of L symbols), while a lower case capital superscript denotes the number of available messages (e.g. T^l denotes l message-ciphertext pairs).

As in the mobile code privacy model, it is assumed that the encryption algorithm is deterministic. Relevant equivocations that provide a measure of secrecy are then $H(K|C, T^l)$; $H(M|C, T^l)$ and $H(K|M, C, T^l)$.

Two types of plaintext attacks must be distinguished in order to be able to extend Shannon's definition on perfect secrecy. The first is that no limitations are set to the known message-ciphertext pairs. The attacker has access to l message-ciphertext pairs and the keys that are used to transform these messages in ciphertexts can all be different. The pairs of messages and corresponding ciphertexts are generated as follows.

Protocol 1
1. *Select a message $m_i \in \mathcal{M}$.*
2. *Select a key $k_j \in \mathcal{K}$ at random.*
3. *Encrypt message m_i using key k_j: $c_h = E_{k_j}(m_i)$.*
4. *If (m_i, c_h) is an element of $\mathcal{T}^{i-1}$, repeat steps (1)-(3), else (m_i, c_h) is added to the set of (m, c) pairs, resulting in $\mathcal{T}^i$.*

Steps (1) - (4) are repeated for $i = 1, 2, \ldots, l$.

The protocol results in l pairs of messages and corresponding ciphertexts. The messages can be selected at random or according to some protocol. The method used to select the messages determines whether it concerns a chosen or known plaintext attack. Different indices are used for m, k and c, because if message m_1 is selected and encrypted resulting in ciphertext c_1, encryption with a different key should lead to c_2. For both pairs (m_1, c_1) and (m_1, c_2) it should be possible to be elements of $\mathcal{T}$. Therefore, different indices should be used. The source that generates messages of length L can be considered memoryless.

The attacker's objective is to derive the plaintext when a new ciphertext is given, based on the l known (m, c) pairs and to derive the key used for that last encryption.

The second type is that where all the l obtained message-ciphertext pairs and the new ciphertext have been encrypted using the same key. The objective is to obtain the key or the message using these l message-ciphertext pairs. It is assumed that the encryption algorithm is deterministic. If the same key is used for each encryption, the encryption of one message will always result in the same ciphertext. In the model the attacker can have access to ciphertexts and their corresponding plaintexts. If the

sender encrypts a message twice, the attacker knows the corresponding message based on a ciphertext he has seen before. Therefore, a sender will encrypt each message at maximum once. Hence, the source to generate the messages will have the a priori condition that once a message has been selected, the message will be removed from the set of messages that can be selected in the future[1]. The l pairs of messages and ciphertexts are in this case generated using protocol 2.

Protocol 2 *1. Select a key $k_j \in \mathcal{K}$ at random.*

(a) Select a message $m_i \in \mathcal{M}$ such that m_i is not already an element of the (m, c) pairs in $\mathcal{T}^{i-1}$.

(b) Encrypt message m_i using key k_j: $c_h = E_{k_j}(m_i)$.

(c) The pair (m_i, c_h) is added to the set of pairs, resulting in $\mathcal{T}^i$.

The steps (a) - (c) are repeated for $i = 1, 2, \ldots, l$.

By following this protocol, a set of l pairs of messages and corresponding ciphertexts are generated, which are known to the attacker. This type of plaintext attack where the key is equal for each message-ciphertext pair is the more common approach to assess encryption schemes with respect to plaintext attacks, because it is usually considered in public key cryptography. Using this assumption, a measure can be given of how many times a key can be used while an optimal level of secrecy is still obtained.

For these two types it is then possible to extend Shannon's definition on perfect secrecy to overcome the problem of knowledge of a limited number of plaintext-ciphertext pairs. In the remainder of this chapter, these two types are treated separately. First, the general case of no limitations to the keys used is analyzed, followed by the case where only one key is used.

7.3 Plaintext Attacks Based on Usage of Different Keys

In this section the first type of plaintext attacks is considered. It is assumed that the attacker has access to l message-ciphertext pairs, and one ciphertext of which the attacker does not know the original plaintext. The keys used to generate the known message-ciphertext pairs may be different. This assumption is important as this will influence the definition of perfect secrecy. It is not necessary for the key used to generate the ciphertext (of which the corresponding plaintext is not available) to be one of the keys used to generate the l message-ciphertext pairs.

As an immediate consequence of this assumption the number of available message-ciphertext pairs to the attacker can be as great as the total number of message-ciphertext pairs available in the cryptographic system and still a maximum level of secrecy can be provided. This will be explained in more detail later after we have defined perfect secrecy.

[1] This can be compared to "sampling with replacement" as is known from probability theory [56]

7.3.1 Perfect Secrecy

Section 6.3 gave Shannon's original definition of perfect secrecy. This definition can easily be extended to known plaintext attacks where different keys are used. This results in the following definition.

Definition 8 *Let C be the encryption of message M using key K. The encrypted message has perfect secrecy against a plaintext attack where l message-ciphertext pairs are given if and only if:*

$$H(M|C, T^l) = H(M), \tag{7.1}$$

where T^l represents l pairs of corresponding plaintexts and ciphertexts. The l (m, c) pairs are generated following protocol 1. ◇

The conditional entropy $H(M|C, T^l)$ represents the average uncertainty of a message once its corresponding ciphertext and l message-ciphertext pairs have been given. It may be the case that c is a part of an element of $\mathcal{T}^l$, where $\mathcal{T}^l$ represents the set of l message-ciphertext pairs.

This definition is equivalent to Shannon's original definition from section 6.3, but the attacker has more knowledge of the system, therefore perfect secrecy against this kind of attacks is provided only if the attacker does not gain more certainty about the plaintext when he knows the ciphertext and l message-ciphertext pairs compared to the situation where he has no knowledge about these items.

Definition 8 can also be written as: perfect secrecy is guaranteed when it holds that $p(m|c, t_1, \ldots, t_l) = p(m)$ for all messages, ciphertexts and all combinations of (m, c) pairs. As this definition must hold for all known (m, c) pairs, it is independent of the type of plaintext attack, known or chosen-plaintext attack. Therefore, the general term 'plaintext attack' may be used. The distinction between chosen and known plaintext attacks only becomes relevant when discussing a particular encryption algorithm.

7.3.2 Properties of Perfect Secrecy

It is now possible to derive the minimum level for uncertainty of the key when perfect secrecy is provided.

Theorem 7 *A cryptosystem that provides perfect secrecy according to definition 8 will have the characteristic*

$$H(K) \geq H(M). \tag{7.2}$$

◇

Proof. The joint entropy of the key, message, ciphertext and the l (m, c) pairs can be written as:

$$\begin{aligned} H(K, M, C, T^l) &= H(K|M, C, T^l) + H(M, C, T^l), & (7.3)\\ &= H(M|K, C, T^l) + H(K, C, T^l), & (7.4) \end{aligned}$$

where $H(M|K,C,T^l) = 0$ as knowledge of the key and ciphertext is sufficient to obtain the corresponding plaintext. The joint entropies $H(M,C,T^l)$ and $H(K,C,T^l)$ can be written as:

$$H(M,C,T^l) = H(M|C,T^l) + H(C,T^l), \tag{7.5}$$
$$H(K,C,T^l) = H(K|C,T^l) + H(C,T^l). \tag{7.6}$$

The equations (7.3) and (7.4) are equal and substituting equations(7.5) and (7.6) gives:

$$H(K|M,C,T^l) = H(K|C,T^l) - H(M|C,T^l). \tag{7.7}$$

As entropy is always non-negative, equation (7.7) fulfills:

$$H(K|C,T^l) \geq H(M|C,T^l).$$

Using $H(K) \geq H(K|C,T^l)$, the following holds:

$$H(K) \geq H(M|C,T^l). \tag{7.8}$$

Based on the definition of perfect secrecy, inequality (7.8) results in theorem 7. □

Theorem 7 gives the lower bound for the uncertainty of the key and this bound is equal to the bound where plaintext attacks are not taken into account. This is logical as each time an encryption takes place, a new key is selected, and hence the information gained from the l (m,c) pairs will not contribute to the certainty of the message given a ciphertext in case of perfect secrecy. If all messages have an equal probability of occurrence (e.g. when a maximum level of uncertainty is achieved), the key space must be at least as large as the message space.

It has been proven that the one-time pad (OTP) provides perfect secrecy according to Shannon's original definition when each key is used only once. Here it is proven that the OTP fulfills definition 8 when for each encryption a new key is selected.

Theorem 8 *Let (M,C,K) denote a cryptosystem based on the encryption scheme of the one-time pad, where $|\mathcal{M}| = |\mathcal{C}| = |\mathcal{K}|$. Using protocol 1, l message ciphertext pairs are generated. This system provides perfect secrecy in the sense of definition 8.* ◇

Proof. In the one-time pad encryption scheme all keys have equal probability, e.g. $\frac{1}{|\mathcal{K}|}$. Furthermore, for each $m \in \mathcal{M}$ and $c \in \mathcal{C}$ there is a unique key such that $E_k(m) = c$. To prove the perfect secrecy it is necessary to show that $P_{M|CT^l}(m|c,t_1,\ldots,t_l) = P_M(m)$, which is equivalent to proving $H(M|C,T^l) = H(M)$, see appendix B. The joint probability of the ciphertext and l message-ciphertext pairs $P_{CT^l}(c,t_1,\ldots,t_l)$ can be written as

$$P_{CT^l}(c,t_1,\ldots,t_l) = P_{C|T^l}(c|t_1,\ldots,t_l)P_{T^l}(t_1,\ldots,t_l), \tag{7.9}$$

The conditional probability of a ciphertext, given l pairs of (m,c), is determined by how many combinations of (m,k) pairs result in that specific ciphertext. These

messages and keys are chosen independently following a uniform distribution. The probability $P_{M|T^l}(m = d_k(c)|t_1, \ldots, t_l)$ is equal to one when the (k, c) pair leads to that particular message and otherwise zero . Then the conditional probability $P_{C|T}(c|t_1, \ldots, t_l)$ can be expressed as

$$P_{C|T^l}(c|t_1, \ldots, t_l) = \sum_{k \in \mathcal{K}} \sum_{m \in \mathcal{M}} P_{M|T^l}(m|t_1, \ldots, t_l) P_{K|T^l}(k|t_1, \ldots, t_l) P_{M|T^l}(m = d_k(c)|t_1, \ldots, t_l) \quad (7.10)$$

For a one-time pad, the number of combinations of (k, m) that lead to a specific ciphertext is equal to $|\mathcal{M}|$, therefore equation (7.10) can be written as

$$P_{C|T^l}(c|t_1, \ldots, t_l) = \frac{1}{|\mathcal{K}|}. \quad (7.11)$$

The probability of l (m, c) pairs is $\frac{l}{|\mathcal{T}|}$. Substitution of this and equation (7.9) in equation (7.11) gives

$$P_{CT^l}(c, t_1, \ldots, t_l) = \frac{1}{|\mathcal{K}|} \frac{l}{|\mathcal{T}|}.$$

The conditional probability $P_{M|CT^l}(m|c, t_1, \ldots, t_l)$ can be written as

$$P_{M|CT^l}(m|c, t_1, \ldots, t_l) = \frac{P_{C|MT^l}(c|m, t_1, \ldots, t_l) P_{MT^l}(m, t_1, \ldots, t^l)}{P_{CT^l}(c, t_1, \ldots, t_l)}. \quad (7.12)$$

The variables M and T are independent, as each message has equal probability to be chosen once l message-ciphertext pairs are known, therefore
$P_{MT^l}(m, t_1, \ldots, t_l) = P_M(m) P_{T^l}(t_1, \ldots, t_l)$. In addition, if $c = E_k(m)$, then
$P_{C|MT^l}(c|m, t_1, \ldots, t_l) = P_K(k) = \frac{1}{|\mathcal{K}|}$. Then (7.12) results in

$$P_{M|CT^l}(m|c, t_1, \ldots, t_l) = \frac{\frac{1}{|\mathcal{K}|} P_M(m) \frac{l}{|\mathcal{T}|}}{\frac{1}{|\mathcal{K}|} \frac{l}{|\mathcal{T}|}},$$

hence $P_{M|CT^l}(m|c, t_1, \ldots, t_l) = P_M(m)$. □

The maximum number of allowed message-ciphertext pairs with which still perfect secrecy can be provided is equal to $|\mathcal{T}| = |\mathcal{M}|^2$. Consider the one-time pad, where the attacker knows all the possible message-ciphertext pairs. If the attacker receives a ciphertext, he will not be able to determine the corresponding message as in his database, the ciphertext will occur $|\mathcal{K}|$ times, once for every key. As he does not know the key, he will not be able to determine the original message.

7.3.3 Unicity Distance

Section 6.4 gave several methods to compute the unicity distance. The unicity distance was defined as the minimum value of L (length of ciphertext) such that $H(K|C^L) = 0$. If the opponent has access to this amount of ciphertext on average he will be able to

compute the key (given unlimited computation time). Important was the realization that for a language, there is a set of meaningful and a set of meaningless messages, i.e., a set of messages with a relatively high probability of occurring versus messages with a low probability of occurring. Furthermore, the value of the unicity distance is directly related to the level of redundancy in the language used. In this section, the unicity distance will be derived for plaintext attacks. The model used here assumes that the lengths of messages and ciphertexts (the ciphertexts as part of the pairs and the one the attacker wishes to decipher) are of equal length. The derivation for the unicity distance described by Stinson [89] will be used as it gives a clear relation between the number of spurious keys and the redundancy of a language. As in chapter 6, a cryptographic system can be represented by a set of lines that connects messages to ciphertexts. The cardinality of the set of plaintext symbols is equal to that of ciphertext symbols.

The unicity distance taking plaintext attacks into account is given by:

Definition 9 (Unicity distance (plaintext attacks)) *Let (M^L, C^L, K) be a cryptosystem where $E_K(M^L) = C^L$ is the encryption of message M (of length L) based on key K resulting in ciphertext C^L (of length L). Let T^l denote l pairs of messages and corresponding ciphertexts. The unicity distance is defined as*

$$UD = \min\{L \in \mathbb{N} | H(K|C^L, T^l) = 0\}.$$

$\diamond$

Theorem 9 *Let (M, C, K) denote a symmetric encryption scheme such that $E_k(m) = c$; l pairs of messages and ciphertexts are generated by executing protocol 1. Then the unicity distance is given by:*

$$UD \approx \frac{H(K)}{R_{lang} \log |\mathcal{A}|}, \tag{7.13}$$

where R_{lang} represents the redundancy of the plaintext, and $\mathcal{A}$ is the set of symbols used to construct a message. $\diamond$

Proof. Let $\mu(c, l)$ represent the number of candidate keys for a given ciphertext c and l pairs of (m, c) (based on protocol 1). The number of spurious keys is then for a particular ciphertext equal to $\mu(c, l) - 1$. The average number of spurious keys for a given ciphertext of length L as a function of l is then given by

$$\bar{s}_L(l) = \sum_{c \in \mathcal{C}(L)} \sum_{t_1 \in \mathcal{T}(L)} \cdots \sum_{t_l \in \mathcal{T}(L)} P_{CT^l}(c, t_1, \ldots, t_l)(\mu(c, l) - 1),$$

where $\mathcal{C}(L)$ and $\mathcal{T}(L)$ represent the sets of ciphertexts of length L and of message ciphertext pairs of length L, respectively. This can be represented as

$$\bar{s}_L(l) = \sum_{c \in \mathcal{C}(L)} \sum_{t_1 \in \mathcal{T}(L)} \cdots \sum_{t_l \in \mathcal{T}(L)} [P_{CT^l}(c, t_1, \ldots, t_l)\mu(l, c)] - 1, \tag{7.14}$$

because $\sum_{c\in\mathcal{C}(L)}\sum_{t_1\in\mathcal{T}(L)}\cdots\sum_{t_l\in\mathcal{T}(L)}[P_{CT^l}(c,t_1,\ldots,t_l)] = 1$. To determine the unicity distance when plaintext attacks are allowed, one should not just consider the conditional entropy $H(K|C^L)$ but should also take into account that the attacker has access to l message-ciphertext pairs. Hence, it is important to determine $H(K|C^L,T^l)$, the uncertainty of the key given a ciphertext of length L and l (m,c) pairs (all of length L), $H(K|C^L,T^l)$. This conditional entropy can be written as

$$H(K|C^L,T^l) = H(K,C^L,T^l) - H(C^L,T^l). \tag{7.15}$$

The joint entropy $H(K,C^L,T^l)$, can be rewritten. The joint entropy $H(K,M^L,C^L,T^l)$ is equal to

$$\begin{aligned} H(K,M^L,C^L,T^l) &= H(M^L|K,C^L,T^l) + H(K,C^L,T^l) && (7.16)\\ &= H(C^L|K,M^L,T^l) + H(K,M^L,T^l) && (7.17)\end{aligned}$$

The conditional entropy $H(M^L|K,C^L,T^l)$ is equal to zero as knowledge of the key and ciphertext makes it possible to compute the corresponding message. The conditional entropy $H(C^L|K,M^L,T^l)$ is also equal to zero, because knowledge of the message and key makes it possible to encrypt the message resulting in C^L. Therefore, from equations (7.16) and (7.17) and these conditional entropies it is derived that $H(K,C^L,T^l) = H(K,M^L,T^l)$. Using this and rewriting the joint entropy in the form of conditional entropy, equation (7.15) evolves in

$$H(K|C^L,T^l) = H(M^L|K,T^l) + H(K,T^l) - H(C^L,T^l). \tag{7.18}$$

According to protocol 1, each time a message must be encrypted, the key is chosen at random, therefore

$$\begin{aligned} H(K,M^L,T^l) &= H(K|M^L,T^l) + H(M^L,T^l)\\ &= H(K) + H(M^L,T^l) && (7.19)\end{aligned}$$

The joint entropy $H(K,M^L,T^l)$ can also be written as

$$\begin{aligned} H(K,M^L,T^l) &= H(M^L|K,T^l) + H(K,T^l)\\ &= H(M^L|K,T^l) + H(K) + H(T^l). && (7.20)\end{aligned}$$

The latter is true because the key is chosen at random for each encryption. Combining equations (7.19) and (7.20) results in

$$H(M^L|K,T^l) = H(M^L,T^l) - H(T^l). \tag{7.21}$$

Hence, $H(M^L|K,T^l) = H(M^L|T^l)$.

Furthermore, the joint entropies $H(K,T^l)$ and $H(C^L,T^l)$ can be written as conditional entropies. Equation (7.18) then becomes:

$$H(K|C^L,T^l) = H(M^L|T^l) + H(K|T^l) - H(C^L|T^l). \tag{7.22}$$

Because the cardinality of the sets for plaintext symbols and ciphertext symbols is equal, the maximum value $H(C^L|T^l)$ can take is equal to $L\log|\mathcal{A}|$, where $\mathcal{A}$ is defined as the set of symbols for the plaintexts. Therefore:

$$H(C^L|T^l) \leq L\log|\mathcal{A}|. \tag{7.23}$$

This upper limit of the conditional entropy is equal to the upper limit of $H(C^L)$. This is realistic, because each time, the message is chosen independently from the known (m,c) pairs, therefore knowledge of the (m,c) pairs will not influence the uncertainty of a certain ciphertext. Using inequality (7.23), equation (7.22) results in:

$$H(K|C^L,T^l) \geq H(M^L|T^l) + H(K|T^l) - L\log|\mathcal{A}|. \tag{7.24}$$

Now an expression for $H(M^L|T^l)$ is needed. Because where different keys are chosen for each encryption, each message can also be chosen independently from the previously selected messages, hence $H(M^L|T^l) = H(M^L)$. Let H_{lang} denote the entropy of the language per symbol and is defined as $H_{lang} = 1 - R_{lang}\log|\mathcal{A}|$, as was given in section 6.4. Provided L is reasonably large, $H(M^L)$ can be approximated by $H(M^L) \approx LH_{lang}$. Hence, $H(M^L|T^l) = L(1-R_{lang})\log|\mathcal{A}|$, equation (7.24) results in

$$H(K|C^L,T^l) \geq H(K|T^l) - LR_{lang}\log|\mathcal{A}|. \tag{7.25}$$

As this is an expression in terms of the ciphertext length L, we want to relate the number of spurious keys to $H(K|C^L,T^l)$ because this will relate to the unicity distance. $H(K|C^L,T^l)$ can also be written as:

$$H(K|C^L,T^l) = \sum_{c\in\mathcal{C}(L)}\sum_{t_1\in\mathcal{T}(L)}\cdots\sum_{t_l\in\mathcal{T}(L)} P_{CT^l}(c^L,t_1,\ldots,t_l)H(K|c^L,t_1,\ldots,t_l). \tag{7.26}$$

The entropy $H(K|c^L,t_1,\ldots,t_l)$ provides the uncertainty of the key given a specific ciphertext and l pairs of (m,c). This uncertainty is determined by the number of candidate keys e.g. $\mu(c,l)$. The maximum value of this conditional entropy is therefore $\log(\mu(c,l))$. Therefore, equation (7.26) can be rewritten as:

$$H(K|C^L,T^l) \leq \sum_{c\in\mathcal{C}(L)}\sum_{t_1\in\mathcal{T}(L)}\cdots\sum_{t_l\in\mathcal{T}(L)} P_{CT^l}(c^L,t_1,\ldots,t_l)\log(\mu(c,l)). \tag{7.27}$$

Using Jensen's inequality, this results in

$$H(K|C^L,T^l) \leq \log\left[\sum_{c\in\mathcal{C}(L)}\sum_{t_1\in\mathcal{T}(L)}\cdots\sum_{t_l\in\mathcal{T}(L)} P_{CT^l}(c^L,t_1,\ldots,t_l)\mu(c,l)\right].$$

Hence, using equation (7.14), it results in

$$H(K|C^L,T^l) \leq \log(\bar{s}_L(l)+1). \tag{7.28}$$

Combining equations (7.25) and (7.28) results in a relation between the ciphertext length and the average number of spurious keys:

$$\log(\bar{s}_L(l)+1) \geq H(K|T^l) - LR_{lang}\log|\mathcal{A}|.$$

Then the average number of spurious keys becomes

$$\bar{s}_L(l) \geq \frac{2^{H(K|T^l)}}{|\mathcal{A}|^{LR_{lang}}} - 1. \tag{7.29}$$

Furthermore, the choice of keys is independent from the l (m,c) pairs, therefore $H(K|T^l) = H(K)$, the average number of spurious keys becomes

$$\bar{s}_L(l) \geq \frac{2^{H(K)}}{|\mathcal{A}|^{LR_{lang}}} - 1.$$

The unicity distance is given by the minimum value of L for which $\bar{s}_L(l)$ can be zero, hence the unicity distance is equation (7.13).

□

Equation (7.13) is equal to the unicity distance, as was derived in section 6.4 where no plaintext attack was allowed. This can be explained as follows. The attacker will gain more knowledge through this access to l (m,c) pairs, but because these pairs are generated with keys that randomly selected for each encryption, they will not provide extra knowledge on the ciphertext of which the attacker wants to compute the key. This results in the same expression for the unicity distance as that where the attacker does not know the (m,c) pairs. If all keys are equiprobable $H(K)$ can be replaced by $\log|\mathcal{K}|$.

7.3.4 Conclusions

In this section, Shannon's secrecy model was adapted such that it takes into account that an attacker may have access to a number of message-ciphertext pairs. However, for each encryption a new key is generated at random. Because of this assumption, the requirement to obtain perfect secrecy is equal to Shannon's original requirement. Equivalently, the unicity distance is expressed by the same equation used in the model where plaintext attacks are not taken into account.

Concluding, it can be said that when keys are chosen independently for each encryption, terms like unicity distance and perfect secrecy are independent from an attacker's access to previously encrypted messages and their corresponding ciphertexts.

7.4 Plaintext Attacks Based on Usage of Identical Keys

This section shows how the original definition of perfect secrecy can be extended to include plaintext attacks, but here the known message-ciphertext pairs are generated using one key. These pairs are generated using protocol 2. A new definition of perfect

secrecy must be given, as it will be shown that Shannon's original definition will not be adequate anymore. Several options for this extended definition are explored and arguments are given why one particular definition for perfect secrecy has been chosen. Then we will derive the number of keys necessary related to the number of possible messages and of known message-ciphertext pairs. Using this, we will then give an example of such a system and provide proof of perfect secrecy. Similar to the previous section, the unicity distance is determined to give the minimal length of ciphertext that must be available on average to be able to uniquely determine the corresponding key.

7.4.1 Definition of Maximum Secrecy

When plaintext attacks, based on one key, are taken into account, the definition of perfect secrecy is

Definition 10 (Perfect secrecy.) *Let C be the encryption of a message M using key K. The encrypted message has perfect secrecy against a plaintext attack where l message-ciphertext pairs are given if and only if*

$$H(M|C, T^l) = H(M), \tag{7.30}$$

where T^l represents l pairs of corresponding plaintext - ciphertext pairs, generated following protocol 2.

The definition of perfect secrecy states that finding the correct message is independent of acquired knowledge (of ciphertext and message-ciphertext pairs). This corresponds to the definitions of perfect secrecy given previously. However, the definition of perfect secrecy is only a useful measure of a level of secrecy if this level can be achieved in practice. For the previous definitions of prefect secrecy, examples such as the one-time pad can be given that provide this level of secrecy.

In the case of definition 10, however, it is impossible to design a system that satisfies this definition. It would mean that for the attacker, it does not make a difference whether or not he has access to these l message-ciphertext pairs and the extra ciphertext; the uncertainty of finding the correct message will not change. In section 7.2, it is stated that it is assumed that the encryption algorithm is deterministic. This means that when the attacker receives a ciphertext that corresponds to a ciphertext in the l message ciphertext pairs, he knows the corresponding plaintext. In that case no uncertainty exists about the plaintext, hence the knowledge of these l (m, c)-pairs does influence the probabilities for the plaintexts. In the case where ciphertext C is also part of an element in $\mathcal{T}^l$, knowledge of C will influence the uncertainty of the message, therefore in these cases, $P_{M|CT^l}(M|C, T^l) \neq P_M(M)$. This means that in theory, equation (7.30) cannot be fulfilled, and therefore the maximum level that can be achieved in practice is less than perfect secrecy.

To provide a measure for the level of secrecy we need a new definition that defines the maximum level of secrecy that can be achieved when plaintext attacks are taken into account. The definition of the maximum level of secrecy that can be achieved is

Definition 11 (Maximum secrecy) *Let C be the encryption of a message M using key K. The encrypted message has perfect secrecy against a plaintext attack when l message-ciphertext pairs are given if and only if*

$$H(M|C,T^l) = H(M|T^l), \tag{7.31}$$

where T^l represents l pairs of corresponding plaintext - ciphertext pairs, generated following protocol 2.

When maximum secrecy is provided, the probability of obtaining the correct message is independent of knowledge of the ciphertext, but dependent on knowledge of the l message ciphertext pairs. Because all known messages are encrypted using the same key, the number of candidates for the correct key will decrease when more of the (m,c) pairs are known. Therefore, the maximum level of secrecy that can be achieved depends on the number of known (m,c) pairs.

Definition 11 states that the ciphertext is independent of the message in case of maximum secrecy.

Note that there are situations where definition 11 is fulfilled but no secrecy is available. When the attacker has access to $|\mathcal{M}|-1$ pairs of messages and ciphertexts, there is no secrecy with respect to a new ciphertext. If this new ciphertext is part of one of the known pairs, the message is known and if this is not the case, only one message is candidate as that is the only message that is not an element of the known pairs.

In the remainder of this thesis, definition 11 will be used to determine the maximum level of secrecy taking into account plaintext attacks when only one key is used to encrypt the various messages.

7.4.2 Properties of Maximum Secrecy

This new definition of maximum secrecy obviously has implications for the requirements of a secrecy system that is to achieve this level of security. The following theorem can be derived from this definition.

Theorem 10 *A cryptosystem that provides maximum secrecy according to definition 11, will have the characteristic*

$$H(K) \geq H(M|T^l). \tag{7.32}$$

◇

Proof. Using the general properties of entropy, the following equations can be set up:

$$\begin{aligned} H(K,M,T^l,C) &= H(K|M,T^l,C) + H(M,T^l,C) && (7.33)\\ &= H(M|K,T^l,C) + H(K,T^l,C). && (7.34) \end{aligned}$$

Combining equations (7.33) and (7.34) and using the fact that $H(M|K, T^l, C) = 0$ gives

$$H(K|M, T^l, C) = H(K|T^l, C) - H(M|T^l, C). \tag{7.35}$$

The measure of information is always non-negative, therefore $H(K|M, T^l, C) \geq 0$. This results in

$$H(K|T^l, C) \geq H(M|T^l, C). \tag{7.36}$$

Because $H(K) \geq H(K|T^l, C)$, it is true that:

$$H(K) \geq H(M|T^l, C).$$

In case of perfect secrecy, $H(M|T^l, C) = H(M|T^l)$; this means that:

$$H(K) \geq H(M|T^l). \tag{7.37}$$

□

Inequality (7.32) states that when maximum secrecy is achieved, the uncertainty of the key is equal to or greater than the uncertainty of the message given l message-ciphertext pairs. It provides a property at the moment l (m, c) pairs are known but not a measure of how many keys are necessary to provide maximum secrecy, as was the case with perfect secrecy in chapter 6. One can only say that when the attacker has access to l message - ciphertext pairs and all messages that are not part of $\mathcal{T}^l$ are chosen with equal probability, then the number of candidate keys must be at least equal to the number of messages that can be chosen, e.g. $|\mathcal{M}| - l$.

Based on the proof of theorem 10 the inequality $H(K|T^l) \geq H(M|T^l)$ will hold in case maximum secrecy is provided. As $H(K|T^l) \geq H(K|T^l, C)$ and inequality (7.36) holds, this leads to

$$H(K|T^l) \geq H(M|T^l, C).$$

If maximum secrecy is provided it means that $H(K|T^l) \geq H(M|T^l)$. This inequality provides a minimum bound on the uncertainty of the key when l pairs of messages and ciphertexts are known. It provides more insight in the number of keys necessary to provide maximum secrecy. If all messages are equiprobable it means that the number of candidate keys (for the attacker) when l pairs are known to the attacker should be equal or larger than the number of possible messages at that point in time.

To derive the total number of keys needed in a secrecy system to provide maximum secrecy, one must take several observations into account. First, when a system is resistant against knowledge of at maximum l pairs, it must also be resistant against knowledge of $l - 1$ pairs and $l - 2$ pairs, etc. This is important to derive the minimum number of keys necessary to achieve maximum secrecy.

Second, it must be clear that an encryption scheme that is resistant against a plaintext attack, need more keys between one message and ciphertext pair. If this were not the case, then it would be possible to uniquely determine the key with knowledge of one plaintext-ciphertext pair as for the OTP. Each message and ciphertext is connected

by exactly one line (the key) in figure 6.2. Knowing one (m, c) pair results therefore in knowledge of the key. In the case where multiple keys can be responsible to transform one message in a particular ciphertext, still uncertainty exists on the used key, which may prevent correct decryption of a second ciphertext.

The remainder of this section is a discussion on the number of keys necessary to transform one plaintext into its corresponding ciphertext related to the number of known message-ciphertext pairs. The reason why the number of keys needs to dependend on the number of message-ciphertext pairs is that the attacker has unlimited computing power and memory storage. It must therefore be assumed that the attacker might be able to compute all the possible keys that connect one message to a particular ciphertext. Only by using multiple keys for one pair one can obtain maximum secrecy in case of a known plaintext attack.

The parameter $n(l)$ will denote the minimum number of keys between one plaintext and ciphertext pair such that it is still possible to obtain maximum secrecy when l (m, c) pairs are available to the attacker. This number depends on the number of plaintext-ciphertext pairs known to the attacker. The relation between the necessary number of keys and achieving maximum secrecy is given by the following theorem.

Theorem 11 *Let C be the encryption of message M using key K, and $|\mathcal{M}| = |\mathcal{C}|$. The probability of selecting a message is defined as follows.*

$$P_{M|T^l}(m|t_1, \ldots, t_l) = \begin{cases} 0 & \text{if } m \in \mathcal{T}^l \\ \frac{1}{|\mathcal{M}|-l} & \text{if } m \notin \mathcal{T}^l \end{cases}$$

For every c, at least one key exists to decrypt c that results in a unique message for that particular key. Before any (m, c) pairs have been made publicly available, all keys are equiprobable. The minimum number of keys that connect one (m, c)-pair is denoted by $n(l)$ such that the total number of keys is $|\mathcal{K}| = n(l)|\mathcal{M}|$. Perfect secrecy is possible if and only if

$$n(l) = \begin{cases} \prod_{i=1}^{l} (|\mathcal{M}| - i) & \text{if } 1 \leq l \leq |\mathcal{M}| - 2. \\ 1 & \text{if } l = 0. \end{cases} \tag{7.38}$$

◇

Proof.

First it is proven that when perfect secrecy is provided, the minimum number of keys connecting one (m, c)-pair is equal to equation (7.38). It must be shown that when perfect secrecy is provided, $n(l)$ should be of equal value for each (m, c)-pair. Initially, all keys are equiprobable. It means that when a message is selected, followed by a key selection; it should be possible to encrypt that message with any key resulting in a valid ciphertext. In a graph representation as was first shown in section 6.4.1, it will be the case that from each message an equal number of lines are leaving (e.g. $|\mathcal{K}|$). This implies that at each ciphertext an equal number of lines are arriving. If this were not the case, some ciphertexts would not result in a unique plaintext after decryption. Therefore, $n(l)$ is equal for each (m, c)-pair.

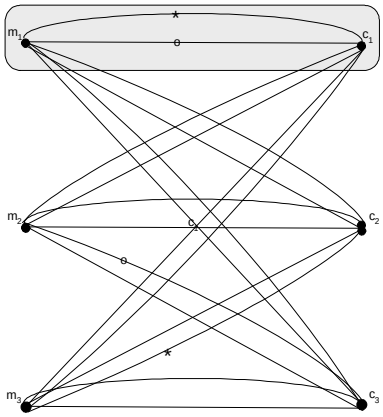

Figure 7.1: When (m_1, c_1) is a pair known to the attacker, the number of lines between that pair must be at least 2 as these lines must be divided over the two remaining ciphertexts in order to obtain perfect secrecy when $l = 1$.

In case $l = 0$, no plaintext attacks are allowed and the situation is equal to the conventional Shannon encryption scheme: $n(0) = 1$.

A system is designed to be robust against an attack where l (m, c) pairs are known. This means that the system must be capable of resisting attacks where l or less (m, c) pairs are known. Let $\lambda(\alpha)$ denote the number of candidate keys for the attacker when α (m, c) pairs are known, where $\alpha \leq l$. When $\lambda(\alpha) = 1$, there is a unique solution for the key, which means that no security is guaranteed. When $\lambda(\alpha) > 1$ for all $\alpha \leq l$, the system is resistant against a plaintext attack with l known (m, c) pairs. Now it will be proven what minimum value $\lambda(\alpha)$ must have, which is directly related to $n(l)$. It is an iterative process which starts with $\alpha = 0$ up to $\alpha = l + 1$.

Consider the situation of a system designed to be resistant against a plaintext attack where l (m, c) pairs are known. In case $\alpha = 0$, this is equal to the conventional case where no plaintext attack is taken into account, and the number of candidate keys for a good solution, $\lambda(0)$, is then $|\mathcal{K}|$. When the attacker has access to one (m, c)-pair, the number of candidate keys has been reduced to $\frac{|\mathcal{K}|}{|\mathcal{M}|}$ as only the keys that connect that particular (m, c)-pair are now possible candidates. Therefore the value for $\lambda(1)$ will be at least $\frac{|\mathcal{K}|}{|\mathcal{M}|}$ (see figure 7.1).

When the attacker has access to two (m, c) pairs and $\lambda(2) > 1$, the number of keys that were candidate for one (m, c)-pair (e.g. $\lambda(1)$) must be divided over $|\mathcal{M}| - 1$ messages as the message chosen for the first pair cannot be chosen again, hence $\frac{\lambda(1)}{|\mathcal{M}|-1}$ must be an integer. The minimal number of candidate keys for $\alpha = 2$ becomes

$$\lambda(2) = \frac{\lambda(1)}{|\mathcal{M}| - 1}.$$

This continues until $\alpha = l$, where it must still hold that $\lambda(l) > 1$ as the system is to be resistant against a plaintext attack where l (m, c) pairs are known:

$$\lambda(l) = \frac{\lambda(l-1)}{|\mathcal{M}| - (l-1)} = \frac{n(l)}{\prod_{i=1}^{l-1}(|\mathcal{M}| - i)}.$$

As the system is resistant against l (m, c) pairs, it may be the case that $\lambda(l+1) = 1$. The expression for $\lambda(l+1)$ is

$$\lambda(l+1) = \frac{\lambda(l)}{(|\mathcal{M}| - l)} = \frac{n(l)}{\prod_{i=1}^{l}(|\mathcal{M}| - i)}.$$

When $\lambda(l+1) = 1$, this means that there is only one candidate for the correct key and when $\lambda(\alpha) > 1$ for all $\alpha \leq l$, this provides the minimum number of keys between one message ciphertext pair. Hence in case of perfect secrecy,

$$n(l) = \prod_{i=1}^{l}(|\mathcal{M}| - i) \quad \text{for } 1 \leq l \leq |\mathcal{M}| - 2.$$

Parameter l can only take on values up to $|\mathcal{M}| - 2$, because this is the maximum number of (m, c) pairs allowed to be public while secrecy can still be provided to the remaining two messages.

Now it is necessary to provide proof in the opposite direction.

The following provides proof that when $n(l)$ is given by equation (7.38) and the basic conditions of the theorem are met, perfect secrecy is provided, therefore $P_{M|T^l}(m|t_1, \ldots, t_l) = P_{M|CT^l}(m|c, t_1, \ldots, t_l)$. As is defined in the theorem $P_{M|T^l}(m|t_1, \ldots, t_l) = \frac{1}{|\mathcal{M}|-l}$ when m is not part of an element in $\mathcal{T}^l$. The conditional probability $P_{M|CT^l}(m|c, t_1, \ldots, t_l)$ is equal to the probability of a key k such that $D_k(c) = m$ when $t_1, \ldots, t_l$ are given. As there are $\lambda(l)$ candidates for the key and l known (m, c) pairs only one of these candidates will fulfill this condition for this new c that is given, hence

$$P_{M|CT^l}(m|c, t_1, \ldots, t_l) = \frac{1}{\lambda(l)}. \tag{7.39}$$

When the number of keys between one (m, c)-pair is given by $n(l)$ and all keys are divided equally, then $\lambda(\alpha)$ can be computed as follows: $\lambda(1) = n(l)$ as there are $n(l)$ candidates for the correct key when one (m, c)-pair is given. $\lambda(2)$ is then equal to the number of candidates when $\alpha = 1$, divided over the remaining number of messages.

$$\lambda(2) = \frac{\lambda(1)}{(|\mathcal{M}| - 1)}.$$

Following this reasoning, $\lambda(l)$ can be determined and is expressed as

$$\lambda(l) = \frac{n(l)}{\prod_{i=1}^{l-1}(|\mathcal{M}| - i)}. \tag{7.40}$$

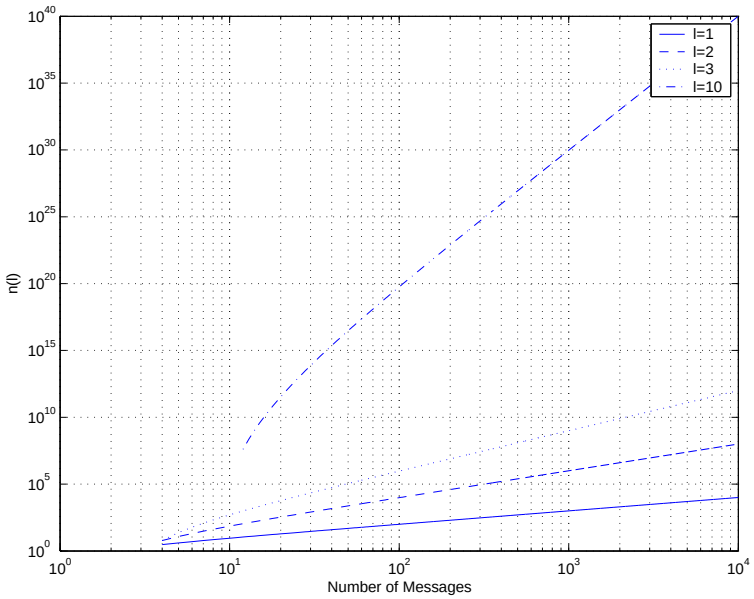

Figure 7.2: The number of keys per (m, c) pair necessary related to message length and known (m, c) pairs.

Substituting (7.40) in (7.39) leads to

$$P_{M|CT^l}(m|c, t_1, \ldots, t_l) = \frac{\prod_{i=1}^{l-1}(|\mathcal{M}|i)}{\prod_{i=1}^{l}(|\mathcal{M}| - i)} = \frac{1}{|\mathcal{M}| - l},$$

which is equal to $P_{M|T^l}(m|t_1, \ldots, t_l)$, hence perfect secrecy is provided. □

Figure 7.2 shows the relation between the number of messages and the minimum number of keys necessary to connect one (m, c)-pair when perfect secrecy is required. The value of parameter l is chosen relatively small to demonstrate the enormous amount of keys needed. Even when l is small (e.g. $l = 3$) and the number of messages is equal to a hundred, the number of keys between one message and ciphertext is already a factor 10^4 larger than the size of the message space. The one-time pad is not often used in practice because the key length must be equal to the message length. Here, the key space must be even larger than the message length, and therefore this type of system is not practical, even though it provides a high level of secrecy.

In the previous sections we always assumed that when a system is designed such that maximum secrecy is achieved when an attacker has access to l message ciphertext pairs, this attacker will have access to l (m, c) pairs. However, the case where an attacker has access to α ($\alpha \leq l$) pairs of (m, c) when the system is designed for a maximum of l (m, c) pairs is interesting too.

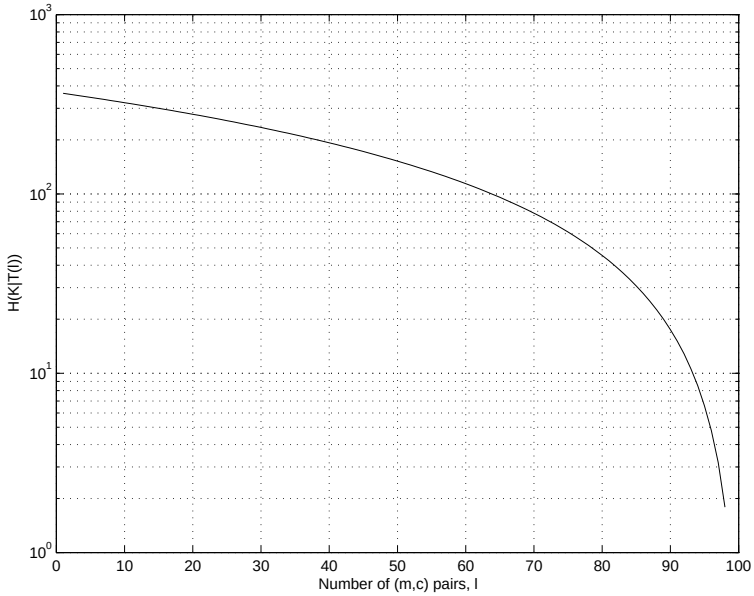

Figure 7.3: The relation between $H(K|T^l)$ and the number of (m, c) pairs.

Figure 7.3 gives the relation between the entropy $H(K|T^l)$ and the number of message-ciphertext pairs that are known to the attacker. The cardinality of the message space is equal to 100 and the total number of keys in the system is set tot the level where maximum secrecy is achieved when l has a maximum value. In this case, the maximum value of l is 98 and therefore the total number of keys is $|\mathcal{M}| \prod_{i=1}^{l} (|\mathcal{M}| - i)$.

The uncertainty of the key given by $H(K|T^\alpha)$ represents the uncertainty of the key when the number of message - ciphertext - pairs known to the attacker is α ($\alpha \leq l$).

All keys are chosen at random; hence the uncertainty of the key given α (m, c) pairs is given by

$$H(K|T^\alpha) = \log(\lambda(\alpha)),$$

where $\lambda(\alpha)$ represents the number of candidate keys when α (m, c) pairs are known.

When $\alpha = 0$, the number of candidate keys is equal to the total number of keys $|\mathcal{K}|$. When $\alpha = 1$, $\lambda(\alpha)$ is reduced to $\frac{|\mathcal{K}|}{|\mathcal{M}|}$ as was shown in the proof of theorem 11. When 99 pairs of (m, c) are known to the attacker, no uncertainty exists about the message when a ciphertext is given. Note that the messages and ciphertexts in these pairs are of equal length to the ciphertext of which the attacker wishes to obtain the corresponding plaintext.

7.4.3 Unicity Distance

Similar to section 7.3.3, it is possible to determine the unicity distance if the system is designed to be resistant against plaintext attacks and the publicly known (m, c) pairs have been generated using the same key. It is assumed that the cardinality of the set of plaintext symbols is equal to that of ciphertext symbols. The unicity distance is defined according to definition 9.

Theorem 12 *Let (M, C, K) denote a symmetric encryption scheme, where $E_k(m) = c$. An attacker knows l message-ciphertext pairs, generated by executing protocol 2. Then the unicity distance is given by*

$$UD \approx \frac{H(K|T^l)}{R_{lang} \log |\mathcal{A}|}, \tag{7.41}$$

where $l \ll |\mathcal{M}(L)|$ and R_{lang} represents the redundancy of the language and $\mathcal{A}$ is the set of symbols used to construct a message. ◇

Proof.

The proof of this theorem is analogous to the proof of theorem 9, but because the knowledge of the attacker is based on the execution of protocol 2 instead of protocol 1, several aspects of the proof are not trivial and are pointed out here.

As in the proof of theorem 9, the number of spurious keys is equal to $\mu(c, l) - 1$, where $\mu(c, l)$ provides the number of candidate keys given a certain ciphertext and l pairs of (m, c). Equation (7.14) therefore also holds in this case.

Equations (7.16) and (7.17) are based on general characteristics of information theory and are valid in all cases. Furthermore, $H(M^L|K, C^L, T^l)$ is said to be equal to zero, which also holds in this case as knowledge of the ciphertext and key leads to the corresponding message, hence

$$H(K|C^L, T^l) = H(M^L|K, T^l) + H(K, T^l) - H(C^L, T^l). \tag{7.42}$$

Protocol 2 describes how the l pairs of (m, c) are generated. The messages are selected at random with the condition that they should not already be in the set $\mathcal{T}^l$. These messages are chosen independent from the key, hence $H(M^L|K, T^l) = H(M^L|T^l)$. Equation (7.42) can then be written as:

$$H(K|C^L, T^l) = H(M^L|T^l) + H(K|T^l) - H(C^L|T^l). \tag{7.43}$$

The upperbound for $H(C^L|T^l)$ is $L \log |\mathcal{A}|$ where $\mathcal{A}$ represents the cardinality of the set of symbols for the plaintext, but this is equal to the cardinality of the set of symbols for the ciphertext. Hence,

$$H(K|C^L, T^l) \geq H(M^L|T^l) + H(K|T^l) - L \log |\mathcal{A}|. \tag{7.44}$$

The entropy $H(M(L|T^l)$ can be estimated. To compute the entropy of one symbol of plaintext the language statistics were taken into account. However, the new plaintext is chosen according to protocol 2, which states that only messages can be chosen

that are not part of T^l. Only when $l \ll |\mathcal{M}(L)|$, the number of known messages will not influence the uncertainty of M^L. Then $H(M^L|T^l) = H(M^L)$ and therefore $H(M^L|T^l) = L(1 - R_{lang} \log |\mathcal{A}|$. Therefore,

$$H(K|C^L, T^l) \geq H(K|T^l) - LR_{lang}) \log |\mathcal{A}|. \tag{7.45}$$

The remaining of the proof is equal to the one of theorem 9. The average number of spurious keys is:

$$\bar{s}_L(l) \geq \frac{2^{H(K|T^l)}}{|\mathcal{A}|^{LR_{lang}}} - 1. \tag{7.46}$$

The unicity distance is the minimum length of the ciphertext when $\bar{s}_L(l) = 0$, hence

$$UD \approx \frac{H(K|T^l)}{R_{lang} \log |\mathcal{A}|}, \tag{7.47}$$

□

The unicity distance is the minimal number of ciphertext symbols such that the entropy $H(K|T^l)$ equals zero. It provides a measure of the number of symbols that can be encrypted using the same key such that secrecy still can be provided. In the case where plaintext attacks are taken into account, the attacker has access to l (m, c) pairs. When the ciphertext of which he does not know the corresponding plaintext is of length of unicity distance, and therefore all known messages and corresponding ciphertexts are of this length too, it means that based on this information it is on average possible to uniquely determine the correct key.

In general, in case of ciphertext-only attacks the unicity distance will be larger than in case of plaintext attacks as in the latter, the attacker does not only have information about one cipher text but also about l (m, c) pairs.

7.4.4 Conclusions

This section considered the case where the attacker has access to l message ciphertext pairs and each ciphertext is generated using the same key. A definition of perfect secrecy was given. However, it is not possible to achieve this level of uncertainty in practice. Therefore, we defined the maximum level of secrecy.

Based on this definition of maximum secrecy, several properties could be derived. First, when maximum secrecy is provided, $H(K) \geq H(M|T^l)$. Second, because the assumption is made that each ciphertext is generated based on the same key, the number of keys that relate a particular message to a particular ciphertext must be larger than 1. Finally, when maximum secrecy is achieved, it is possible to derive the minimum cardinality of the key space. However, the necessary number of keys is so large that in practice it will not be feasible to achieve the level of maximum secrecy either.

Finally, an expression for the unicity distance was given. This distance provides a good measure of how many ciphertexts can be encrypted using the same key while still being able to provide confidentiality.

7.5 Conclusions

This chapter gave several results on the requirements for a secrecy system that must be resistant against a known plaintext attack. Two types of plaintext attacks were distinguished. The first one is the type where the known message-ciphertext pairs are generated using different keys, followed by the type where these pairs are generated using only one key.

For the first type, it is theoretically possible to achieve perfect secrecy. However, this is not possible for the second type and for this therefore a maximum level of secrecy was defined.

Based on perfect secrecy (first type) and maximum secrecy (second type), several properties of such systems can be derived of which the minimal number of keys is an important one. Furthermore, we obtained the unicity distance for these two types.

In the next chapter, the theory derived here will be applied to the mobile code privacy model.

Chapter 8

Information Theoretic Approach for Mobile Code Privacy Model

This chapter uses the theory of chapters 6 and 7 to derive theoretical boundaries of the level of privacy that can be provided in the mobile code privacy model of section 2.3. Terms like perfect secrecy, maximum level of secrecy and cryptographic dilemma are used for this purpose.

8.1 Introduction

Section 2.3 described the mobile code privacy model. This model will be used to apply the information theoretic concepts of the previous chapters. An essential characteristic of the model is the assumption that the attacker has access to unlimited computing resources and time. Furthermore, mobile code privacy is seen as providing confidentiality to mobile code by means of encryption. Mobile code is represented by a function F. The function F is encrypted using a key K; $G = E_K(F)$. The encrypted function G is then executed with parameter X as input; $U = G(X)$.

Two types of attacks are considered: ciphertext-only attacks and plaintext attacks. These cases will be treated separately.

This chapter is organized as follows. First an information theoretic approach is taken to derive the theoretical boundaries of mobile code protection against ciphertext-only attacks. This is followed by a similar approach for plaintext attacks. Finally, conclusions are given.

8.2 Ciphertext-only Protection

With ciphertext-only attacks the attacker only has access to ciphertext. The attacker wants to derive the corresponding plaintext or the key that was used to encrypt the message. The plain function F is encrypted using key K, resulting in cipher function G: $E_K(F) = G$. The attacker has access to cipher function G, input X and output U.

In the next sections, a definition will be given of perfect secrecy, its properties and the cryptographic dilemma with respect to the mobile code privacy model. The results with respect to ciphertext-only attacks were also published in [22].

8.2.1 Perfect Secrecy

Section 6.3 gave Shannon's definition of perfect secrecy, which was suitable for ciphertext-only attacks, which is the case here. Perfect secrecy is achieved when knowledge of the ciphertext does not give the attacker an advantage over an attacker without such access. A similar definition of perfect secrecy can be given in the case of mobile code. However, in the mobile code privacy model, the attacker may have access to the cipher function, but also to some input parameter of the cipher function and the corresponding output. Perfect secrecy is then defined as follows.

Definition 12 (Perfect secrecy mobile code) *Let G be the encryption of mobile code F using key K. The result of G, using input X, after execution is U. The encrypted mobile code has perfect secrecy if and only if*

$$H(F|G, X, U) = H(F). \tag{8.1}$$

It is important that after encryption the code is still executable, as the original intention is to execute code at a remote location. Therefore, the function G should be of such a form that when given an input value X an output value, U, can be computed, e.g. the symbols of G should form a function again. Based on this definition of perfect secrecy, properties can be derived to provide perfect secrecy.

Theorem 13 *A mobile code cryptosystem that provides perfect secrecy will have the characteristics*

$$H(K) \geq H(F) \text{ and} \tag{8.2}$$
$$H(K) \geq H(U|X). \tag{8.3}$$

Proof.

To prove inequality (8.2), the joint entropy $H(K, F, G, X, U)$ can be written as:

$$H(K, F, G, X, U) = H(K|F, G, X, U) + H(F, G, X, U) \tag{8.4}$$
$$= H(F|K, G, X, U) + H(K, G, X, U). \tag{8.5}$$

Combining (8.4) and (8.5), rewriting the joint entropies, and using the fact that when K and G are known, F can be computed (e.g. $H(F|K, G, X, U) = 0$) gives

$$H(K|F, G, X, U) = H(K|G, X, U) - H(F|G, X, U). \tag{8.6}$$

Since entropy is non-negative, $H(K|F,G,X,U) \geq 0$. Substituting this in (8.6) and noting that $H(A) \geq H(A|B)$ and thus $H(K) \geq H(K|G,X,U)$ results in

$$H(K) \geq H(F|G,X,U). \tag{8.7}$$

Using the definition of perfect secrecy (8.1), equation (8.7) evolves in inequality (8.2).

For the second inequality (8.3), the following equality can be used:

$$H(K|F,X,U) = H(K|F,X) - H(U|F,X), \tag{8.8}$$

which can be derived in a similar manner as (8.6), but taking into account that $H(U|K,F,X) = 0$. Due to the non-negative property of entropy, $H(K|F,X,U) \geq 0$. After substituting this into (8.8), it is clear that

$$H(K|F,X) \geq H(U|F,X), \tag{8.9}$$

and in general it holds that $H(K) \geq H(K|F)$. Then (8.9) evolves in

$$H(K) \geq H(U|F,X). \tag{8.10}$$

The joint entropy $H(F,X,U)$ can be written as

$$\begin{aligned} H(F,X,U) &= H(U|F,X) + H(F|X) + H(X) && (8.11)\\ &= H(F|U,X) + H(U|X) + H(X), && (8.12) \end{aligned}$$

which results in

$$H(U|F,X) + H(F|X) = H(U|X) + H(F|U,X).$$

In case of perfect secrecy, $H(F|G,X,U) = H(F)$, therefore

$$H(F|U,X) = H(F|X) = H(F),$$

hence

$$H(U|F,X) = H(U|X). \tag{8.13}$$

Substituting (8.13) in (8.10) completes the proof. □

Theorem 13 can be interpreted as follows. When the occurrence of all functions is equiprobable, a maximum for $H(F)$ is reached, e.g. $H(F) = \log|\mathcal{F}|$. In order to achieve perfect secrecy, it is among others necessary for the key space to be at least as large as the function space.

The second property ($H(K) \geq H(U|X)$) gives a condition on the average uncertainty of the key with respect to the output given a certain input. When U and X are completely independent ($H(U|X) = H(U)$) and all possible outputs are equiprobable, the key space must be as large as or larger than the output space $\mathcal{U}$. In most case this situation where $H(U|X) = H(U)$ is likely to occur as one value of X will not provide information on the value of U. Only combined with information about G, U can be determined ($H(U|G,X) = 0$).

8.2.2 Example of Perfect Secrecy.

This section gives an example of an encryption scheme that provides perfect secrecy to a function. In this example, the set of functions consists of polynomials. A function is constructed by selecting elements from different sets, and these are combined in a sequence of which the order of the elements obeys some rules. For example, a plus operator will always be followed by a number.

The following encryption scheme provides perfect secrecy [22].

Definition 13 (One-time pad for polynomials) *(OTPP) Three sets are defined:*

- *Numbers:* $N := \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$,
- *Operators:* $O := \{+, \times, -, \div\}$,
- *Variables:* $V := \{x\}$.

A function f is defined as a combination of elements of the three sets. The encryption of f is given by

$$E_k(f) = f \circ k, \tag{8.14}$$

where $\circ$ can be seen as addition defined over the various alphabets. The key is chosen according to the symbols of the function. If the symbol at position j is an element of the set N, then the key at position j will also be an element of the set N. The encryption operation for set N is equal to addition $\mathrm{mod}10$. In set O, the encryption operation $\circ$ is defined as:

$$\begin{array}{llll} + \circ + = \times & ; & - \circ - = \div \\ + \circ \times = + & ; & - \circ \div = - \\ \times \circ + = + & ; & \div \circ - = - \\ \times \circ \times = \times & ; & \div \circ \div = \div. \end{array}$$

The encryption scheme OTPP can be used as follows. Let $f(x)$ be $f(x) = 7x^2 + 4x + 3$; then encryption and decryption are respectively:

$$\begin{aligned} f(x) &: 7 \times x \times x + 4 \times x + 3 \\ k(x) &: 2 + x \times x \times 3 + x + 2 \\ g(x) &: \overline{9 + x \times x + 7 + x \times 5} \end{aligned}$$

and

$$\begin{aligned} g(x) &: 9 + x \times x + 7 + x \times 5 \\ k(x) &: 2 + x \times x \times 3 + x + 2 \\ f(x) &: \overline{7 \times x \times x + 4 \times x + 3} \end{aligned}$$

In the above example it is clear that $g(x)$ can be executed.

Theorem 14 *The encryption scheme OTPP provides perfect secrecy for mobile code if the mobile code is a (set of) polynomial(s).*

Proof. To prove perfect secrecy equation (8.1) must hold. This is equivalent to

$$P_{F|G,X,U}(f|g,x,u) = P_F(f). \tag{8.15}$$

To compute the probability $P_{F|G,X,U}(f|g,x,u)$ it is important to note that knowledge of (x,u) does not contribute extra knowledge to the knowledge of g as (x,u) is a point on the curve of g. Knowledge of g does not provide knowledge on all points. Therefore $P_{F|G,X,U}(f|g,x,u) = P_{F|G}(f|g)$. The probability $P_{F|G}(f|g)$ is determined by the number of keys that decrypts g with result of f. Therefore, the probability $P_{F|G,U}(f|g,u)$ can be expressed as:

$$P_{F|G,X,U}(f|g,x,u) = \sum_{k\in\mathcal{K}} P_K(k)P_F(d_k(g) = f), \tag{8.16}$$

where $|K|$ is the cardinality of K and $d_k(g)$ denotes the decryption of g using key k. For each key, there is an equal probability of occurrence $\frac{1}{|K|}$:

$$P_{F|G,X,U}(f|g,x,u) = \frac{1}{|K|}\sum_{k\in\mathcal{K}} P_F(d_k(g) = f). \tag{8.17}$$

The probability $P_F(d_k(g) = f)$ can only take the values 1 or 0. Either the key k is responsible for the transformation of f into g or not. As the transformations are uniquely defined, only one key per transformation is possible, hence

$$\sum_{k\in\mathcal{K}} P_F(d_k(g) = f) = 1. \tag{8.18}$$

This leads to

$$P_{F|G,X,U}(f|g,x,u) = \frac{1}{|K|} = P_F(f). \tag{8.19}$$

Concluding, the above system is perfectly secure with respect to f. □

It must be said that it is easy for the attacker to know what type of function has been encrypted (e.g. whether it is a polynomial and of what degree), but even when this knowledge is available, perfect secrecy according to definition 12 is still provided to the mobile code.

Knowledge like that described above is comparable to that obtained when the OTP is used. There, an attacker can know the length of the message from the length of the ciphertext. An easy method to prevent the attacker from obtaining this knowledge is by adding words or plaintext to the text to be encrypted. In mobile code, this can be done by adding auxiliary polynomials to the polynomials to be encrypted as is shown in [77].

8.2.3 The Mobile Code Dilemma

Chapter 6 described the cryptographic dilemma, which relates message, key and key appearance equivocation. In the case of mobile code, various important equivocations

can be related to each other. The relevant equivocations in mobile code are $H(K|G)$, $H(K|F,G)$, $H(F|G)$ and the conditional entropies related to the extra knowledge about U; e.g. $H(K|F,U)$, $H(F|K,U)$, $H(K|U)$ and $H(F|U)$. These equivocations are related such that they result in two dilemmas.

By means of equivocations one can show that text cannot be equally be protected against all possible attacks at the same time. This results in two dilemmas.

Theorem 15 (Cryptographic dilemmas for mobile code) *Let $(F, G, K, E_K(.), D_K(.))$ denote a mobile code system protected by encryption, $E_K(.)$, and decryption, $D_K(.)$ with key K. Then the following dilemmas exist:*

$$H(K|F,G) = H(K|G) - H(F|G) \quad (8.20)$$

$$H(K|F,U) = H(F|K,U) + H(K|U) - H(F|U). \quad (8.21)$$

Proof. To prove (8.20), $H(K,F,G)$ can be written as

$$H(K,F,G) = H(K|F,G) + H(F,G) = H(F|K,G) + H(K,G). \quad (8.22)$$

Using the property $H(F|K,G) = 0$, (8.22) evolves in:

$$H(K|F,G) = H(K,G) - H(F,G). \quad (8.23)$$

Rewriting the joint entropies results in (8.20).

The proof of (8.21) is similar. The entropy $H(K,F,U)$ can be written as

$$H(K,F,U) = H(K|F,U) + H(F,U) = H(F|K,U) + H(K,U). \quad (8.24)$$

Equation (8.24) can be rewritten as

$$H(K|F,U) = H(F|K,U) + H(K,U) - H(F,U). \quad (8.25)$$

After rewriting of the joint entropies, (8.25) results in (8.21). □

The first dilemma (8.20) is equal to Shannon's original approach and states that increasing the protection of the system against a known-plaintext attack ($H(K|F,G)$ will increase) will result in a decrease of protection against a ciphertext-only-attack ($H(F|G)$ will decrease), when $H(K|G)$ remains constant. The second dilemma (8.21) combines the knowledge on U and the code to be protected F and key K. Increasing the uncertainty of the key when F and U are known immediately results in the decrease of the entropy of F when U is known, when $H(F|K,U) + H(K|U)$ remains constant.

8.2.4 Conclusions

This section applied the information theoretic concepts provided in chapter 6 to the mobile code privacy model. Only the case of ciphertext-only attacks was considered. Concepts similar to perfect secrecy and the cryptographic dilemma could be defined for mobile code. The main difference of mobile code encryption from conventional

data encryption is that the input parameter X and U may contribute to knowledge on the original function.This is the reason why perfect secrecy for mobile code has two properties and why the cryptographic dilemma is given by two equalities.

In the next section, similar concepts will be derived but for a system to be protected against plaintext attacks.

8.3 Plaintext Attacks and Mobile Code

Within the mobile code privacy model, plaintext attacks cannot be completely prevented and therefore they must be taken into consideration. They influence the theoretical boundaries for mobile code protection. The results are also published in [23].

Chapter 7 described the information theoretic approach for secrecy systems considering plaintext attacks. A distinction was made between plaintext attacks based on different keys and attacks based on one key. In practice, the latter is more interesting as it provides information on how many times a certain key can be used while a level of secrecy is guaranteed. For this reason, this is the only case that will be considered to derive theoretical boundaries for the protection of mobile code. The method to generate the l pairs of functions and corresponding cipher functions is, therefore, equal to protocol 2 on page 88. Message m is replaced by function f and ciphertext c by cipher function g. The length of the functions and cipher functions is equal. Then the following protocol is used to generate a set of l pairs of (f, g).

Protocol 3 *1. Select a key $k_j \in \mathcal{K}$ at random.*

(a) Select a message $f_i \in \mathcal{F}$ such that f_i is not an element of the (f, g) pairs in $\mathcal{T}^{i-1}$.

(b) Encrypt message m_i using key k_j: $g_h = E_{k_j}(f_i)$.

(c) The pair (f_i, g_h) is added to the set of pairs, resulting in $\mathcal{T}^i$.

The steps (a) - (c) are repeated for $i = 1, 2, \ldots, l$.

In section 7.4 it was shown that perfect secrecy cannot be provided if text is protected against plaintext attacks using one key. Therefore the definition of maximum secrecy was proposed. This concept of maximum secrecy will be used here for mobile code.

This definition leads to a number of properties a mobile code system needs to provide maximum secrecy.

8.3.1 Maximum Secrecy

The following definition of maximum secrecy for mobile code is proposed, where the attacker has access to l pairs of functions and cipher functions. One pair of function and cipher function is denoted by (f, g); l pairs of (f, g) form the set $\mathcal{T}^l$.

Definition 14 (Maximum secrecy) *Let G be the encryption of mobile code F using key K. The result of G with input X after execution is U. The encrypted mobile code*

has perfect secrecy, in the sense that it is resistant against a plaintext attack where l (f, g) pairs are known to the attacker, according to protocol 3, if and only if

$$H(F|G, X, U, T^l) = H(F|T^l). \tag{8.26}$$

◇

This definition states that maximum secrecy is provided if the probability to obtain the correct function F is independent of knowledge about its encrypted version G and the output and input of G. Informally, when the attacker has access to G and U this will not give him better chances of finding the correct F. The only parameters that may provide an advantage are the l (f, g) pairs (e.g. T^l). In equation (8.26) the term $H(F|T^l)$ is used for the same reason as $H(M|T^l)$ in definition 11 (chapter 7) $H(M|T^l)$.

When an encryption algorithm provides maximum secrecy according to definition 14, the system will have specific properties.

Theorem 16 *A mobile code cryptosystem that provides maximum secrecy has the following characteristics*

$$H(K) \geq H(F|T^l) \text{ and} \tag{8.27}$$
$$H(K) \geq H(U|X, T^l). \tag{8.28}$$

◇

Proof.

$$H(K, F, G, T^l) = H(K|F, G, T^l) + H(F, G, T^l) \tag{8.29}$$
$$= H(F|K, G, T^l) + H(K, G, T^l) \tag{8.30}$$

where $H(F|K, G, T^l) = 0$ as G and K uniquely determine F. Combining (8.29) and (8.30) leads to

$$H(K|F, G, T^l) = H(K|G, T^l) - H(F|G, T^l). \tag{8.31}$$

As entropies are non-negative, $H(K|F, G, T^l) \geq 0$, hence

$$H(K|G, T^l) \geq H(F|G, T^l). \tag{8.32}$$

Using $H(K) \geq H(K|G, T^l)$ leads to

$$H(K) \geq H(F|G, T^l). \tag{8.33}$$

The definition of maximum secrecy is that $H(F|G, X, Y, T^l) = H(F|T^l)$. This also means that $H(F|G, T^l) = H(F|T^l)$ because $H(F|G, T^l) \geq H(F|G, X, U, T^l)$, but $H(F|G, T^l) \leq H(F|T^l)$. As the cryptosystem provides maximum secrecy, equation (8.33) leads to

$$H(K) \geq H(F|T^l). \tag{8.34}$$

To prove the second inequality we give an equivalent proof, but based on the equation

$$H(K|F, X, U, T^l) = H(K|F, X, T^l) - H(U|F, X, T^l). \tag{8.35}$$

As entropies are non-negative and $H(K) \geq H(K|F, X, T^l)$, it follows that

$$H(K) \geq H(U|F, X, T^l). \tag{8.36}$$

$H(F, X, U, T^l)$ can be written as

$$\begin{aligned} H(F, X, U, T^l) &= H(U|F, X, T^l) + H(F, X, T^l) && (8.37)\\ &= H(F|X, U, T^l) + H(X, U, T^l) && (8.38) \end{aligned}$$

Combining (8.37) and (8.38) gives

$$H(U|F, X, T^l) + H(F|X, T^l) = H(F|X, U, T^l) + H(U|X, T^l).$$

In general it holds that $H(F|X, T^l) \geq H(F|X, U, T^l)$ and $H(F|X, T^l) \leq H(F|T^l)$, but because in the case of maximum secrecy $H(F|X, U, T^l) = H(F|T^l)$, it holds that $H(F|X, U, T^l) = H(F|X, T^l)$. Then it follows that

$$H(U|F, X, T^l) = H(U|X, T^l). \tag{8.39}$$

Substitution of (8.39) in (8.36) gives (8.28) □

The inequalities of theorem 16 provide minimum conditions to provide perfect secrecy, although when these conditions are fulfilled, perfect secrecy is not necessarily provided. Note that both conditions provide a minimum uncertainty of the key when l pairs of f and g are known to the attacker.

If maximum secrecy is provided it will hold that $H(K) \geq H(F|T^l)$. It does not mean that the minimum amount of keys necessary to provide maximum secrecy is equal to $|\mathcal{F}|$. From equation (8.32) it can be seen that

$$H(K|T^l) \geq H(F|G, T^l), \tag{8.40}$$

because $H(K|T^l) \geq H(K|G, T^l)$. If maximum secrecy is provided $H(F|G, X, U, T^l)$ equals $H(F|T^l)$ (see equation (8.26)), therefore $H(F|G, T^l) = H(F|T^l)$ if maximum secrecy is provided. Hence, it will hold that $H(K|T^l) \geq H(F|T^l)$. Thus, if maximum secrecy is provided the cryptosystem will have the characteristic that $H(K|T^l) \geq H(F|T^l)$. It means that at any time an observer or attacker observes the system the uncertainty of the key should be equal or larger than the uncertainty with respect to the original function. If all functions are equiprobable it means that the number of candidate keys when l pairs are known should be equal or larger than the number of possible functions.

An equivalent interpretation holds for the second inequality in theorem 16. The uncertainty of the output U given the corresponding X value and l (f, g) pairs, provides a lower bound for the uncertainty of the key. Again this condition provides the required status for the moment when l (f, g) pairs are known. The condition does not directly provide the minimum number of keys, however, it can be derived similar to the list for the first condition (given that all outputs are equiprobable).

Based on the assumption that each function has an equal probability of occurring, it is possible to derive the minimal number of keys necessary to protect the mobile code against knowledge of l (f, g) pairs.

8.4 Conclusions

This chapter defined perfect secrecy for mobile code. The definition is similar to Shannon's definition of perfect secrecy for static data, but the context is different as the attacker can be located in the execution environment. Based on this definition of perfect secrecy it is possible to derive conditions under which perfect secrecy is possible. A condition to provide perfect secrecy is that the key space should be at least as large as the function space and the output space. Furthermore, an example encryption scheme was given that provides perfect secrecy.

Using the properties of entropy, it is possible to derive two cryptographic dilemmas, where increasing the uncertainty of one parameter decreases the entropy of a different parameter.

Furthermore, this information theoretic approach was extended with the approach of chapter 7, where plaintext attacks are taken into account. This led to a definition of maximum secrecy and to minimal conditions under which perfect secrecy can be provided.

Chapter 9

Unicity Distance in Mobile Code

In this chapter, the unicity distance is derived for the situation where mobile code must be protected against ciphertext-only and plaintext attacks. These cases are treated separately.

9.1 Introduction

In chapters 6 and 7 we derived expressions for the unicity distance in the situations of ciphertext-only and plaintext attacks for classical secrecy systems. For mobile code we can also derive an expression for the unicity distance. The model of figure 2.3 is used to define and to derive the expression for the unicity distance.

At this stage, it is not yet investigated what cipher data contributes to the unicity distance. The function F is encrypted based on key K resulting in G, therefore G is cipherdata that is relevant for the unicity distance. However, the output values U may also contribute to the unicity distane. In this chapter it is analyzed whether this is the case.

The outline of this chapter is as follows. First, the unicity distance is defined for the application to mobile code. This is followed by a discussion in section 9.3 on what data contributes to the unicity distance in the mobile code privacy model. Section 9.4 derives the unicity distance in the case where ciphertext-only attacks are taken into account; section 9.5 gives it for plaintext attacks. Finally, in section 9.6, conclusions are given.

9.2 Defi nition Unicity Distance for Mobile Code

It is assumed (as in figure 2.3) that the relation between the function F and cipher function G is given by $E_K(F) = G$ and $G(X) = U$. The following definition

provides a formal definition for the unicity distance for mobile code.

Definition 15 (Unicity distance mobile code (ciphertext-only attacks)) *Let (F^L, G^L, K, X, U) be a mobile code system where $E_K(F^L) = G^L$ is the encryption of function F (of length L) based on key K resulting in cipherfunction G^L (of length L), and $G(X) = U$. The unicity distance is defined as*

$$UD = \min\{L \in \mathbb{N} | H(K|G^L) = 0\}.$$

◇

If the attacker has access to a cipherfunction of length UD, on average he will be able to compute the key (given unlimited computation time). The unicity distance does not provide a guarantee that the attacker will succeed in obtaining the key as at unicity distance the equivocation $H(K|G^L)$ equals to zero and this is an average value.

The definition for unicity distance in case of plaintext attacks for mobile code is then given by the following definition.

Definition 16 (Unicity distance mobile code (plaintext attacks)) *Let (F^L, G^L, K, X, U) be a mobile code system where $E_K(F^L) = G^L$ is the encryption of function F (of length L) based on key K resulting in cipherfunction G^L (of length L). Let T^l denote l pairs of functions and corresponding cipherfuntions. The unicity distance is defined as*

$$UD = \min\{L \in \mathbb{N} | H(K|G^L, T^l) = 0\}.$$

◇

In conventional cryptosystems, the attacker only has access to ciphertext, and the unicity distance is determined by the number of symbols of this ciphertext. In the mobile code privacy model, however, the attacker has access to more types of data, e.g. the cipher function, and input-output pairs of this function. This chapter analyzes whether the case of message encryption or function encryption leads to a different expression for the unicity distance.

In the next sections the unicity distance is derived for the case of ciphertext-only and plaintext attacks. Shannon derived the unicity distance based on the random cipher model (section 6.4.1). If the unicity distance is derived for a particular cipher, it is reasonable to assume that formulas derived for the random cipher may be applied in such particular cases. However, in some cases it is necessary to apply certain corrections, such as when letter frequencies are preserved in the ciphertexts [83]. In this chapter, these corrections for particular ciphers are not taken into account, as here the general derivation for unicity distance is given and not for one particular algorithm.

9.3 Unicity distance for mobile code

The unicity distance is expressed in terms of length of ciphertext. Dependent on what parameters form the cipher text, the unicity distance can be derived. Possible relevant parameters are the cipherfunction and the results of the executed cipherfunction.

Decisions based on the outcome, U, of the encrypted function, can either be made directly from U or via a decryption of U to Y such that $Y = F(X)$. Access to multiple (X, Y)-coordinates may lead to reconstruction of F (section 2.3.1). This could possibly influence the unicity distance.

The attacker has access to the encrypted function G and can execute the function based on the choice of X. Knowledge of G leads to knowledge on the output of the function, e.g. $H(U|G, X) = 0$. Since the joint entropy $H(G, X, U)$ can be expressed as

$$H(G, X, U) = H(G, X) + H(U|G, X),$$

it follows that $H(G, X, U) = H(G, X)$. To express the influence of different parameters or knowledge of the key, we can express $H(K, G, X)$ as

$$H(K, G, X) = H(K, G, X) + H(U|K, G, X) = H(K, G, X, U),$$

because $H(U|K, G, X) = H(U|G, X) = 0$. Rewriting the expressions for the joint entropies leads to

$$H(K|G, X) + H(G, X) = H(K|G, X, U) + H(G, X),$$

and therefore $H(K|G, X) = H(K|G, X, U)$. This means that knowledge of U does not contribute to knowledge of the key once the function G is known. Therefore, for the unicity distance only the cipher function G influences the uncertainty of the key.

The unicity distance is only determined by the cipher function G and therefore it is not relevant whether decisions can be made on U or whether first a decryption is necessary. Furthermore, as only G contributes to the unicity distance, this situation is equivalent to the one of classical cipher systems (chapter 6). The function G may consist of symbols from a different alphabet compared to classical ciphertext, but this is irrelevant for the derivation of the unicity distance (although the redundancy in the language is relevant). For the case of ciphertext-only attacks the same reasoning holds and will therefore be equal to the unicity distance as was given in section 7.4.3.

As the U-values do not contribute to the value of the unicity distance, the unicity distance for mobile code will be equal to the distance for classical secrecy systems. Although it is debatable whether these expressions for the unicity distance should be considered as theorems or corollaries, we present them as separate theorems to emphasize the semantic difference between a classical secrecy system and the mobile code privacy model.

9.4 Unicity Distance in the Case of Ciphertext-only Attacks

The following theorem gives the unicity distance for mobile code in the case of ciphertext-only attacks.

Theorem 17 *Let $(F, G, K, X, Y, U, E_K(.), D_K(.))$ denote a mobile code system protected by encryption, $E_K(.)$, and decryption, $D_K(.)$, with key K. Function F is encrypted using key K: $G = E_K(F)$. The code G is executed based on input X: $U = G(X)$. The unicity distance is given by*

$$UD \approx \frac{H(K)}{R_{func} \log |\mathcal{O}|}, \tag{9.1}$$

where $\mathcal{O}$ represents the set of function symbols and R_{func} the average redundancy in the functions. ◇

Proof.

To be consistent with our previous notation, we let the entropies $H(F^L)$ and $H(G^L)$ represent that of a function F and G of length L, respectively. The joint entropy $H(K, F^L, G^L)$ can be expressed as

$$H(K, F^L, G^L) = H(G^L|K, F^L) + H(K, F^L).$$

Because $H(G^L|K, F^L) = 0$, therefore $H(K, F^L, G^L) = H(K, F^L)$. Key K is chosen at random (e.g. $H(K, F^L) = H(K) + H(F^L)$. The joint entropy can be written as

$$H(K, F^L, G^L) = H(K) + H(F^L). \tag{9.2}$$

Furthermore, the joint entropy can also be expressed as

$$H(K, F^L, G^L) = H(F^L|K, G^L) + H(K, G^L), \tag{9.3}$$

where it holds that $H(F^L|K, G^L) = 0$. Combining equations (9.2) and (9.3) gives

$$H(K, G^L) = H(K) + H(F^L).$$

This can be rewritten as

$$H(K|G^L) = H(K) + H(F^L) - H(G^L). \tag{9.4}$$

The uncertainty of a function depends on the probability distribution of the possible symbols of which a function consists and the amount of redundancy in a function. The redundancy is given by

$$R_{func} = 1 - \frac{H_{func}}{\log |\mathcal{O}|}, \tag{9.5}$$

where H_{func} represents the uncertainty of a symbol in a function and $\mathcal{O}$ is the set of symbols used to construct a function. The average uncertainty of a function of length L, $H(F^L)$, can then be approximated by $L \cdot H_{func}$:

$$H(F^L) \approx L(1 - R_{func}) \log |\mathcal{O}|.$$

Furthermore, it holds that $H(G^L) \leq L \log |\mathcal{O}|$. Substituting these values in equation (9.4) results in

$$H(K|G^L) \geq H(K) - LR_{func} \log |\mathcal{O}|. \tag{9.6}$$

Following Stinson's approach, the average number of spurious keys is determined. Let $\mu(g)$ represent the number of candidate keys when a cipher function is given. The minimum length of the cipher function whose number of spurious keys is zero determines the unicity distance. The average number of spurious keys is given by

$$\bar{s}_L = \sum_{g \in \mathcal{G}(L)} P_G(g)(\mu(g) - 1).$$

Because $\sum_{g \in \mathcal{G}(L)} P_G(g) = 1$, the average number of spurious keys is equal to

$$\bar{s}_L = \sum_{g \in \mathcal{G}(L)} (P_G(g)\mu(g)) - 1. \tag{9.7}$$

The conditional entropy $H(K|G^L)$ can be expressed as

$$\begin{aligned} H(K|G^L) &= \sum_{g \in \mathcal{G}(L)} P_G(g) H(K|g) \\ &\leq \sum_{g \in \mathcal{G}(L)} P_G(g) \log(\mu(g)). \end{aligned}$$

Using Jensen's inequality, this can be rewritten as

$$H(K|G^L) \leq \log \sum_{g \in \mathcal{G}(L)} P_G(g)\mu(g).$$

Combining this inequality and the expressions in (9.6) and (9.7) gives

$$\log(\bar{s}_L + 1) \geq H(K) - LR_{func} \log |\mathcal{O}|. \tag{9.8}$$

The unicity distance is the minimum value of L for which the number of spurious keys is zero. This results in the following expression for the unicity distance:

$$UD \approx \frac{H(K)}{R_{func} \log |\mathcal{O}|}. \tag{9.9}$$

□

When all keys are equiprobable, $H(K)$ can be replaced by $\log |\mathcal{K}|$. Here the encryption of a function can be considered as encryption of data; only the language used is different. The difference in language is expressed in the terms R_{func} and H_{func}. A smaller value of R_{func} will contribute positively to the unicity distance. Less redundancy results in a larger value of the unicity distance, which positively influences the security of a system. This means that more cipher functions (encrypted with identical keys) can be used while privacy is provided.

9.5 Unicity Distance in the Case of Plaintext Attacks

In chapter 7 the unicity distance was derived when plaintext attacks are taken into account. This result will be used in this section to derive the unicity distance for mobile code protection against plaintext attacks. Important is that the pairs (f, g) are generated according to protocol 3 (section 8.3). The length of the functions and corresponding cipher functions are all equal to that of the cipher functions and corresponding plain functions, which the attacker knows and the cipher function of which he does not know the corresponding plain function.

Function F is encrypted using key K, which results in an encrypted function G. When G is executed on a certain input X, it results in parameter U. The unicity distance is then as given in theorem 18.

Theorem 18 *Let $(F, G, K, X, Y, U, E_K(.), D_K(.))$ denote a mobile code system protected by encryption, $E_K(.)$, and decryption, $D_K(.)$, with key K. Function F is encrypted using key K: $G = E_K(F)$. The code G is executed based on input X: $U = G(X)$. The attacker has access to l (f, g) pairs, generated by protocol 3. The unicity distance is given by*

$$UD \approx \frac{H(K|T^l)}{R_{func} \log |\mathcal{O}|}, \tag{9.10}$$

where $\mathcal{O}$ represents the set of function symbols and R_{func} the redundancy in the functions. ◇

Proof. The proof of this theorem is a combination of the proofs of theorem 12 and theorem 17, that provides the unicity distance in case of plaintext attacks and mobile code respectively. Although the proof is a combination of these two theorems, full proof is given here to provide more clarity.

Let $\mu(g, l)$ represent the number of candidate keys when a certain ciphertext g and l (f, g) pairs are given. The number of spurious keys is then for a particular ciphertext equal to $\mu(g, l) - 1$. The average number of spurious keys for a given ciphertext of length L as a function of l is then given by:

$$\bar{s}_L(l) = \sum_{g \in \mathcal{G}(L)} \sum_{t_1 \in \mathcal{T}(L)} \cdots \sum_{t_l \in \mathcal{T}(L)} P_{GT^l}(g, t_1, \ldots, t_l)(\mu(g, l) - 1),$$

where $\mathcal{G}(L)$ and $\mathcal{T}$ represent the sets of cipher functions of length L and of (f, g) pairs of length L, respectively. This can be represented as

$$\bar{s}_L(l) = \sum_{g \in \mathcal{G}(L)} \sum_{t_1 \in \mathcal{T}(L)} \cdots \sum_{t_l \in \mathcal{T}(L)} [P_{GT^l}(g, t_1 \ldots t_l)\mu(l, g)] - 1, \tag{9.11}$$

because $\sum_{g \in \mathcal{G}(L)} \sum_{t_1 \in \mathcal{T}(L)} \cdots \sum_{t_l \in \mathcal{T}(L)} [P_{GT^l}(g, t_1 \ldots t_l)] = 1$. To determine the unicity distance when plain-text-attacks are allowed one should take into account that

the attacker has access to l message-ciphertext pairs. Hence, it is important to determine $H(K|G^L, T^l)$, where G^L denotes function G existing of L symbols. This can be written as:

$$H(K|G^L, T^l) = H(K, G^L, T^l) - H(G^L, T^l). \tag{9.12}$$

The joint entropy $H(K, G^L, T^l)$ is equal to $H(K, F^L, T^l)$ as one key connects a particular message and ciphertext, e.g. a key represents a one-to-one relation for a particular message and ciphertext. Using this and rewriting the joint entropy in the form of conditional entropy, equation (9.12) evolves in

$$H(K|G^L, T^l) = H(F^L|K, T^l) + H(K, T^l) - H(G^L, T^l). \tag{9.13}$$

Because the keys and messages are chosen independently
$H(F^L|K, T^l) = H(F^L|T^l)$. The joint entropies $H(K, T^l)$ and $H(G^L, T^l)$ can be written as conditional entropies. Equation (9.13) then becomes:

$$H(K|G^L, T^l) = H(F^L|T^l) + H(K|T^l) - H(G^L|T^l). \tag{9.14}$$

The maximum value $H(G^L|T^l)$ can take is equal to $l \log |\mathcal{B}|$, where $\mathcal{B}$ is defined as the set of symbols for the ciphertexts. We assume that the number of symbols for the ciphertext is equal to the number of symbols of the plain function, where the set of plaintext symbols is denoted by $\mathcal{A}$, hence $H(G^L|T^l) \leq L \log |\mathcal{A}|$. Then, equation (9.13) results in

$$H(K|G^L, T^l) \geq H(F^L|T^l) + H(K|T^l) - L \log |\mathcal{A}|. \tag{9.15}$$

In case $l \ll |\mathcal{M}(L)|$ it will hold that $H(F^L|T^l) \approx H(F^L) \approx LH_{func}$, where H_{func} is defined as the entropy of the language per symbol ($H_{func} = (1 - R_{func}) \log |\mathcal{A}|$). R_{func} denotes the redundancy of functions. Equation (9.15) results in:

$$H(K|G^L, T^l) \geq H(K|T^l) - LR_{func} \log |\mathcal{A}|. \tag{9.16}$$

As this is an expression in terms of the ciphertext length L, it is desirable to relate the number of spurious keys to $H(K|G^L, T^l)$ because this will relate to the unicity distance. $H(K|G^L, T^l)$ can also be written as:

$$\begin{aligned} H(K|G^L, T^l) &= \sum_{g \in \mathcal{G}(L)} \sum_{t_1 \in \mathcal{T}(L)} \cdots \sum_{t_l \in \mathcal{T}(L)} P_{G|T^l}(G^L, t_1 \ldots t_l) H(K|G^L, t) \\ &\leq \sum_{g \in \mathcal{G}(L)} \sum_{t_1 \in \mathcal{T}(L)} \cdots \sum_{t_l \in \mathcal{T}(L)} P_{GT^l}(G^L, t_1 \ldots t_l) \log(\mu(g, l)). \end{aligned}$$

Using Jensen's inequality, this results in

$$H(K|G^L, T^l) \leq \log \sum_{\mathcal{G}(L)} \sum_{t_1 \in \mathcal{T}(L)} \cdots \sum_{t_l \in \mathcal{T}(L)} P_{GT^l}(G^L, t_1 \ldots t_l) \mu(g, l).$$

Hence, using equation (9.11) it results in

$$H(K|G^L, T^l) \leq \log(\bar{s}_L(l) + 1). \tag{9.17}$$

Combining equations (9.16) and (9.17) results in a relation between the ciphertext length and the average number of spurious keys:

$$\log(\bar{s}_L(l) + 1) \geq H(K|T^l) - lR_{func} \log |\mathcal{A}|.$$

Then the average number of spurious keys becomes:

$$\bar{s}_L(l) \geq \frac{2^{H(K|T^l)}}{|\mathcal{A}|^{LR_{func}}} - 1. \tag{9.18}$$

□

The unicity distance is proportional to the uncertainty on the key given l pairs of (f, g). In general, the uncertainty of the key given l pairs of (f, g) will be lower than that when these pairs are not known (ciphertext-only attack). Therefore, the unicity distance for plaintext attacks will be smaller than for ciphertext-only attacks.

9.6 Conclusions

The unicity distance is an important measure for designing a cryptosystem, as it provides a measure of how many times a key can be used to encrypt functions and still confidentiality can be provided. An increase of the unicity distance means that more functions of a specific length can be encrypted based on the same key.

This chapter gave expressions for the unicity distance in the case of ciphertext-only and plaintext attacks. They differ in the factors $H(K)$ and $H(K|T^l)$. This is logical as in the case of plaintext attacks knowledge of several plain functions and cipher functions reduces the uncertainty on the key, which has influence on the value of the unicity distance (the unicity distance becomes smaller).

The unicity distance for mobile code or classical message encryption are equal as the output values U do not contribute to the unicity distance. Therefore, for the derivation of the unicity distance the output of the cipher function does not influence the unicity distance, it is not relevant whether an algorithm exists to decrypt U.

Chapter 10

Conclusions and Discussion

The objective of this thesis was to provide solutions to privacy protection and theoretical limits to the level of privacy provided for mobile code. This chapter gives the conclusions in the form of a summary, separately for the agent privacy model and the mobile code privacy model. This is followed by a section on the consequences how the first model influences the second model and vice versa. Finally, recommendations are given for future research.

10.1 Summary of the Results

10.1.1 Agent Privacy Model

The first model described was the agent privacy model, which describes a more practical approach to finding solutions for practical applications. An agent environment consists of different players, and each of these players may have a different level of trustworthiness. Software agents run on an agent platform that is operated by a host. As agents are assumed to have the capability of being mobile, it is not always known beforehand to where they will travel. The host that an agent visits may not be trustworthy, and in this thesis, it is assumed that the host is untrustworthy in the sense that it is curious about the actions and content of the agent. The hosts are considered to be trustworthy in the sense that they execute the agent's code correctly and do not alter its code, but by observing the agent's actions, it will try to obtain advantageous information.

Communication.
The first problem was the problem of communication between agents. In an agent environment this is a difficult challenge when one must take into account that even the platform where the code is executed may not be able to successfully eavesdrop on the communication between agents. A double encryption such that a key transformation on the encrypted data can take place in such a way that at no moment in time the data is available in clear text, provides a solution to this problem. It is possible to perform

this operation using the ElGamal encryption scheme; however how the algorithm is used mainly determines the level of security that can be provided during this data exchange.

Execution Privacy.
The second problem addressed was the protection of the agent's tasks. An agent is programmed to perform a certain task on behalf of the user. When this task or certain of its parameters are accessible to the agent platform, this may be to his advantage, especially if he must provide some input to the task. Using the ElGamal encryption scheme, it is possible to encrypt polynomials and perform computations in the encryption domain. The reasoning part is harder, and in this thesis, a simple start has been made by proposing a secure but impractical solution.

Because of the outcome of our research on execution privacy we used information theory to obtain more knowledge on the maximum level of privacy that can be provided; the results of this are described in the next section.

Agent Integrity Mechanisms
If agents are not capable of signing a document without compromising privacy, this seriously limits the success of agent technology. One of the mechanisms used to provide integrity and source authentication is the digital signature. In an agent environment with a curious host, however, using a digital signature poses a particular risk as signing a digital document involves a computation with the private key. To prevent the host from accessing the private key, a new agent digital signature was proposed, using the concept of a blind signature to hide the private key. Using this agent digital signature it is possible to let the agent sign a document on the untrustworthy host, without the host being able to compute the original private key. A drawback of this signature is that an agent platform may repeat the process of signing a document, on a second message of his own choice and the signature will still be valid. A solution is proposed to prevent this, but it is only a threshold for the platform ensuring that it is not to his advantage to perform such double signing.

10.1.2 Conclusions on the Theoretical Approach

The second part of this thesis focussed on deriving theoretical limits of the extent in which mobile code can be protected. The mobile code privacy model was used for this purpose. Instead of the term 'agent', the term 'mobile code' was used to emphasize the generality of the approach used. Here, for mobile code, information theory was used and compared to Shannon's original approach.

Shannon's approach of using information theory actually only covers ciphertext-only attacks. In chapter 7, Shannon's results were extended to plaintext attacks. New definitions of perfect secrecy and maximum secrecy were given for plaintext attacks. Especially in the case where all the ciphertexts are encrypted with the same key, it is concluded that when perfect secrecy is required, the relationship between the message and ciphertext should not be given by one key but multiple keys. The number of keys needs to stand between one message and one ciphertext depends on the number of

ciphertext pairs to which the attacker has access. This lower bound of the number of keys is derived in chapter 7. Because plaintext attacks are considered, it also influences the requirements for providing perfect secrecy.

The main difference between providing confidentiality to mobile code compared to conventional data is that the code is executed in its encrypted form. Especially in mobile code plaintext attacks are relevant, as code may be executed at multiple hosts, these hosts may conspire and combining the separate observations may result in knowing the exact behavior of the code. This characteristic of mobile code protection leads to different theoretical results than those of conventional data protection.

In chapter 8, information theory was used in the mobile code privacy model. In the case of ciphertext-only protection, a new definition of perfect secrecy was given, combined with the condition under which perfect secrecy can be provided. In the case of mobile code protection, the attacker possibly has knowledge of the input and output of the function, which may provide an advantage in obtaining the key. This knowledge is taken into account in the definition of perfect secrecy. From the definition of perfect secrecy it can be concluded that in case of ciphertext-only attacks and when all functions are equiprobable, not only should the key space be at least as large as the function space, but also the output space. Furthermore, a mobile code dilemma was derived consisting of two dilemmas. Increasing the protection of the key when both the function and cipher function are known (e.g. a plaintext attack) will decrease the uncertainty of the function when only the cipher function is known (ciphertext-only attack), given that the uncertainty with respect to the key given cipherfunction remains constant. Similarly, increasing the uncertainty of the key when the function and output value are given, will decrease the uncertainty of the function when only the output value is given, given that all other uncertainties remain constant. Both situations are undesirable, and when designing a protection for mobile code, one must take this knowledge into account as a trade-off must be made in order to provide an adequate level of confidentiality.

In case of protection against plaintext attacks, the situation is a little different. A definition of maximum secrecy was given for mobile code protection. This entails that multiple keys must be responsible for the transformation of one message into a particular ciphertext. The amount of keys that are necessary to achieve this maximum level of secrecy was given. As the amount of keys grows exponentially with respect to the number of possible messages, it is not practical to have a system with maximum secrecy.

Chapter 9 extended the approach from the previous chapter by deriving the unicity distance for mobile code to be protected against both ciphertext-only attacks and plaintext attacks. Although a classical secrecy system (as defined by Shannon) differs from the mobile code privacy model in the fact that an encrypted function can be executed, the unicity distances for these two systems are equal.

Concluding, it can be said that in this theoretical approach, information theory was successfully used to derive theoretical limitations to the protection of mobile code.

However, this did not always result in conditions that can be used in practice.

10.1.3 Theory and Practice Combined

The first part of this thesis discussed practical solutions for mobile code protection. All of these solutions were based on public key cryptography. Because information theory cannot be used to model public key systems, these solutions do not fulfill the requirements for perfect secrecy or any of the other limits derived in the second part of this thesis.

However, these practical solutions do demonstrate that a certain level of protection of mobile code can be achieved. Using public key cryptography in an untrustworthy environment has at least the advantage that encryptions can be done in such an environment. But the level of secrecy provided depends on the computation power of the host. In theory it is possible to provide perfect secrecy for mobile code, but as was shown in the analysis for plaintext attacks, a massive number of keys is required to provide this level of secrecy, rendering mobile code systems that provide this maximum level of secrecy impractical. As is common in cryptography, a trade-off must be made between the level of practicality and security.

10.2 Discussion

Although it was tried in this thesis to present a complete part of research, still many recommendations for future research can be given.

Agent privacy model
The objective of providing confidentiality during data exchange was achieved, but some recommendations can still be given. The solution provided in this thesis only considers the one-way communication of the exchange of confidential data, while one is actually also interested in providing confidentiality during an interactive communication. If an agent receives a message from another agent, it should be able to respond to this message. Within the agent privacy model, this would mean that the response and the reasoning how to respond would have to occur in the encrypted domain. The concept of execution privacy may be useful here, but more research is necessary to obtain knowledge what is the maximum level of confidentiality one can achieve for a two-way communication.

In the agent model, the agent was seen as the extension of the user and it would perform all the tasks as if it were the user. A different approach is to let the user employ multiple agents and by co-operation of these agents they perform one task. A result is that the model changes and therefore also the requirements to guarantee privacy. By distributing several security techniques it may be possible to improve on the level of security that can be achieved. A technique like multi-party computation [102] may provide solutions.

This approach may help to prevent the double signing problem. If two agents instead of one are used to sign a document, it may be possible that the activation of

the signature is now a procedure of combining the results of these two agents, and by dividing the signing process into smaller parts it may result in a higher security level.

Furthermore, the task may be divided over a set of agents, but even with these subtasks the agent must be able to make decisions based on data it receives. Therefore, it is of high interest to perform more research in this area. To be able to make decisions based on encrypted data it requires that some information is present in the encrypted data on which a decision can be made. However, it is against the fundamentals of an encryption algorithm that information about the data should be present in the ciphertext as the objective of an encryption algorithm is to make that link between ciphertext and plaintext invisible. Therefore, it seems unlikely to be able to develop an algorithm that can make decisions on encrypted data, and provide an optimal level of confidentiality at the same time. However, it may be possible to develop an encryption algorithm that can make decisions based on encrypted data at the cost of a lower level of confidentiality. If this is possible, it is important that a measure or quantity is present that can represent this trade-off between security level and autonomy of the agent such that the user can tune the system towards his preference.

Mobile code privacy model

Several recommendations can be given for further research within the mobile code privacy model.

In this thesis a theory was developed that takes plaintext attacks into account, such that it was possible to derive properties like maximum secrecy and the unicity distance. However, not one example could be given of an encryption scheme that has the property of maximum secrecy. Although, it is known beforehand that such a scheme will be impractical due to the large amount of keys, it would be of high interest to know whether such a scheme exists. The main requirement is, that one message must be linked to a particular ciphertext by multiple keys, but a second message should be linked to a second ciphertext by a different set of keys. These two sets may overlap, but may not be equal. It is a question whether a more efficient encryption algorithm can be developed than a look-up-table.

The mobile code privacy model was designed to provide privacy towards the user, when he makes use of mobile code. Based on this the solution was proposed to provide confidentiality to the mobile code to protect privacy. Characteristics and properties were set up with an attacker in mind that has unlimited resources and time. However, the strategy of the attacker was not taken into account. The concept that the attacker operates according to a strategy was first introduced by [10], where error probability is used as a measure to compute the (inverse of) success of the attacker. Because the attacker in the mobile code privacy model has so much power and opportunities to attack the system, it is of interest to research this strategy of the attacker with respect to its probability of success.

Appendix A

Notations

Table A.1: Explanation of the used letters, most of them can be written according to the various notations of table A.2

Letter	Explanation
A	Symbols of message
C	Ciphertext
G	Cipher function
F	Plain function
K	Key
L	Length
M	Message
T	(m, c) or (f, g) pairs
U	Output of encrypted function
X	Input function
Y	Output function

Table A.2: Notations used in chapters 6 - 9

1	General notation for a message (notation for a random variable)	M
2	A message*	m_i or m
3	Set of all messages	$\mathcal{M}$
4	Number of messages in the set $\mathcal{M}$	$\|\mathcal{M}\|$
5	A set of l messages (is a subset of $\mathcal{M}$)	$\mathcal{M}^l$
6	Number of messages in a set of l messages	$\|\mathcal{M}^l\| = l$
7	General notation for l (respectively a number of) messages from $\mathcal{M}$	M^l
8	General notation to denote a i-th message	M_i
9	A message of the subset of l messages*	m^l of m_i^l
10	Notation for l messages from $\mathcal{M}$	$m_1, \ldots, m_l$
11	General notation for a symbol (notation for a random variable)	A
12	A specific message from the set of messages	$\hat{a}$
13	A symbol	a_i or a
14	Set of all symbols	$\mathcal{A}$
15	Number of symbols in the set $\mathcal{A}$	$\|\mathcal{A}\| = \alpha$
16	A subset of l symbols	$\mathcal{A}^l$
17	A number of symbols that form together a message	$m = a_1 a_2 \ldots a_L$
18	General notation for a message existing out of L symbols	M^L
19	A message consisting of L symbols	m^L
	* Where no confusion is possible, the notation without indices will be used.	

Appendix B

Perfect Secrecy

Several expressions are equivalent [36].

Theorem 19 *Let (M, C, K) denote a cryptosystem. The following expressions are equivalent:*

1. *The cryptosystem provides perfect secrecy, according to definition 6.3.*
2. $P_{M|C}(m|c) = P_M(m)$.
3. $H(M|C) = H(M)$.

$\diamond$

Proof. First it is proven that when $P_{M|C}(m|c) = P_M(m)$, it holds that $I(M;C) = 0$. Using the definition of conditional entropy, $H(M|C)$ can be expressed as

$$H(M|C) = -\sum_{m\in\mathcal{M}}\sum_{c\in\mathcal{C}} P_{M|C}(m|c)P_C(c)\log P_{M|C}(m|c).$$

Because $P_{M|C}(m|c) = P_M(m)$, this results in

$$\begin{aligned} H(M|C) &= -\sum_{m\in\mathcal{M}}\sum_{c\in\mathcal{C}} P_M(m)P_C(c)\log P_{M|C}(m|c) \\ &= -\sum_{c\in\mathcal{C}} P_C(c)\sum_{m\in\mathcal{M}} P_M(m)\log P_{M|C}(m|c) \\ &= -\sum_{m\in\mathcal{M}} P_M(m)\log P_M(m) = H(M). \end{aligned}$$

When we look at the definition of mutual information, it is clear that when M and C are independent, $I(M;C) = 0$. To prove the opposite direction it must be shown that when $I(M;C) = 0$, the variables M and C are independent. When $I(M;C) = 0$, it holds that $H(M) = H(M|C)$. The joint entropy can be written as

$$H(M,C) = H(M|C) + H(C).$$

In case of $I(M;C) = 0$, this becomes

$$H(M,C) = H(M) + H(C). \tag{B.1}$$

Therefore, if equation (B.1) can be proven to be true only when M and C are independent, it is also proven that when $I(M;C) = 0$, M and C are independent. According to the definition of entropy, $H(M)$ can be written as

$$H(M) = -\sum_{m\in\mathcal{M}} P_M(m)\log P_M(m),$$

$$H(M) = -\sum_{m\in\mathcal{M}}\sum_{c\in\mathcal{C}} P_{MC}(m,c)\log P_M(m). \tag{B.2}$$

An equivalent relation holds for the entropy of the ciphertext:

$$H(C) = -\sum_{c\in\mathcal{C}}\sum_{m\in\mathcal{M}} P_{MC}(m,c)\log P_C(c). \tag{B.3}$$

Adding equations (B.2) and (B.3) results in

$$H(M) + H(C) = -\sum_{m\in\mathcal{M}}\sum_{c\in\mathcal{C}} P_{MC}(m,c)\log(P_M(m)P_C(c)).$$

The joint entropy is given by

$$H(M,C) = -\sum_{m\in\mathcal{M}}\sum_{c\in\mathcal{C}} P_{MC}(m,c)\log P_{MC}(m,c).$$

And it holds that

$$H(M,C) \leq H(M) + H(C), \tag{B.4}$$

and based on corollary 1, equation (B.4) is an equality if and only if $P_{MC}(m,c) = P_M(m)P_C(c)$. Hence $I(M;C) = 0$ if and only if M and C are independent. Therefore, the two definitions of perfect secrecy are equivalent. □

Theorem 20 *Let (M,C,K) denote a cryptosystem. Protocol 1 is used to generate l pairs of (m,c). The following expressions are equivalent.*

1. *The cryptosystem provides perfect secrecy according to definition 8.*
2. $P_{M|CT^l}(m|c,t_1,\ldots,t_l) = P_M(m)$.
3. $H(M|C,T^l) = H(M)$

Proof. The proof is equivalent to the one of theorem 19, but the extra variable T is incorporated. □

Bibliography

[1] M. Abadi and J. Feigenbaum. Secure circuit evaluation. *Journal of Cryptology*, 2:pages 5–21, 1990.

[2] M. Abe and E. Fujisaki. How to date blind signatures. *Advances in Cryptology - Asiacrypt'96*, pages 244–51, 1996.

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

[4] D. Banisar. Privacy and human rights. *Electronic Privacy Information Center*, Washington/London, 2000.

[5] B. Barak and O. Goldreich. On the (im)possibility of obfuscating programs. *Crypto 2001, Lecture Notes in Computer Science*, pages 1–18, 2001.

[6] P. Beauchemin and G. Brassard. A generalization of hellman's extension to shannon's approach to cryptography. *Journal of Cryptology*, 1(2):129–1132, 1988.

[7] M. Bellare and S.K. Miner. A forward-secure digital signature scheme. In *Advances in Cryptology - CRYPTO'99*, volume Lecture Notes in Computer Science, 1666. Springer-Verlag, New York, 1998.

[8] C.H. Bennett, F. Bessette, and G. Brassard. Experimental quantum cryptography. *Journal of Cryptology*, (1):pages 3–28, 1992.

[9] I. Biehl, B. Meyer, and S. Wetzel. Ensuring the integrity of agent-based computations by short proofs. *Mobile agents, Lecture Notes in Computer Science 1477*, pages 183–94, 1998.

[10] D.E. Boekee and J.C.A. van der Lubbe. Error probabilities and transposition ciphers. In *Ninth Symposium on Information Theory in the Benelux*, pages 155–162, 1988.

[11] J.J. Borking and E.P. Siepel. Intelligent agents and privacy. Technical report, Achtergrondstudies en verkenningen, nr 13, Registratiekamer, 1999.

[12] N. Borselius, C.J. Mitchell, and A. Wilson. On mobile agent based transactions in moderately hostile environments. In *Advances in Network and Distributed Systems Security - Proceedings of IFIP I-NetSec'01*, pages 173–186. Kluwer Academic, 2001.

[13] Stefan Brands. Untraceable off-line cash in wallet with observers (extended abstract). In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages pages 302–318. Springer-Verlag, 22–26 August 1993.

[14] G. Brassard and C. Crépeau. 25 years of quantum cryptography. *SIGACT news*, 27(3):pages 13–24, 1996.

[15] C. Cachin, J. Camenisch, J. Kilian, and J. Müller. One-round secure computation and secure autonomous mobile agents. *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP), Lecture notes in Computer Science*, 1853:512–23, 2000.

[16] Christian Cachin and Ueli M. Maurer. Unconditional security against memory-bounded adversaries. *Lecture Notes in Computer Science*, 1294:pages 292–, 1997.

[17] J.L. Camenisch, J-M. Piveteau, and M.A. Stadler. Blind signatures based on the discrete logarithm problem. *Advances in Cryptology - Eurocrypt'94*, pages 428–32, 1994.

[18] J.L. Camenisch, J-M. Piveteau, and M.A. Stadler. Blind signatures based on the discrete logarithm problem. In *Advances in Cryptology - Eurocrypt'94*, pages 428–32. Springer-Verlag, 1994.

[19] R.M. Capocelli, A. De Santis, L. Gargano, and U. Vaccaro. On the size of shares for secret sharing schemes. In *Advances in Cryptology - Crypto 91*, volume LNCS 196, pages 101–113, 1992.

[20] K. Cartrysse and J.C.A. van der Lubbe. An agent digital signature in an untrusted environment. *Proceedings of the 2nd International Workshop on Security in Mobile Multiagent Systems*, pages 12–17, 2002.

[21] K. Cartrysse and J.C.A. van der Lubbe. Providing privacy to agents in an untrustworthy environment. *Handbook of Privacy and Privacy-Enhancing Technologies*, pages 79–96, 2003.

[22] K. Cartrysse and J.C.A. van der Lubbe. Secrecy in mobile code. *25th Symposium on Information Theory in the Benelux*, pages 161–168, 2004.

[23] K. Cartrysse and J.C.A. van der Lubbe. Information theoretical approach to mobile code. In *International Symposium on Informaton Theory ISIT2005*, 2005.

[24] K. Cartrysse and J.C.A. van der Lubbe. Privacy in mobile agents. *First IEEE Symposium on Multi-Agent Security and Survivability*, August 2004.

[25] K. Cartrysse and J.C.A. van der Lubbe. Privacy in agents: theory, deliverable 13b. Technical report, PISA-project, Delft University of Technology, February 2003.

[26] K. Cartrysse, J.C.A. van der Lubbe, and A. Youssouf. Privacy protection mechanisms, deliverable 11. Technical report, PISA-project, May 2002.

[27] D. Chaum. Blind signatures for untraceable payments. *Advances in Cryptology - Crypto'82, Springer-Verlag*, pages 199–203, 1983.

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

[29] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Advances in Cryptology - Crypto'88*, pages 319–327, 1990.

[30] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *CACM*, 24(2):pages 84–88, February 1981.

[31] J. Claessens, B. Preneel, and J. Vandewalle. Secure communication for secure agent-based electronic commerce. *E-commerce Agents: Marketplace Solutions, Security Issues, and Supply and Demand*, Lecture Notes in Computer Science, vol. 2033, 2001.

[32] C. Collberg and C. Thomborson. On the limits of software watermarking. *http://www.cs.arizona.edu/c̃ollberg*.

[33] C. Collberg and C. Thomborson. Software watermarking: models and dynamic embeddings.

[34] European Commission. *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*.

[35] T.M. Cover and J.A. Thomas. *Elements of information theory*. John Wiley & Sons, 1991.

[36] H. Delfs and H. Knebl. *Introduction to cryptography*. Springer, 2002.

[37] C. Diaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In Springer-Verlag, editor, *Privacy Enhancing Technologies*, 2002.

[38] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Infromation Theory*, 22(6):pages 644–654, 1976.

[39] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):pages 469–72, 1985.

[40] W. Farmer, J. Guutman, and V. Swarup. "Security for mobile agents: Authentication and State Appraisal". In *"In proceedings of the 4th European Symposium on Research in Computer Science (ESORICS'96)"*, pages 118–130, September 1996.

[41] L.C. Ferreira and R. Dahab. Blinded-key signatures: securing private keys embedded in mobile agents. *Proceedings of the 2002 ACM symposium on Applied computing*, pages 82–86, 2002.

[42] P.A. Fouque, J. Stern, and G.J. Wackers. Cryptocomputing with rationals. *Financial-Cryptography.-6th-International-Conference*, Lecture-Notes-in-Computer-Science-Vol.2357. 2003: pages 136-46, 2002.

[43] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*. Springer-Verlag, 1996.

[44] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding Routing Information. In Ross Anderson, editor, *Information hiding: first international workshop, Cambridge, U.K., May 30–June 1, 1996: proceedings*, volume 1174 of *Lecture Notes in Computer Science*, pages 137–150. Springer-Verlag, 1996.

[45] Object Management Group; Agent Platform Special Interest Group. Omg document agent/00-09-01. Technical report, September 2000.

[46] M.E. Hellman. An extension of the shannon theory approach to cryptography. *IEEE transactions on information theory*, 23(3):289–294, May 1977.

[47] R. Hes and J. Borking. Privacy-enhancing technologies: The path to anonymity. *Achtergrondstudies en Verkenningen 11*, 2000.

[48] F. Hohl. Time limited blackbox security: Protecting mobile agents from malicious hosts. *Mobile agents and security, Lecture notes in computer science*, pages 92–113, 1998.

[49] F. Hohl. A framework to protect mobile agents by using reference states. *Proceedings ofthe 20th International Conference on Distributed Computing Systems CICDS2000*, 2000.

[50] JADE. Java agend development framework. *http://jade.tilab.com*.

[51] P. Janca. Pragmatic application of information agents. *BIS Strategic Decisions, Norwell, United States*, 1995.

[52] D.B. Johnson and A.J. Menezes. Elliptic curve dsa (ecdsa): an enhanced dsa. Technical report, Certicom.

[53] G. Karjoth, N. Asokan, and G. Glc. Protecting the computation results of free-roaming agents. *Mobile agents '98, Lecture Notes in Computer Science*, pages 195–207, 1998.

[54] P. Kotzanikolaou, M. Burmester, and V. Chrissikopoulos. Secure transactions with mobile agents in hostile envirionments. *Information Security and privacy, Proceedings of the 5th Australasion Conference, ACISP2000, Lecture Notes in Computer Science*, pages 289–297, 2000.

[55] H. Krawczyk. Simple forward-secure signatures from any signature schme. In *Proceedings of the seventh ACM conference on computer and communications security*, pages 108–115, 2000.

[56] A. Leon-Garcia. *Probability and random processes for electrical engineering*, volume second edition. Addison-Wesley, 1994.

[57] S. Loureiro and R. Molva. Function hiding based on error correcting codes. In *Proceedings of the CryptTEC'99 International Workshop on Cryptographic Techniques and Electronic Commerce*, pages 92–98, 1999.

[58] S. Loureiro and R. Molva. Privacy for mobile code. *Proceedings of distributed object security workshop, OOPSLA'99, Lecture Notes in Computer Science*, pages 184–99, November 1999.

[59] S. Loureiro, R. Molva, and A. Pannetrat. Secure data collection with updates. *Proceedings of the workshop on agents in electronic commerce*, pages 121–130, 1999.

[60] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems.

[61] J. L. Massey. An introduction to contemporary cryptology. *IEEE proceedings*, 76(5):pages 533–549, May 1988.

[62] U. M. Maurer. The role of information theory in cryptography. In *Fourth IMA Conference on Cryptography and Coding*, pages 49–71. IMA, 13–15 1993.

[63] U. M. Maurer. Information-theoretic cryptography (extended abstract). *Lecture Notes in Computer Science*, 1666:pages 47–64, 1999.

[64] U.M. Maurer. Secret key agreement by public discussion from common information. *IEEE Transactions on Information Theory*, 39:pages 733–742, 1993.

[65] U.M. Maurer. Information-theoretic cryptography. In *Advances in cryptology - crypto' 99*, pages 47–64. Springer-Verlag, 1999.

[66] A.J. Menezes. *Elliptic curve public key cryptosystems*. Kluwer Academic Publishers, 1993.

[67] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC press, 1997.

[68] S.-K. Ng. *Protecting mobile agents against malicious hosts*. PhD thesis, "Chinese University of Hongkong", June 2000.

[69] M.G. Reed, P.F. Syverson, and D.M. Goldschlag. Anonymous Connections and Onion Routing. *IEEE Journal on Special Areas in Communications*, 16(4):pages 482–494, May 1998.

[70] M. Reiter and A. Rubin. Crowds: anonymity for web transactions. *ACM transactions on information and system security*, 1:pages 66–92, 1998.

[71] Michael K. Reiter and Aviel D. Rubin. Anonymous Web transactions with crowds. *Communications of the ACM*, 42(2):pages 32–48, February 1999.

[72] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21:pages 120–126, 1978.

[73] A. Romao and M.M. Da Silva. Proxy certificates: a mechanism for delegating digital signatue power to mobile agents. In *Proceedings of the workshop on Agents in Electronic Commerce*, pages 131–140, 1999.

[74] V. Roth. Secure recording of itineraries through cooperating agents. In *"Proceedings of the ECOOP workshop on distributed object security and 4th workshop on mobile object systems: Secure Internet Mobile Computations"*, pages 147–154, France, 1998. INRIA.

[75] V. Roth. On the robustness of some cryptographic protocols for mobile agent protection. *Mobile Agents 2001, Lecture Notes in Computer Science*, pages 1–14, 2001.

[76] T. Sander and C. F. Tschudin. Protecting mobile agents against malicious hosts. *Mobile agents and security, Lecture Notes in Computer Science*, pages 44–60, 1998.

[77] T. Sander and C.F. Tschudin. On software protection via function hiding. *Information hiding, Lecture Notes in Computer Science 1525*, pages 111–23, 1998.

[78] T. Sander and C.F. Tschudin. Towards mobile cryptography. *Proceedings 1998 IEEE symposium on security and privacy*, pages 215–224, 1998.

[79] T. Sander, A. Young, and Y. Moti. Non-interactive cryptocomputing for nc_1. *40th Annual Symposium on Foundations of Computer Science, IEEE*, pages 554–66, 1999.

[80] Fred B. Schneider. Towards Fault-tolerant and Secure Agentry (Invited paper). In *Proceedings of the 11th International Workshop on Distributed Algorithms*, Saarbucken, Germany, September 1997. Also available as TR94-1568, Computer Science Department, Cornell University, Ithaca, New York.

[81] B. Schneier. *Applied cryptography, second edition*. John Wiley & Sons Inc., 1996.

[82] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Privacy Enhancing Technologies*, 2002.

[83] C. E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28:pages 656–715, 1949.

[84] C.E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27(3):pages 379–423 and 623–656, 1948.

[85] G.J. Simmons. A survey of information authentication. In *Proceedings of the IEEE*, volume 76, pages 603–620, 1988.

[86] N. Smart. *Cryptography: an introduction*. McGraw-Hill, 2003.

[87] Markus A. Stadler, Jean-Marc Piveteau, and Jan L. Camenisch. Fair blind signatures. *Lecture Notes in Computer Science*, 921:209+, 1995.

[88] William Stallings. *Cryptography & Network Security: Principles & Practice*. Prentice Hall International, 3rd edition edition, 2003.

[89] D.R. Stinson. *Cryptography, theory and practice*. CRC press, 1995.

[90] W. Trappe and L.C. Washington. *Introduction to Cryptography with Coding Theory*. Prentice-Hall, Inc., 2002.

[91] J. C. A. van der Lubbe. *Basic methods of cryptography*. Cambridge University Press, 1994.

[92] J.C.A. van der Lubbe. *Information Theory*. Cambridge University Press, 1997.

[93] H.C.A. van Tilborg. *An introduction to cryptology*. Kluwer acdemic publishers, 1988.

[94] G.S. Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *AIEE*, 45:pages 109–115, 1926.

[95] G. Vigna. Cryptographic traces for mobile agents. *Mobile agents and Security, Lecture Notes in Computer Science*, pages 137–153, 1998.

[96] D. Welsh. *Codes and Cryptography*. Oxford science publications, 1988.

[97] C.U.D. Westhoff, M. Schneider, and F.Kenderali. Methods for protecting a mobile agents route. In *Proceedings of the Second International Information Security Workshop*, pages 57–71, 1999.

[98] A. Westin. Privacy and freedom. *Atheneum*, New York, 1967.

[99] U.G. Wilhelm, S. Staamann, and L. Buttyán. Introducing trusted third parties to the mobile agent paradigm. *Secure Internet Programming, Lecture notes in computer science*, 1603:pages 469–89, 1999.

[100] S. Wolf. Unconditional security in cryptography. *Lectures on Data Security*, LNCS 1561, Springer-Verlag:pages 217–250, 1999.

[101] A.D. Wyner. The wire-tap channel. *Bell System Technical Journal*, 54(8):pages 1355–1387, 1975.

[102] A.C. Yao. Protocols for secure computations. *FOCS*, 1982.

[103] B. Yee. A sanctuary for mobile agents. *Secure Internet Programming, Lecture notes in computer science*, 1603:pages 261–73, 1999.

[104] X. Yi, C.K. Siew, and M.R. Syed. Digital signature with one-time paiir of keys. *Electron. lett.*, pages 130–131, 2000.

[105] A. Young and M. Yung. Sliding encryption: a cryptographic tool for mobile agents. *Proceedings of Fast Software Encryption Workshop 1997, Springer-Verlag, Lecture Notes in Computer Science*, pages 230–241, 1997.

Samenvatting

Dit proefschrift heeft als doelstelling het geven van oplossingen om de privacy te waarborgen in het geval dat mobiele code wordt gebruikt. Mobiele code is software code dat zich kan verplaatsen over het netwerk en uitgevoerd kan worden op platformen op andere locaties. Een voorbeeld van mobiele code is een ”mobile software agent”. Deze agent is in staat om taken uit naam van zijn gebruiker uit te voeren. De agent reist over het netwerk en wordt op verschillende plaatsen uitgevoerd, maar het is niet altijd vooraf bekend of deze locaties vertrouwd kunnen worden. Een software agent voert taken uit, uit naam van zijn gebruiker en daarom is het noodzakelijk dat deze agent zich bewust is van de privacy risico’s voor de gebruiker. Een voorbeeld is dat een agent de taak heeft om een vliegticket voor zijn gebruiker te kopen. Als de platformen, waar de agent wordt uitgevoerd bijvoorbeeld een vliegtuigmaatschappij is, is het niet de bedoeling dat dit platform ziet wat de strategie van de agent is om een vliegticket te kopen.

In dit proefschrift worden een aantal oplossingen voor dit probleem gepresenteerd. Daarnaast wordt ook in een analyse beschreven welk niveau van privacy theoretisch maximaal haalbaar is. Hetgene waar privacy voor mobiele code zich onderscheidt van privacy voor klassieke softwaresystemen, is dat het onderliggende platform waar de mobiele code wordt uitgevoerd niet vertrouwd kan worden. Dit onderliggend platform wordt gezien als nieuwsgierig. Het is nieuwsgierig naar de inhoud van de code, maar brengt geen wijzigingen aan in deze code.

Dit proefschrift beschrijft twee benaderingen. De eerste is een praktische aanpak, waarbij enkele oplossingen worden gegeven voor verschillende privacy problemen. Deze oplossingen geven een vorm van privacy, maar deze geven geen informatie over het niveau van privacy dat hiermee gegarandeerd kan worden. Om hier meer duidelijkheid over te krijgen is een tweede benadering gebruikt om te analyseren welk niveau van privacy theoretisch gewaarborgd kan worden.

Bij de praktische aanpak zijn oplossingen voor drie problemen gegeven. Het eerste probleem is het probleem van agent communicatie. Als een agent door zijn gebruiker geïnitieerd wordt, is het vaak niet bekend met welke agenten hij zal communiceren. Als agenten met elkaar wensen vertrouwelijk te communiceren vormt dit een probleem. Als de te communiceren data in vercijferde vorm is opgeslagen in de agent, dient deze eerst ontcijferd te worden en vervolgens weer vercijferd met de sleutel van de communicatiepartner. Tijdens dit proces is er een moment waarbij de vertrouweli-

jke data in klare tekst toegankelijk is voor het platform. Dit kan een privacy risico zijn. In dit proefschrift wordt een oplossing voorgesteld waarbij gebruik gemaakt wordt van dubbele encryptie. Tijdens de communicatie wordt de vercijferde data in de agent nogmaals vercijferd (nu met de sleutel van de communicatiepartner). In de voorgestelde oplossing is het nu zo dat het mogelijk is om een decryptie uit te voeren zodanig dat het resultaat data is die alleen nog vercijferd is met de sleutel van de communicatiepartner. Op deze manier is de vertrouwelijke data op geen enkel moment beschikbaar in klare tekst.

Het tweede probleem waar een oplossing voor gegeven wordt is het probleem dat agenten in staat moeten zijn om software programma's uit te voeren op onbeveiligde platformen zonder dat deze platformen de inhoud van deze programma's mogen weten. Een oplossing wordt gegeven in de vorm van het vercijferen van programma's. Het encryptie-algoritme ElGamal wordt gebruikt om polynomen te vercijferen. Vervolgens wordt een oplossing gegeven hoe beslissingen gemaakt kunnen worden gebaseerd op vercijferde data. Deze oplossing is door zijn hoeveelheid aan berekeningen niet praktisch.

Bij het derde probleem wordt gekeken naar hoe agenten digitale documenten kunnen ondertekenen zonder dat de privacy van zijn gebruiker in gevaar komt. Het zetten van een digitale handtekening vraagt om het gebruik van een geheime sleutel, maar deze data is erg privacy-gevoelig. Een digitale handtekening voor een software agent wordt voorgesteld, waarbij er gebruik gemaakt wordt van het idee achter "blind signatures". De geheime sleutel wordt verborgen op een zodanige manier dat het platform niet in staat is om de geheime sleutel te achterhalen. Een nadeel van deze digitale handtekening is echter dat het platform het proces van het plaatsen van een handtekening kan herhalen en daarmee een tweede document uit naam van de agent kan ondertekenen zonder zijn toestemming. Enkele oplossingen worden aangedragen om deze situatie te voorkomen, waaronder het toevoegen van de identiteit van het platform tijdens het ondertekenen.

De hiervoor genoemde oplossing om polynomen te vercijferen zorgt voor confidentialiteit, maar het geeft geen maat over het maximale niveau van privacy dat behaald kan worden. Om deze maat te kunnen geven, is gebruik gemaakt van de informatietheorie binnen een mobiele code privacy model. Hierbij wordt aangenomen dat een aanvaller toegang heeft tot onbeperkte rekenkracht en tijd.

Een eerste observatie binnen dit mobiele code privacy model is dat klare-tekst-aanvallen niet voorkomen kunnen worden. Echter Shannon's model houdt hier geen rekening mee en in dit proefschrift is dit model aangepast zodanig dat klare-tekst-aanvallen wel hierin passen. Dit leidt tot een nieuwe definitie van absolute veiligheid en de daarbij behorende karakteristieken. Ook de uniciteitsafstand is gegeven voor het geval van klare-tekst-aanvallen.

De theorie met betrekking tot klare-tekst-aanvallen is toegepast op de situatie van privacy voor mobiele code. Privacy wordt bereikt door middel van vercijfering van de functies in de mobiele code. Grenzen worden bepaald voor het maximale niveau van privacy dat behaald kan worden. Ook geeft het informatie over het minimum aantal sleutels dat noodzakelijk is om dit niveau van privacy te behalen. Ook de uniciteitsaf-

stand is gedefinieerd en afgeleid voor het mobiele code privacy model.

Door gebruik te maken van informatietheorie was het mogelijk om theoretische grenzen te geven voor het mobiele code privacy model. De belangrijkste aanbeveling van dit proefschrift is om onderzoek te verrichten wat deze resultaten betekenen in de praktijk, of het mogelijk is om een algoritme te ontwikkelen waarbij het maximale niveau van privacy bereikt kan worden.

Summary

This thesis' objective is to provide privacy to mobile code. A practical example of mobile code is a mobile software agent that performs a task on behalf of its user. The agent travels over the network and is executed at different locations of which beforehand it is not known whether or not these can be trusted. Because a mobile software agent performs actions on behalf of its user, agent must be protected in order to provide privacy to the user. For example, a software agent must purchase a flight ticket. Other parties like airlines are then not entitled to know the agent's strategy when the agent will purchase the ticket.

In this thesis, several solutions are provided to this problem, and to obtain more in depth knowledge on protecting privacy in mobile code a theoretical analysis was performed. Essential in this problem is that the execution environment cannot be trusted. It is curious in the content of the mobile code, although it does not change the content.

Two approaches are covered in this thesis. First, a practical approach is given that presents several solutions to various aspects of the mobile code privacy problem. These solutions provide a level of privacy protection, however, it does not provide knowledge on what level of privacy can be provided. To answer this question a theoretical analysis is made to derive limits of the maximum privacy level that can be achieved.

For the practical approach three problems are addressed. First, the problem of agent communication is investigated. If an agent needs to communicate to other agents, but beforehand it is not always known with whom it will communicate, a problem rises when the data to be transmitted is confidential. If the confidential data is stored encrypted in the agent, it normally requires first a decryption and then an encryption with a session key before it can be transmitted. However, this means that at some point in time the host will be able to see the data in clear text. A solution is proposed to this problem that makes use of double encryption. Based on the encrypted data a transformation is performed such that the result is again encrypted data but the key is the correct key such that the other party can decrypt the message. This transformation is done such that at no point in time the clear text is available, and therefore the host cannot access the data anymore.

The second problem is that agents should be capable of executing programs, but the host, where the program is executed, should not be able to know the content.

A solution is to encrypt the program. By using ElGamal's encryption scheme it is achieved that polynomials can be encrypted. Furthermore, a solution is given how decisions can be made on encrypted data, although this solution is impractical.

Software agents should be able to sign documents when they are located at a remote host. However, a signature requires usage of the private key. It means that the host can have access to the agent's private key, which is the agent's most private data. Therefore, an agent digital signature is proposed that is based on the concept of blind signatures. The private key is blinded such that the host cannot have access to this private key. Based on this agent digital signature it is for the host possible to sign a second document without the agent's consent. Several solutions are proposed to prevent this situation.

The solution to encrypt polynomials does provide confidentiality, however, it provides little insight on the maximum level of privacy that can be achieved. Therefore, a mobile code privacy model was designed, such that by using information theory theoretically boundaries can be derived. An attacker is considered with unlimited computation power and memory resources.

A first result is that for the mobile code privacy model plaintext attacks cannot be prevented. Shannon's secrecy model does not take these into account. Therefore, Shannon's theory was extended to plaintext attacks. New definitions of perfect secrecy were given as well as characteristics when systems provide these levels of secrecy. Furthermore, the unicity distance is derived in this new situation.

The theory of incorporating plaintexts is then applied to the mobile code privacy model, as well as the theory for ciphertext-only attacks. Privacy is provided by encryption of the functions in the mobile code. It provides boundaries on the level of privacy that can be achieved. Moreover, it also provides information on the minimum amount of keys necessary to obtain a maximum level of secrecy. Furthermore, the unicity distance is derived for the possible types of protection.

By using an information theoretic approach theoretical bounds were given in case of mobile code privacy, taking into account the attacker with unlimited computation power. The most important recommendation for future research is the question what these results means in practice; whether or not an algorithm exists that fulfills this maximum level of secrecy.

Acknowledgments

Many people contributed in one way or the other to this thesis.

First, I would like to thank my supervisors Jan van der Lubbe and Inald Lagendijk. They have been of great help throughout the years by supporting my ideas and asking critical questions.

I would also like to thank all the participants of the PISA project. The people in the project helped me to stay focussed and I learned a lot from this multidisciplinary project. Here, I would like to thank Martijn van Breukelen and Paul Verhaar in particular.

I always enjoyed working at the Information and Communication Theory Group, because of the great atmosphere. I would like to thank everyone who has been part of that group over the last five years. A few people I would like to thank in particular. Annett and Anja were of great help for all administrative tasks and the many nice conversations we had. Ben, Hans, Robbert, Erik and Janroel have always been of great help for the many computer problems I had over the years. I always enjoyed the conversations on cryptography and many cultural aspects with Bartek and Kosmas. Furthermore, I would like to thank Jesper, Mark and Peter-Jan for the numerous breaks I had with them. Thanks to Geerd for all the discussions and fun we had throughout our studies and work at the ICT group.

A special thanks goes to Annelies as she designed the cover of this thesis. Furthermore, we have been such good friends for so many years, that I thank her for much more than just the cover. I would like to thank all my other friends for their support.

The family Blomjous I would like to thank for their sincere interest in my PhD. and everything that comes with that.

To my sister, Anouk, thanks for all the support of course, but also for the many times she listened to my mathematical and technical problems. Although she often did not have to give the answers, just the listening provided me the solutions.

The cliché that there are not enough words to say thanks, definitely holds for my parents. Thanks for all the opportunities you gave me and the support. The fact that they always believed in me, made me come this far.

Finally, I would like to thank Willem-Jan for his love and patience. He is the one that had to deal with all my mood swings and the times I thought I would never finish this thesis. Thanks and I hope we will have many more great years together.

Curriculum Vitae

Kathy Cartrysse was born in Knokke-Heist (Belgium) on April 17, 1977. She obtained her VWO-diploma at the Aloysiuscollege in Den Haag in 1995.

This was followed by a study of electrical engineering at Delft University of Technology. During these studies she was an exchange student at Worcester Polytechnic Institute, Massachusetts, USA, in 1998 - 1999, where she received a master of science degree in 2000. For Delft University of Technology she graduated in 2000 with a master thesis entitled "Payment scheme providing privacy in a mobile environment". The research for this thesis was conducted at ING Group, Amsterdam.

In January 2001 she started as a PhD.-student at the Information and Communication Theory Group at Delft University of Technology in the research area cryptography. Her work for this thesis was conducted within the PISA-project funded by the European Union. Since April 2005, she holds a postdoc position in the same group.