Securing Mobile Ubiquitous Services using Trusted Computing

Adrian Ho Yin Leung

Technical Report
RHUL–MA–2009–17
10 July 2009

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England

Securing Mobile Ubiquitous Services using Trusted Computing

by

Adrian Ho Yin Leung

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
Department of Mathematics
Royal Holloway, University of London

2009

To my daughters,
Louisa and Lavigne

Declaration

These doctoral studies were conducted under the supervision of Professor Chris J. Mitchell.

The work presented in this thesis is the result of original research carried out by myself, in collaboration with others, whilst enrolled in the Information Security Group, Department of Mathematics, Royal Holloway, University of London as a candidate for the degree of Doctor of Philosophy. The research presented in Chapter 5 is my own work, although I received many constructive comments from Liqun Chen and Chris J. Mitchell. The results presented in Chapter 6 are joint work with Raphael Phan. The weaknesses described in Section 6.4 of the SSB Scheme were jointly identified by both of us, while the suggested solutions in Section 6.5 are my work. The results in Chapter 7 and 8 were research carried out by myself with constructive comments from Chris J Mitchell. The protocol described in Chapter 9 was jointly developed with Geong Sen Poh. However, the idea to use trusted computing to solve the problem was mine.

This work has not been submitted for any other degree or award in any other university or educational establishment.

Adrian Ho Yin Leung
June, 2009

Acknowledgements

My heartfelt thanks to my supervisor Professor Chris J. Mitchell for his guidance, encouragement, motivation, advice, patience and dedication. Chris you are an excellent teacher and mentor; I have learnt a great deal from you in the last few years.

I would like to thank my examiners, Professor Bruce Christianson and Professor Peter Wild for taking the time to examine my thesis and for their constructive comments.

I would like to thank the British Foreign and Commonwealth Office (FCO) and Royal Holloway for awarding the British Chevening/Royal Holloway Scholarship to me. This research would not have been possible without this very generous financial support. Many thanks also to the EU-funded ECRYPT network of excellence for providing conference and travel support. I am also grateful to the Virtual Centre of Excellence in Mobile and Personal communications (Mobile VCE) for inviting me to participate in this exciting research programme on ubiquitous services.

I would like to thank my co-authors Roderick Bloem, Liqun Chen, Haitham Cruickshank, Karin Greimel, Raphael Phan, Geong-Sen Poh, Yingli Sheng, Ronald Toegl and Po-Wah Yau. I am particularly grateful to Liqun Chen for patiently answering my numerous queries on DAA. It has been a pleasure working with and learning from all of you.

And not forgetting the ISG for (i) the very supportive and friendly research environment, (ii) the very approachable academic/research staff (in particular John Austen, Shane Balfe, Jason Crampton, Hilary Ganley, Kenny Paterson, Fred Piper, Hoon Wei Lim, Keith Martin, Scarlet Schwiderski-Grosche, Allan Tomlinson, Stephen Wolthusen, and Po-Wah Yau) with whom I have had the honour of interacting, (iii) the helpful and ever smiling support staff, in particular Pauline Stoner, Jenny Lee, Lisa Nixon, and Jon Hart, and (iv) my fellow PhD students, especially my office mates from McCrea 255.

My deepest gratitude goes to my parents for all their years of unwavering support and love. Thanks also to my wife for taking this journey with me, and my dear daughters Louisa and Lavigne for keeping me motivated, and for making everything worthwhile.

Abstract

This thesis examines how trusted computing technology can be used to enhance the security of ubiquitous services in mobile environments.

It is envisaged that, in a mobile ubiquitous environment, users (through one of their mobile devices and using a range of network access technologies) will be able to seamlessly discover, select, and access a rich offering of services and content from a range of service providers. To realise this vision, it is important that security and privacy issues are addressed from the outset.

Initially we introduce the model of mobile ubiquitous computing that underlies the discussions in the remainder of the thesis. We then identify the security requirements for ubiquitous service provision arising in the context of this model.

In Part II of the thesis we examine the technology of trusted computing. We consider the effectiveness of a recently proposed attack on one of the trusted computing primitives, namely the Direct Anonymous Attestation protocol, and also examine ways in which the attack can be prevented. We further cryptanalyse a trusted computing based protocol designed to secure the storage and distribution of secrets.

In the final part of the thesis, we propose three novel schemes for mobile services security, all using trusted computing as the primary building block. Firstly, we describe a Secure and Private Service Discovery Protocol in which, during the service discovery process, the trustworthiness of a user platform is anonymously authenticated to a service provider, whilst a service provider is simultaneously authenticated to the user. The novel scheme possesses the following desirable properties: user anonymity, service information confidentiality, unlinkability, and rogue blacklisting.

We next present a Device Management Framework for Secure Service Delivery. Apart from providing secure service interactions between the service provider and user devices, the framework is designed to reduce the complexity of device security management tasks for users. The framework also protects the interests of service providers by preventing unauthorised credential sharing amongst user devices. One other novel feature of the framework is that compromised devices are self-revoking, hence removing the need for a cumbersome revocation infrastructure.

Finally, we construct a Privacy-Preserving Content Watermarking Scheme. Our scheme minimises the reliance on a TTP for privacy protection, as the buyer can

generate verifiable pseudonyms on its own. As a result, we are able to reduce communication overheads, and hence improve the overall efficiency compared to existing schemes. In addition, the content provider is able to obtain assurance that a buyer-generated watermark is well-formed. The scheme also provides the following security features: framing resistance, user anonymity, content information confidentiality, unlinkability (even against the TTP), and transaction linkability.

Contents

List of Figures

List of Tables

Abbreviations

AIK:	Attestation Identity Key
API:	Application Program Interface
BBB:	BIOS Boot Block
CA:	Certification Authority
CDP:	Content Distribution Protection
CRL:	Certificate Revocation List
CRTM:	Core Root of Trust for Measurement
DAA:	Direct Anonymous Attestation
DH:	Diffie-Hellman
DME:	Device Management Entity
DoS:	Denial of Service
DRM:	Digital Rights Management
EK:	Endorsement Key
GSM:	Global System for Mobile Communications
IOI:	Items of Interest
ISO:	International Organization for Standardization
MAC:	Message Authentication Code
OAEP:	Optimal Asymmetric Encryption Padding
OCSP:	Online Certificate Status Protocol
P3P:	Platform for Privacy Preferences
PCR:	Platform Configuration Register
PDA:	Personal Digital Assistant
PDE:	Personal Distributed Environment
PII:	Personally Identifiable Information
PKI:	Public Key Infrastructure
RFID:	Radio Frequency Identification
RSA:	Rivest-Shamir-Adleman
RTM:	Root of Trust for Measurement
RTR:	Root of Trust for Reporting
RTS:	Root of Trust for Storage
SHA:	Secure Hash Algorithm
SML:	Stored Measurement Log
SRK:	Storage Root Key
TC:	Trusted Computing
TCG:	Trusted Computing Group
TPM:	Trusted Platform Module
TTP:	Trusted Third Party
VM:	Virtual Machine
VMM:	Virtual Machine Monitor
W3C:	World Wide Web Consortium

Notation

$\|\|$	The concatenation operator
(A_{pk}, A_{sk})	A public/private key pair of a principal A
(K_{pk}, K_{sk})	A public/private key pair for an asymmetric cryptographic algorithm
$A \rightarrow B$: X	The transmission of data X from principal A to principal B
$Cert_A$	A public key certificate for a principal A
$Cert_{K_{pk}}$	A public key certificate for the public key K_{pk}
$D_K(X)$	The symmetric decryption of ciphertext X, using the secret key K
$\bar{D}_{K_{sk}}(X)$	The asymmetric decryption of ciphertext X, using the private key K_{sk}
$E_K(X)$	The symmetric encryption of plaintext X, using the secret key K
$\bar{E}_{K_{pk}}(X)$	The asymmetric encryption of plaintext X, using the public key K_{pk}
H	A cryptographic hash-function
ID_A	An identifier for a principal A
$\mathrm{MAC}_K(X)$	The message authentication code (MAC) of data X, computed using the key K
N	A nonce (a random value)
$Sig_K(X)$	A signature computed on data X using the private signature key K
t_A	A timestamp generated by principal A

Chapter 1

Introduction

Contents

This chapter gives an overview of the thesis. We begin the chapter by outlining the motivation for the research. We next describe the contributions of this thesis. The overall structure of the thesis is then presented, and the chapter concludes with a list of publications covering some of the results described in this thesis.

1.1 Motivation and Challenges

In his seminal paper published in 1991 [157], Weiser provided a glimpse of what to expect of computers in the 21st century. He envisaged that computers would disappear into the background, integrating seamlessly into the world and being woven into the fabric of our daily lives so that we are not even aware of their existence. He termed this new computing paradigm *Ubiquitous Computing.* Today, the evolution of wireless networking technologies has enabled a revolution in markets for network access and service delivery. Furthermore, it is no longer infeasible to embed computation and communication capabilities into virtually all manufactured objects. This means that the Weiser vision is fast becoming a reality.

The Mobile VCE Core 4 Research Programme on Ubiquitous Services[1] is designed

[1]see http://www.mobilevce.com/

to play a part in realising this vision. This project is based on the premise that, in future mobile ubiquitous environments, users (using their mobile devices and available network access technologies) will be able to seamlessly discover, select, and access a rich offering of services and content from a range of service providers. One of the aims of this research programme is to remove some of the barriers to the widespread adoption of such services. Until recently, relatively little attention has been given to the security challenges associated with delivering ubiquitous services over heterogeneous mobile wireless devices and networks, as most of the research effort has been devoted to addressing the challenges of physical layer connectivity. To truly realise the vision of ubiquitous services, it is vital that security and privacy issues are addressed from the outset, alongside other technological innovations.

In parallel with recent developments in mobile ubiquitous computing, a series of specifications have been produced by the Trusted Computing Group[2] (TCG), with the aim of enhancing the security of computing platforms. Trusted Computing involves incorporating trusted hardware functionality, including so called 'roots of trust', into computing platforms. Users can thereby gain greater assurance that a platform is behaving in the expected manner [11, 100, 151]. The trusted hardware involves a hardware component called the Trusted Platform Module (TPM) being integrated into a host platform during manufacture. The TPM provides the platform with a foundation of trust, as well as the basis on which a suite of Trusted Computing security functionality can be built. The TPM and its host are collectively referred to as a *Trusted Platform.*

Trusted Computing offers a collection of powerful security functionality which can be exploited to enhance the security and privacy of mobile ubiquitous environments or to secure mobile ubiquitous services. It is widely anticipated that trusted computing functionality will become ubiquitous in the not too distant future. This assumption is not unreasonable, as trusted computing functionality is increasingly being incorporated into computing platforms (mobile or otherwise). The aim of of this thesis is therefore to examine and identify ways in which trusted computing can be used to help secure ubiquitous services in dynamic and mobile environments.

[2]http://www.trustedcomputinggroup.org/

1.2 Contributions

The application of trusted computing to secure mobile ubiquitous computing and services is, to the best of our knowledge, a relatively new research area, in which very little prior research has been conducted. The main contributions of this thesis involve exploring ways in which trusted computing can be used to help provide security for ubiquitous services. These contributions can be summarised as follows.

- We analyse a possible privacy flaw in the TCG implementation of the Direct Anonymous Attestation (DAA) protocol, discovered by Rudolph. We argue that, in typical usage scenarios, this weakness is not likely to lead to a feasible attack; specifically we argue that the attack is only feasible if honest DAA signers and verifiers never check the behaviour of DAA issuers. We also suggest possible ways of avoiding the attack.

- We analyse a protocol proposed by Sevinç, Strasser and Basin, which uses Trusted Computing functionality to secure the distribution and storage of secrets from a server to a client. We identify two inherent security weaknesses in the protocol, namely, the absence of server-to-client authentication and the unauthenticated encryption of secrets sent from the server to the client. We show how, as a result of these weaknesses, the TPM could be exploited as a signing oracle, undermining the overall security of the scheme. We also propose possible ways of making the protocol more secure.

- We propose Ninja: a non-identity based, privacy preserving, mutual authentication scheme, designed to address the service discovery security and privacy challenges arising in a mobile ubiquitous environment. In this scheme, instead of authenticating the user identity to a service provider, the user's trustworthiness is anonymously authenticated. We also carry out a service discovery threat analysis and identify a corresponding set of security requirements. We use trusted computing functionality to develop the Ninja authentication scheme, and show how it achieves desirable security and privacy properties such as: user anonymity, service information confidentiality, unlinkability and rogue blacklisting.

- We propose the Secure Device Management Framework (SDMF), designed to securely deliver ubiquitous services to end user devices, whilst also hiding security management complexity from users. We conduct a service delivery threat analysis and identify a set of corresponding security requirements. Apart from providing secure service interactions, the framework is also designed to reduce the complexity of device security management tasks for users. Furthermore, the framework protects the interests of service providers by preventing unauthorised credential sharing amongst user devices. One other novel feature of the framework is that compromised devices are self-revoking, hence removing the need for a cumbersome revocation infrastructure. We achieve these objectives by incorporating Trusted Computing functionality into an entity known as the Device Management Entity (DME), and then integrating it with a number of other security mechanisms (such as the MANA protocol [75] and the Ninja service discovery protocol described in Chapter 7).

- We propose a privacy preserving, content distribution protection (CDP) watermarking scheme using trusted computing functionality. Our scheme, apart from being suited for a mobile environment, is designed to allow a buyer to anonymously purchase digital content, whilst enabling the content provider to blacklist the buyers that are distributing content in unauthorised ways. We also carry out a threat and requirements analysis. Our scheme minimises the reliance on a TTP for privacy protection, as the buyer can generate verifiable pseudonyms on its own. A second important feature of our CDP scheme is the ability for the content provider to obtain assurance that a buyer-generated watermark is well-formed. The scheme also provides the following security features: framing resistance, user anonymity, content information confidentiality, unlinkability (even against the TTP), and transaction linkability. We compare the security and performance of our CDP scheme against two other recently proposed schemes.

Although the Mobile VCE architecture provides the context for much of the work described in this thesis, the results are of much wider relevance. The security architectures and protocols are intended to be applicable to any analogous mobile ubiquitous system architecture.

1.3 Organisation of Thesis

The remainder of this thesis is divided into three main parts, and is organised as follows.

Part I — Background: This part consists of Chapters 2 and 3. Chapter 2 introduces the security concepts that are used throughout this thesis. We begin with an overview of the security threats that commonly apply to communication systems. We then list security services (or security objectives) that can be used to address these threats, and finally we introduce security mechanisms and cryptographic primitives that can be used to provide the various security services.

Chapter 3 introduces the concepts of *Ubiquitous Computing*, a *Mobile Ubiquitous Environment*, and *Ubiquitous Services*, and we describe a ubiquitous services scenario. We also identify the security threats that can arise in such environments. We then derive a set of security requirements that may be used to address the identified threats.

Part II — Security Issues in Trusted Computing: This part consists of Chapters 4, 5 and 6. Chapter 4 provides an overview of Trusted Computing technology. We introduce the concept of a trusted platform and describe its components and architecture. We then explain what the key features of trusted computing are, and how they work. We also discuss how trusted computing can be used to support user privacy. Finally we review some possible applications of trusted computing.

Chapter 5 analyses a possible privacy flaw in the Direct Anonymous Attestation (DAA) protocol. We briefly describe the flaw, and we then analyse the feasibility of attacks exploiting this flaw in a number of different scenarios. We also propose three possible ways in which such attacks could be prevented.

Chapter 6 analyses a TPM-based Secret Distribution and Storage Scheme. We briefly describe the scheme and then outline certain security weaknesses in it. We also propose possible modifications to the scheme designed to address the identified weaknesses.

Part III — Applications of Trusted Computing to Ubiquitous Services: This part, consisting of Chapters 7, 8 and 9, describes three novel security schemes. Chapter 7 presents Ninja, a secure and private service discovery scheme. We discuss service discovery security issues arising in a mobile ubiquitous environement, and we also review related research work. We then describe the operation of the scheme, and give an analysis of its security.

Chapter 8 presents SDMF: a framework for secure ubiquitous services delivery. We first review the security issues associated with service delivery. We next introduce the various building blocks that are used to develop the framework. We then describe the operation of the scheme and give an analysis of its security properties.

Chapter 9 presents a privacy preserving, content distribution watermarking scheme. We begin with an overview of digital watermarking technology and related research work, and discuss security issues arising from content distribution. We then introduce the building blocks employed in the scheme. We describe the novel watermarking scheme, analyse its security, and compare it with two previously proposed schemes.

Conclusions: In Chapter 10, the final chapter of this thesis, we provide concluding remarks and summarise the findings of this thesis. We also provide suggestions for possible future work.

1.4 List of Publications

Some of the results described in this thesis have previously been published. Relevant publications are listed below in chronological order.

1. Adrian Leung and Chris J. Mitchell. Towards Secure Zero Configuration. In *Proceedings of Western European Workshop on Research in Cryptography (WeWoRC 2005)*, Leuven, Belgium, July 5–7, 2005. pp. 34–36.

2. Adrian Leung and Chris J. Mitchell. A Service Discovery Threat Model for Ad Hoc Networks. In *Proceedings of the International Conference of Security*

and Cryptography (SECRYPT 2006), Setubal, Portugual, August 7–10, 2006. INSTICC Press, pp. 167–174.

3. Adrian Leung and Geong Sen Poh. An Anonymous Watermarking Scheme for Content Distribution Protection using Trusted Computing. In *Proceedings of the International Conference on Security and Cryptography (SECRYPT 2007)*, Barcelona, Spain, August 28–31, 2007. INSTICC Press. pp. 319–326

4. Adrian Leung and Chris J. Mitchell. Ninja: Non Identity Based, Privacy Preserving Authentication for Ubiquitous Environments. In *J. Krumm et al. (Eds): 9th International Conference on Ubiquitous Computing (UbiComp 2007)*, Innsbruck, Austria, September 16–19, 2007. Proceedings, volume 4717 of Lecture Notes in Computer Science, pp. 73–90. Springer-Verlag, Berlin, Heidelberg, 2007.

5. Adrian Leung, Yingli Sheng and Haitham Cruickshank. The Security Challenges for Mobile Ubiquitous Services. *Information Security Technical Report*, Elsevier, vol 12, no 3. pp. 162–171, 2007.

6. Adrian Leung, Liqun Chen and Chris J. Mitchell. On a Possible Privacy Flaw in Direct Anonymous Attestation (DAA). In *P. Lipp, A.-R. Sadeghi, and K.-M. Koch(Eds): Trust 2008*, Villach, Austria, March 11–12, 2008. Proceedings, volume 4968 of Lecture Notes in Computer Science, pp. 179–190. Springer-Verlag, Berlin, Heidelberg, 2008.

7. Adrian Leung and Chris J. Mitchell. A Device Management Framework for Ubiquitous Service Delivery. In *Proceedings of the Fourth International Conference on Information Assurance and Security (IAS 2008)*, Naples, Italy, September 8–10, 2008. IEEE Computer Society Press, Los Alamitos, CA, pp. 267–274.

8. Ronald Toegl, Georg Hofferek, Karin Greimel, Adrian Leung, Raphael C-W. Phan and Roderick Bloem. Formal Analysis of a TPM-Based Secret Distribution and Storage Scheme. In *Proceedings of the 2008 International Symposium on Trusted Computing (TrustCom 2008)*, Hunan, China, November 18–21, 2008. IEEE Computer Society Press, Los Alamitos, CA, pp. 2289–2294.

9. Adrian Leung. A Mobile Device Management Framework for Secure Service Delivery. *Information Security Technical Report*, Elsevier, vol 13, no 3. pp.

118–126, 2008.

10. Adrian Leung, Po-Wah Yau and Chris J. Mitchell. Using Trusted Computing to Secure Mobile Ubiquitous Environments. In *Security and Privacy in Wireless Networking* (Troubador Publishing Ltd., Leicester, UK, 2009), edited by S. Gritzalis, T. Karygiannis and C. Skianis.

Part I

Background

Chapter 2

Security Preliminaries

Contents

This chapter introduces the security concepts and techniques used throughout the thesis. We begin by introducing security threats arising from open communication systems. We proceed to identify corresponding security services, and, finally, we describe security mechanisms that can be used to provide these services.

2.1 Introduction

In this chapter we provide definitions of fundamental security concepts and techniques used throughout the thesis.

Before providing the definitions, we first establish the meanings of certain fundamental terms, namely *security threats*, *security services* and *security mechanisms*. As given in Dent and Mitchell [45]:

"a *security threat* is something that poses a danger to the security of a system. A *security service* is selected to meet an identified threat, and a *security mechanism* is the means by which a service is provided or implemented."

2.2 Security Threats

In this section, using the *STRIDE* threat model developed by Swiderski and Snyder [145], we describe six of the most common types of security threats affecting open communication systems.

1. **Spoofing:** This involves an entity masquerading as another entity, often assuming the identity of the masqueraded entity. One commonly occurring example of spoofing involves unauthorised use of another individual's username and password to engage in online transactions.
2. **Tampering:** Data tampering involves the malicious modification of data. Examples include unauthorised changes made to persistent data (e.g. data held in a database), or the alteration of data in transit (e.g. data flowing between two computers over an open network, such as the Internet).
3. **Repudiation:** A repudiation threat involves entities denying that they performed an action after having performed that action. For example a user may instruct his/her broker to sell shares, but after the share price has gone up he/she denies having given the instruction.
4. **Information Disclosure:** Information disclosure threats involve the exposure of information to unauthorised entities. For example, an unauthorised third party might eavesdrop on and capture data exchanged between two endpoints over a wireless communication link.
5. **Denial of Service:** A Denial of Service (DoS) threat or attack involves preventing access to services by legitimate users. For example, a malicious entity may mount a DoS attack on a web server by making numerous requests to the web site. This may prevent legitimate users from making transactions, as the attack may cause the web site to experience delays or even become unavailable.

6. **Elevation of Privilege:** In this type of threat, an unprivileged (or unauthorised) user gains privileged access to a system or network. As a result the entity may be able to mount a variety of attacks on the system, or even compromise the entire system if sufficient privileges are gained. For example, if an unauthorised user gains root access to a system, then almost any threat could be realised to that system and its data.

2.3 Security Services

In this section we define security services that can be used to counter the threats outlined in Section 2.2. The definitions below are adapted from Dent and Mitchell [45] and ISO 7498-2 [71].

1. **Authentication:** this service can be sub-divided into two types, namely *entity authentication* and *data origin authentication*:
    - *entity authentication* is defined as the corroboration that the entity at the other end of a communications channel is the one claimed;
    - *data origin authentication* is the corroboration that the source of data received is as claimed.
2. **Data Confidentiality:** this service is concerned with preventing the disclosure of data to unauthorised entities.
3. **Data Integrity:** this service is concerned with preventing unauthorised alteration or destruction of data by an unauthorised entity (in other words unauthorised tampering with data).
4. **Non-Repudiation:** this service is concerned with preventing denial by an entity that it has taken a particular action, e.g. sending or receiving a message.
5. **Access Control:** this service is concerned with preventing unauthorised use of a resource.
6. **Availability:** this service ensures that computer system assets are available to authorised parties when needed.

Table 2.1 shows which of the defined security services can be used to address the security threats listed in Section 2.2.

Table 2.1: Security Threats and Security Services

Security Threats	Security Services
Spoofing	Entity Authentication
Tampering	Data Origin Authentication and Data Integrity
Repudiation	Non-Repudiation
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Access Control

Whilst there exist a number of other security services that are defined elsewhere in the literature, these six categories of security services are the ones most relevant to this thesis.

We next introduce a number of other terms relating to privacy and anonymity. We use the definitions of Pfitzmann and Hansen [117]:

1. **Privacy:** *Privacy* is the right of individuals to control or influence what information related to them may be collected and stored by whom, and to whom that information may be disclosed [71].
2. **Anonymity:** *Anonymity* of a subject means that the subject is not identifiable within a set of subjects, known as the anonymity set.
3. **Unlinkability:** Two or more items of interest (IOIs), e.g. subjects, messages, actions, etc., are *unlinkable* for a subject if that subject cannot distinguish whether or not the IOIs are related (where the meaning of the term 'related' will depend on the context).
4. **Pseudonym:** A *pseudonym* is an identifier for a subject with no semantic content, i.e. with no relation to any of the other identifiers for that subject. The subject to which the identifier refers is known as the *holder* of the pseudonym. A subject is said to be *pseudonymous* if a pseudonym is used to identify it.

2.4 Security Mechanisms

In this section, we introduce security mechanisms that can be used to provide the security services outlined in Section 2.3. Many of the definitions are taken from Menezes, van Oorschot and Vanstone [99].

2.4.1 Symmetric Encryption

Symmetric, or secret key, encryption mechanisms can be used to provide confidentiality services, and can be defined as follows. We define an encryption scheme to consist of a set of encryption transformations $\{E_e : e \in \mathcal{K}\}$ and a corresponding set of decryption transformations $\{D_d : d \in \mathcal{K}\}$, where $\mathcal{K}$ is the keyspace. The encryption scheme is symmetric if, for each associated encryption/decryption key pair (e, d), it is computationally easy to determine both e from d and d from e [99]. In most practical symmetric systems, e is essentially the same as d. In other words, in a symmetric encryption system, the same key is used to encrypt and decrypt messages.

Commonly used symmetric encryption schemes can be divided into two main types, namely *block ciphers* and *stream ciphers*:

- A *block cipher* has the property that the encryption algorithm operates on a block of plaintext, i.e. a string of bits of a defined length, to yield a block of ciphertext. Examples of block ciphers include the Data Encryption Standard (DES) [105], and the Advanced Encryption Standard (AES) [44, 106].
- A *stream cipher* is an encryption scheme in which the plaintext is encrypted bit by bit [120] using a pseudorandom sequence of bits known as a keystream, generated as a function of a secret key. Examples of stream ciphers include the RC4 algorithm [129] and the A5 family of algorithms used in GSM systems [1, 2, 156].

One major problem associated with symmetric encryption schemes, as with all symmetric cryptographic algorithms, is how to agree upon keys securely and efficiently.

This *key management* problem can be made somewhat simpler by adopting asymmetric cryptographic schemes, for which there is no requirement to maintain the secrecy of distributed keys (although guaranteeing their integrity and origin remains fundamentally important).

2.4.2 Cryptographic Hash Functions

A cryptographic hash function (or simply a hash function), also often referred to as a one-way hash function, can be used to help provide integrity and authentication services. The use of a hash function does not involve any secret keys, and such functions are typically used in conjunction with other security mechanisms. A *hash function* is a computationally efficient function mapping arbitrary length binary strings to fixed length binary strings [99]. The fixed length output of a hash function is often referred to as a message digest, hash code, or hash value.

Cryptographic hash functions must possess the following three properties [99]:

1. **Pre-image resistance:** for any pre-specified output, it is computationally infeasible to find an input which maps to that output.
2. **Second pre-image resistance:** for any pre-specified input, it is computationally infeasible to find a second input which yields the same output.
3. **Collision resistance:** it is computationally infeasible to find two inputs which map to the same output.

Examples of cryptographic hash functions include MD5 [128] (16-byte output), SHA-1 [107] (20-byte output), SHA-256 [107] (32-byte output), and the RIPEMD family [74, 122].

2.4.3 Message Authentication Codes

Messages Authentication Code (MAC) functions, also known as keyed hashed functions, can be used to simultaneously provide data origin authentication and integrity services.

A MAC function takes as input a message and a secret key, and outputs a fixed length binary string commonly referred to as the MAC value (or simply the MAC). This MAC value is then sent or stored with the message. When the origin and integrity of a message are to be checked, the verifier recomputes the MAC using the message and the secret key as input, and accepts the message as valid if the newly computed MAC agrees with the value sent or stored with the message. The verifier is thereby assured that (i) the message has not been tampered with, and (ii) it originates from the claimed sender.

In order to achieve confidentiality, integrity and origin protection for a message, it is common practice to first encrypt the message using one secret key, and then compute a MAC on the encrypted data using a second secret key. The order of the two operations (i.e. encryption prior to MAC computation) is important to achieve protection against side channel attacks.

Examples of MAC functions include CBC-MACs [72] (based on block ciphers) and HMAC [73] (based on hash functions).

2.4.4 Asymmetric Encryption

Asymmetric, or public key, encryption can be used to provide confidentiality services. An asymmetric encryption scheme is defined to consist of a set of encryption transformations $\{E_e : e \in \mathcal{K}\}$ and a corresponding set of decryption transformations $\{D_d : d \in \mathcal{K}\}$, where $\mathcal{K}$ is the keyspace. For any encryption/decryption key pair (e, d), it is computationally infeasible to compute d from e (unlike symmetric schemes, where d and e are essentially the same). In practice, the encryption keys e are made widely available (i.e. they are public keys), whereas the decryption (private) keys are kept secret by their owners.

One practical issue with the use of asymmetric encryption schemes, as with any asymmetric cryptographic scheme, is the problem of distributing public keys in such a way that the recipients can verify that they are genuine. The most widely used way of addressing this problem is to use public key certificates, discussed in Section 2.4.7.

Examples of asymmetric encryption schemes include the RSA-OAEP scheme [22, 76], which is based on the Rivest-Shamir-Adleman (RSA) scheme [127, 131], and the ElGamal encryption scheme [49].

2.4.5 Digital Signatures

A digital signature can be used to provide authentication, non-repudiation, and integrity services. A digital signature scheme involves two transformations, a *signing procedure* and a *verification procedure.* The signing procedure takes as input the message and secret information held by the signer A, and outputs a signature. If $\mathcal{M}$ is the set of messages to be signed, and $\mathcal{S}$ is the set of signatures, then a signing transformation S_A maps M to S. S_A is kept secret by A, and is only used for the purpose of creating signatures. V_A is the corresponding verification transformation, which maps the set $\mathcal{M} \times \mathcal{S}$ to the set $\{true, false\}$. V_A is typically widely disseminated and can be used by other entities to verify signatures created by A.

In practice, S_A and V_A are determined by a private and public key, respectively. That is, S_A is determined by a private key sk which is kept secret, and V_A is determined by a public key pk which is made public.

Examples of digital signature schemes include the RSA signature scheme (p.433 of [99]), the Digital Signature Algorithm (p.451 of [99]), and the Fiat-Shamir signature schemes (p.447 of [99]).

2.4.6 Key Agreement

A third class of asymmetric cryptographic schemes is provided by the key agreement schemes; one well known example of such a scheme is the Diffie-Hellman (DH) protocol [47]. This protocol is designed to establish a shared secret between two principals who share no information with each other in advance. We provide a brief description of the protocol (the interested reader is referred to Chapter 5 of Boyd and Mathuria [26] for a more comprehensive discussion).

1. Two principals, A and B, publicly agree on two elements g and p, where p is a large prime chosen such that $p-1$ has a large prime factor q (e.g. $p = 2q+1$), and g is chosen to have multiplicative order q. Thus q generates a multiplicative subgroup of $\mathbb{Z}_p^*$ of prime order q. This step is typically performed once prior to practical use of the scheme, and the values can be shared across a broad domain (for example, the values may be built into the implementation of the scheme).

2. A and B generate random values a and b, respectively ($1 < a < q$; $1 < b < q$).

3. A computes $g^a \bmod p$ and B independently computes $g^b \bmod p$.

4. A and B exchange $g^a \bmod p$ and $g^b \bmod p$, and can thereby independently derive the shared key $K_{AB} = (g^a)^b \bmod p = (g^b)^a \bmod p$.

Unless additional security mechanisms are used, the Diffie Hellman protocol is susceptible to man-in-the-middle attacks, as the messages exchanged between the principals are not authenticated. In other words, an attacker, C, could masquerade as A to B, and simultaneously masquerade as B to A, agreeing different keys with each.

2.4.7 Public Key Infrastructures

As discussed briefly in Section 2.4.4, one fundamental problem for the use of asymmetric cryptography is the reliable distribution of public keys. One way of addressing this problem is through the deployment of a Public Key Infrastructure (PKI).

In a PKI, a trusted third party known as a *Certification Authority* (CA) is responsible for issuing (digitally signing) public key certificates, which bind a public key to the name of its owner (together with other relevant information). The user of a certificate (the *relying party*) must have access to a trusted copy of the public key of the CA that issued the certificate in order to verify the signature on the certificate. Such CA public keys are often referred to as *root public keys* (since they are the root for the trust derived from the use of certificates).

A widely used standard format for such certificates is contained in ITU-T Recommendation X.509 [77]; the most recent version of this document specifies what are known as version 3 (v3) certificates. Such a certificate contains the following data fields:

- Certificate
  - Version
  - Serial Number
  - Algorithm ID
  - Issuer Name (i.e. name of certificate issuer)
  - Validity
  - Subject (i.e. user name or ID)
  - Subject Public Key
- Certificate Signature Algorithm
- Certificate Signature

Certificates may need to be revoked before their expiry for a variety of reasons, e.g. because the private key has been compromised or because the subject has left the relevant organisation. Certificate users (so called *relying parties*) need to have access to current information regarding which certificates have been revoked, and this can be achieved in two main ways. A CA can regularly issue a *Certificate Revocation List* (CRL) listing all the certificates which it has issued that have been revoked — relying parties would then need to ensure they have the most recent CRL before relying on a certificate. Alternatively, an online trusted third party can answer queries regarding the status of a certificate, e.g. using a protocol such as the Online Certificate Status Protocol (OCSP) [104].

Two CAs may *cross-certify* one another, i.e. issue public key certificates for each other's public keys. This will enable the holder of a trusted copy of one CA's public key to obtain a trusted copy of the public key of another CA. For further details regarding the use and management of PKIs, see, for example, [8, 142].

Chapter 3

Mobile Ubiquitous Computing

Contents

This chapter introduces the concepts of a mobile ubiquitous environment and a ubiquitous service, as envisaged within the Mobile VCE Research Programme. The chapter also explores and discusses the various security issues and challenges that are associated with a mobile ubiquitous environment. We conclude by identifying possible security requirements for ubiquitous services.

3.1 Introduction

In 1991, Weiser provided a radical new vision of computing in the 21st Century [157]. He envisaged that computers would disappear into the background, integrating seamlessly into the world and being woven into the fabric of our daily lives so much that we are not even aware of their existence [157]. He termed this new paradigm Ubiquitous Computing. In his seminal paper [157], he identified three key technological requirements for the realisation of Ubiquitous Computing. They are: (a) cheap, low power computers that include equally convenient displays; (b) a network that ties them all together; and (c) software systems implementing ubiquitous applications.

Eighteen years on, the notion of Ubiquitous Computing is still very much alive, especially amongst the research and academic community. Almost every university and research facility around the world is conducting research on ubiquitous computing or on other closely related, if not identical, topics (e.g. pervasive computing, calm computing, or ambient computing). The number of conferences and workshops devoted to this field has also increased considerably, including Ubicomp, Pervasive, IEEE Percom, etc. On the technological front, low cost, low power mobile devices are very rapidly proliferating. This is fuelled by the fact that Moore's law, i.e. that computing power will double every eighteen months [103], is still holding true, coupled with advances in wireless networking technologies (such as Bluetooth, 802.1x, Zigbee, Ultra-Wideband, etc.). As a result, it is no longer infeasible to embed computation and communication capability into virtually all manufactured objects. Furthermore, we are also seeing a growing number of software systems (increasingly from the open source community) capable of performing a variety of sophisticated and complex tasks. These facts together mean that Weiser's requirements are now achievable.

Despite Weiser's requirements being met, there is still some way to go before the vision of ubiquitous computing can be realised. The Mobile VCE research consortium[1], through its Core 4 Research Programme on Ubiquitous Services, aims to help realise the Weiser vision. It is undertaking a three-year research programme aimed at enabling the widespread adoption of ubiquitous services amongst end users and consumers in the future digital and highly mobile market place. This research programme has identified trends and issues which must be addressed before ubiquitous computing can become a reality. Firstly, it was noted that much existing research focuses solely on enhancing computing power, network infrastructure and device capabilities; very little focus has been given to the needs of the end user. Sec-

[1]http://www.mobilevce.com/

ondly, there has been a gradual shift from a device and network-centric paradigm to a service-oriented and user-centric paradigm. Lastly, in a user-centric and service oriented environment, security is paramount. End-users are likely to want assurance that their interactions are secure before they will adopt new technology. For security solutions to be truly effective, they must be addressed from the outset alongside any technological innovations, and not be added as an afterthought. Experience suggests that addressing security breaches can be very costly; prevention is almost always better than cure. The objective of the project is therefore to remove the barriers to the commercialisation and deployment of ubiquitous applications and services in a simple and secure manner. Of the various aspects of the project, we focus here on the secure delivery of ubiquitous services to end users.

The remainder of this chapter is organised as follows. The notions of a mobile ubiquitous environment and a personal distributed environment are introduced in Sections 3.2 and 3.3 respectively. We then describe a simple ubiquitous services scenario in Section 3.4. In Section 3.5 we identify the security threats applying to the delivery of ubiquitous services, and in Section 3.6 we derive a set of ubiquitous services security requirements. These requirements provide the context for much of the work described in the remainder of this thesis.

3.2 A Mobile Ubiquitous Environment

The Mobile VCE Ubiquitous Services project envisages that in a mobile ubiquitous environment, users (via their mobile devices and using available network access technologies) will be able to seamlessly discover and access a rich offering of services and content from a wide variety of service and content providers (as shown in Figure 3.1).

As shown in Figure 3.1, a mobile ubiquitous computing environment has the following characteristics:

- users own multiple devices of varying capabilities,
- users (and their devices) are highly mobile, and
- users communicate using a variety of different wireless network access tech-

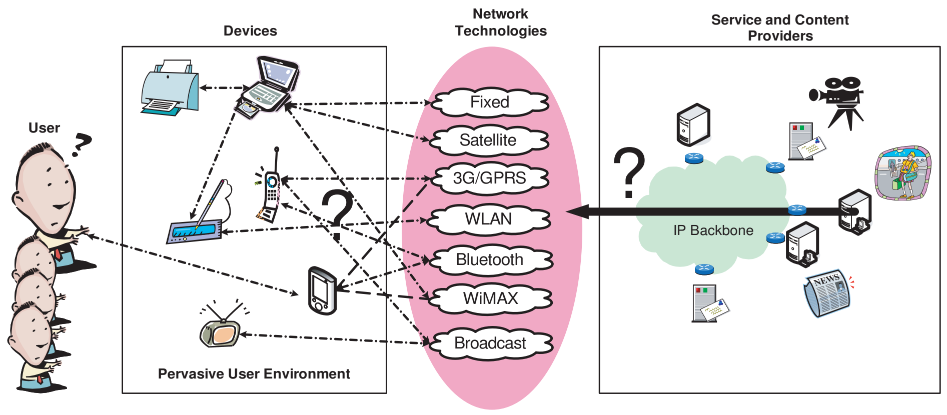


Figure 3.1: A Mobile Ubiquitous Computing Environment

nologies.

Such an environment can be considered from three key perspectives, namely: the *User* perspective, the *Network* perspective and the *Service Provider* perspective. One of the aims of the Ubiquitous Services project is to address the ubiquitous services security issues arising from these three perspectives. We briefly introduce each of the perspectives below in relation to the relevant security issues. Note that, in the rest of this thesis, very little attention will be given to the security issues emanating from the network perspective. Instead we focus on the application layer security issues, that is security issues concerning the two classes of end-parties, namely users and service providers.

1. **User Perspective:** From the user perspective, the fundamental requirement is a simple and consistent means for service access. The Internet today is heterogeneous, with network components having widely varying capabilities (bandwidth, interface speed, edge-to-edge latency and connection type). Also, the use of laptops, palmtops and mobiles has increased, and so has the variety of operating systems. This has led to a great diversity in end system capabilities. This diversity, combined with the huge amounts of information and services available via these technologies, results in an undesirable level of complexity. Inevitably, in this new environment users are exposed to a wide variety of security threats and attacks. It is therefore imperative that the security and privacy of users are protected in this landscape.

2. **Network Perspective:** From the network perspective, providing high levels of security, quality of service (QoS) and mobility to participating networks are all important, along with maintaining an acceptable level of network performance. Heterogeneity has made simultaneously maintaining security, QoS, and mobility management a complex issue. These three requirements inherently conflict, since they compete for system resources. For example, increasing security, e.g. by using stronger encryption techniques, will potentially increase resource usage. Better mobility management mechanisms, such as providing more frequent paging and routing updates, will also increase resource usage. Similarly, providing a good QoS is likely to have a similar effect on resource allocation. A balance must therefore be struck between the three requirements. Limitations on resources mean that conflicts between security, QoS, and mobility management will always be present. Determining the optimal service point is crucial, and remains a challenge (see Figure 3.2). One aim of the Mobile VCE research programme is to devise a common network framework which will efficiently integrate security, QoS and mobility functions.

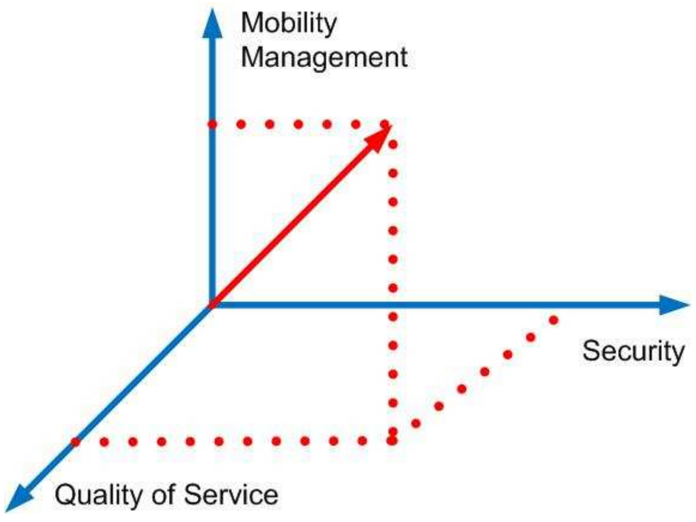


Figure 3.2: Balancing Security, QoS and Mobility

3. **Service Provider Perspective:** From the content and service provider perspectives, multiple co-operating networks promise the possibility of a range of new ubiquitous services, including context-aware services. The need for content and service providers to support multimedia applications over a range of access network technologies (including ad hoc), each with its own capabilities, represents a particular challenge to service delivery. Security is also important

to protect the interests of the service provider against malicious users and other adversaries. One particular issue that will be dealt with in greater detail in this thesis is content distribution protection.

3.3 Personal Distributed Environments

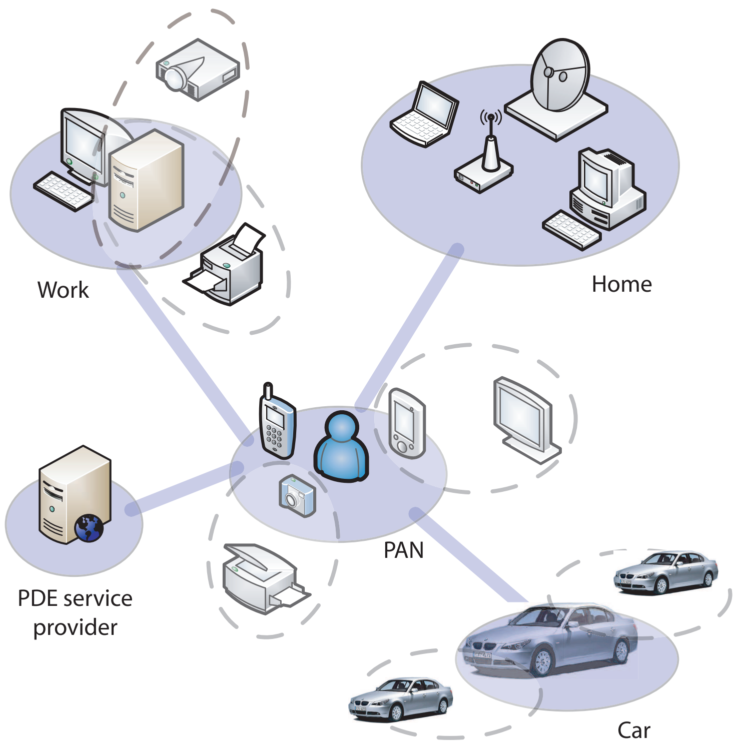
A *personal distributed environment* (PDE) [10] is an overlay networking concept that allows a user to form a 'virtual network' consisting of the networked devices that the user owns, or is authorised to use. The capabilities of these devices will typically vary in terms of computational power, energy source, mobility, communication and network interfaces, and they may reside in different physical locations. Other overlay architectures, similar to the PDE concept, include those described in [101, 109, 110].

3.3.1 PDE scenario

Figure 3.3 illustrates an example of a PDE consisting of devices that the user owns, 'home devices', and also third-party devices that the user may or may not have pre-established rights to access — these are called 'foreign devices' in PDE terminology.

For example, a user may have a home network consisting of a desktop computer, a printer and an ADSL connection for Internet access. Also at home might be the user's digital set-top box and satellite receiver used to receive digital broadcasts. At work, the user might also have a desktop computer connected to the corporate network, allowing access to foreign devices such as corporate servers, applications, etc., that the user is authorised to access and use, possibly remotely.

In addition to the home and work environments, a user may also have a number of mobile environments. The user's car may have an intelligent networking capability enabling it to retrieve satellite navigation data and traffic news via satellite or ad hoc network communications. A user may also carry smaller devices, such as a mobile phone, PDA, and laptop — these home devices might be able to communicate to form a Personal Area Network (PAN). It is further envisaged that additional foreign devices could connect to the PAN on an ad hoc basis. These could, for example,



The shaded ovals represent user subnetworks, and the ovals with dashed outlines show foreign devices that are connected to the user's PDE.

Figure 3.3: A Personal Distributed Environment

include a display monitor on a train, allowing a user to view content on a larger screen, a printer in an airport lounge, or the devices of another user's PDE used to transfer data or play computer games.

These example illustrate how heterogenous networks and devices can be integrated to form a PDE. While this results in additional complexity, the diversity of technologies can be used to deliver ubiquitous services to the user, where the most capable devices and communications means available are selected.

3.3.2 PDE management

Given the potential complexity of a PDE, users need a simple and intuitive means to manage it. This is achieved through a semi-distributed, logical construct called the *Device Management Entity* (DME) [10]. The primary task for the DME is to intelligently select the most appropriate device for service delivery, acting as a Session Initiation Protocol (SIP) [130] proxy to redirect session setup requests. The DME is also designed to abstract the technical complexities faced by users (especially non-expert users) in managing (i.e. configuring and operating) their devices in a mobile ubiquitous environment. The DME acts as an interface between user devices and external entities. In order to perform this task, the DME requires the following functional components:

- The **Features Register** maintains information about the capabilities (storage capabilities, screen resolution, processing power, battery life, networking functions, operating system, installed software, etc.) of each device in a user's Personal Distributed Environment. This information assists the DME when deciding which device is most appropriate for use in accessing specific services or content.
- The **Location Register** maintains location information about user devices. It can be used to determine whether devices are reachable via the access service.
- The **Security Register** maintains security credentials on behalf of users. These credentials could include cryptographic keys, access tokens, and public key certificates. The security register also stores information used to support PDE security functionality.

As stated previously, a user's PDE may consist of a number of subnetworks in disparate locations. For reasons of efficiency and scale, DME functions are distributed to a 'local DME' within a subnetwork or a user's personal area network. Local DMEs then report to a root DME. This effectively creates a two-layer management hierarchy [10].

It is suggested that the root DME should normally reside on the most capable device in a static subnetwork, such as the user's home desktop or, perhaps preferably, an online PDE service provider that the user has subscribed to [137]. Since the DME provides the features that enable a PDE, ensuring its availability is extremely important.

Within each PDE subnetwork, the most capable device is chosen as the local DME. In the case of mobile subnetworks the local DME selection process is dynamic, since one of the fundamental factors influencing capability is residual energy. Atkinson et al. [10] have proposed a broadcast-based protocol to discover the local, home, DME-capable device with the most residual energy — also included in their proposal is a handover protocol to move control of the local DME between devices.

3.4 A Ubiquitous Services Scenario

We now describe a simple ubiquitous services scenario, showing how a ubiquitous services consumer, Jose, is able to receive digital content while he is on the move. This scenario serves to illustrate the security threats and requirements applying to the delivery of ubiquitous services.

The sequences of events are as follows (also depicted in Figure 3.4):

1. Liverpool and AC Milan are playing in the finals of the 2005 UEFA Champions League. Jose, a diehard Liverpool fan, wants to watch the finals live, but he also has another appointment, and needs to be on the move while the match is taking place.

2. While at home, he contacts his content provider (Sky TV). The content (i.e. the football match) is streamed to his set-top box. After twenty minutes, Liverpool are 2-0 down. Jose then has to leave home for his appointment. He is also a subscriber to mobile broadcast content so that he is able receive both broadcast and 3G football content.

3. He takes a taxi to the train station. On the way to the station, he books a train ticket using his mobile device, and pays for it using his credit card. Jose arrives at the train station and immediately boards the train. On the train, he uses his mobile device (via UMTS or 3G connection) to access the football content (or goals). This connectivity is enabled by his mobile network subscription. AC Milan scores another goal, and Jose is able to watch a clip of the goal. Liverpool are 3-0 down at half time.

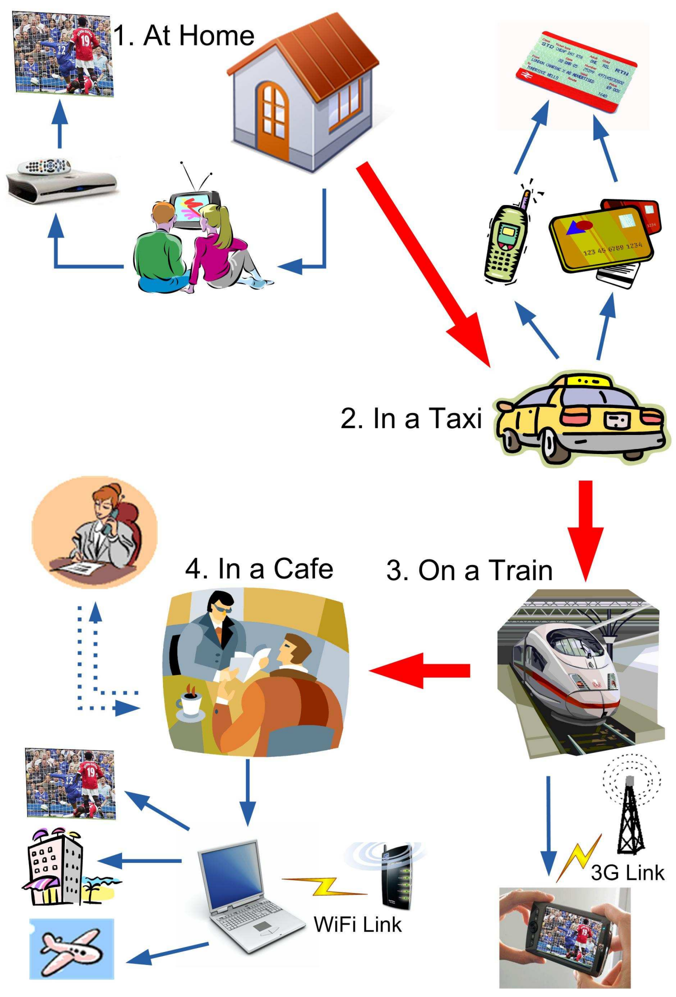


Figure 3.4: A Ubiquitous Services Scenario

4. Jose reaches his destination, and meets his friend Alex (also a Liverpool fan) at a café. The game resumes. Jose continues watching the game with Alex, using the free WiFi service provided by the café (a WiFi hotspot). Liverpool then score a goal, followed by another, and yet another. The score is now 3-3. Jose and Alex are ecstatic. The scores are level at full-time, and there will now be a period of extra-time. Jose receives a call from his wife asking him to book flight tickets and a hotel for a planned trip to Portugal. He books the flight and hotels online using his mobile device (via GPRS). Liverpool go on to win the match in a penalty shootout. Jose and Alex are delirious.

3.5 Ubiquitous Services Security Issues

In this section, we examine the security threats from two of the previously discussed perspectives of a mobile ubiquitous environment, namely, the User and the Service Provider Perspectives. To better illustrate the cause and effect of these security threats, they are also discussed, whenever possible, in relation to the ubiquitous services scenario described in Section 3.4.

3.5.1 User Perspective

The ubiquitous services security threats relevant to a user are as follows:

1. **Spoofing.** A malicious entity may masquerade as a legitimate service provider with the aim of luring a user into a bogus service interaction. For example, in the above scenario, Jose may wish to be sure that he is indeed interacting with his "real" service provider, the SKY TV, and not some bogus entity.
2. **Information Disclosure.** A user's personally identifiable information (PII) (e.g. identity, credit card information, physical location, etc.) may be disclosed to a service provider or passive eavesdropper whilst the user is interacting with a service provider. Possible consequences of this threat include loss of privacy for a user or identity theft. In our scenario, an adversary may steal Jose's credit card details when he is buying the train or flight tickets.
3. **Profiling.** From service interactions that a user has previously had with a service provider, this service provider may be able to learn valuable information about the user (e.g. age, gender, income, habits, spending patterns, etc.), and build up a profile of the user. This might then be used for future targeted marketing purposes. This could compromise user privacy. For example, Jose may frequent a particular supermarket and use a loyalty card when making purchases. The loyalty card allows the supermarket to build up a profile of Jose's spending habits.
4. **Profile Linking.** Service providers may collude to link the profiles and activities/transactions of a user, in order to build a more complete user profile.

The information may subsequently be used to infer the future behaviour of a user. User privacy may hence be compromised. In our scenario, the hotel, airlines and car-hire company might exchange information about Jose.

5. **Framing.** A variety of content distribution protection (CDP) (e.g. watermarking, fingerprinting, etc.) and digital rights management (DRM) mechanisms are employed by content providers to deter (or prevent) users from redistributing proprietary content. Most of the existing CDP or DRM solutions, however, do not protect an honest content user from being falsely accused (or framed) by a content provider of unauthorised content distribution.

6. **Malware Infection.** Unbeknownst to Jose, malicious software (e.g. Spyware, Keystroke Loggers, Trojans, etc.) may be running on his device and harvesting his personal information, such as passwords, PINs, and credit card information.

7. **Information Overloading.** A user may be deluged with a huge amount of service information (in the form of service advertisements) from prospective service providers or spammers. This could lead to two sub-threats:

    (a) **Denial of Service:** A user's device may be flooded with service advertisements (both legitimate and bogus), which may in turn prevent the user from having genuine service interactions.

    (b) **Service Selection Dilemma.** It is highly likely that a sought service may be offered by more than one service provider. When presented with many choices, and having had no prior interactions with any of the service providers, a user may not be in a position to select a service that best suits his/her needs. For example, suppose Jose arrives in a city that he has never visited before and he wishes to find a restaurant. There may be more than twenty restaurants in the vicinity, and he will not know which to choose.

8. **Configuration Complexity.**

    (a) **Device and Application Settings:** A user can own many different types of devices (e.g. laptop, PDA, mobile phone, iPod, etc.). Different devices may be used to access a service, depending on the situation/context, the physical location that the user is in, and/or the type of service that he/she intends to use. Before a service can be used, the access device will first have to be appropriately configured. For a non-expert user this can be an extremely daunting task. A wrongly configured device may even pose a security risk to both the user and the service providers. For example, Jose may unknowingly or accidentally disable the virus protection services on his device.

(b) **Security parameters:** More often than not, a user will pick passwords that are memorable. These passwords are usually very weak. Furthermore, for simplicity, a user may use the same set of user names and/or passwords/PINs for many different services. This makes it far more likely that a user password will fall into malicious hands, and the end user's security could be seriously compromised as a result.

3.5.2 Service Provider Perspective

The threats that may affect a service provider are listed below:

1. **Spoofing.** A malicious entity may masquerade as a legitimate user and interact with a service provider. For example, a hacker may use Jose's account (e.g. username and password) and purchase digital content without the knowledge and permission of Jose.

2. **Service Repudiation.** The payment process (or other business process) may be damaged if a service user can later deny having used a service or consumed content.

3. **Non-Payment.** Service providers will typically be concerned about the receipt of correct payment from a service user. Mobile payment systems are not yet widely available, as many issues (both legal and technical) still exist. At the time of writing, practical anonymous payment schemes are still not available.

4. **Unauthorised Content Distribution.** Unauthorised distribution of proprietary content (e.g. music, films, etc.) poses a major challenge for the digital content industries. This directly translates to a loss of revenue for the content

providers. The mobile and dynamic nature of a ubiquitous environment makes it easier to redistribute content in unauthorised ways.

5. **Rogue Behaviour.** Privacy enhancing technologies may be employed by users to protect their privacy when they are interacting with service providers. If these users misbehave (e.g. by redistributing proprietary content), then service providers may not be able to trace them.

3.6 Security Requirements

Building on the threat analysis in Section 3.5, we next derive a set of ubiquitous services security requirements. We first identify general security requirements, followed by specific security requirements for the User and Service Provider perspectives.

3.6.1 General Security Requirements

The 'standard' set of general security requirements, also described in Section 2.3, are:

1. Authentication;
2. Data Confidentiality;
3. Data Integrity;
4. Non-Repudiation;
5. Access Control, and
6. Availability.

3.6.2 Specific User and Service Provider Security Requirements

Since user and service provider interactions occur at the application layer, the user and service provider security requirements are described together, as follows:

1. **Secure Service Selection/Recommendation.** When more than one service provider is offering a service, a selection or recommendation (e.g. reputation, ranking) mechanism may be employed by a user to assist in the decision-making process. The selection agent may take as input a user's preferences and habits, along with context information. This process should be secured to prevent a malicious entity (e.g. a competing service provider) manipulating the rankings.

2. **Secure Zero Configuration.** Device or network settings (e.g. IP address, DNS) may automatically be configured on behalf of a user through a process known as Zero Configuration [159]. This process needs to be secured.

3. **Secure Service Discovery.** Before a service can be used, it needs to be located through a process known as service discovery. It is important that the process of service discovery (between a user and a service provider) is secured, or otherwise the security and privacy of a user and the service provider could be compromised (e.g. an eavesdropper may be able to determine the services that a user is looking for).

4. **Secure Service Provision.** The delivery of a service to an end user is known as service provision. It is imperative that this process is secured in order to guarantee that the correct (unmanipulated) service is being delivered to the intended recipient (user).

5. **Privacy Preserving (Mutual) Authentication.** The above-mentioned security requirements normally involve interactions between a user and a service provider. These two entities will need to be mutually authenticated to prevent possible spoofing threats (in either direction). However, if privacy is desired by the user, then privacy preserving (or anonymous) authentication mechanisms may be required by the user. A service provider can still be authenticated to a user in the conventional way, as service providers do not normally require privacy/anonymity, and, for obvious reasons, would typically want to be known by as many users as possible.

6. **Content Distribution Protection (CDP).** A service provider may wish to prevent and/or detect unauthorised distribution of content. For example, a service providers may employ content distribution protection mechanisms

(e.g. watermarking, fingerprinting, or other DRM solutions) to prevent or deter users from distributing proprietary content (which may have been legally purchased earlier) in unauthorised ways. To prevent a content buyer from being accused of unauthorised content distribution, Buyer-Seller watermarking schemes (which requires the content buyer to contribute a watermark) may be used.

7. **Anonymous Payment Schemes.** Users may wish to pay for services anonymously.

8. **Privacy and Anonymity.** Significant number of users are likely to be concerned about their privacy while transacting online. In general, the abovementioned security requirements should collectively provide users with assurance that their privacy is not being compromised. Privacy is often achieved through anonymity, for example through the use of pseudonyms when interacting with other entities. This leads us to another sub-requirement, that of Unlinkability; that is, users may not want their transactions with different service providers to be linkable.

3.7 Summary

In this chapter we introduced the concept of mobile ubiquitous computing. In so doing, we introduced the notions of a mobile ubiquitous environment, a personal distributed environment and a ubiquitous service. We also identified the security threats that may arise in a mobile ubiquitous environment. We then derived the security requirements for such an environment.

Part II

Security Issues in Trusted Computing

Chapter 4

Trusted Computing

Contents

This chapter provides an overview of Trusted Computing technology. We begin with a description of the motivation for Trusted Computing, and introduce the concept of a Trusted Platform. We examine the various building blocks that make up a Trusted Platform and then study the features and functionality of Trusted Computing. We *show how user privacy is supported by Trusted Computing, and conclude the chapter with a review of current research into, and possible applications of, Trusted Computing.*

4.1 Introduction

The current computing landscape is one which is plagued by security issues and threats. Moreover, security incidents and breaches are on the rise. According to statistics complied by the Computer Emergency Response Team[1] (CERT) and Symantec[2], the number of new security threats that are reported is increasing every year [146]. Attacks are also becoming more sophisticated and diverse. For example, security vendor McAfee[3] reported that, as of January 2008, there were a total of 457 different types of mobile malware [84]. Mobile malware was virtually unheard of before 2004. As a consequence of the wide variety of security issues, companies as well as individuals are suffering financial and productivity losses.

Over the years, organisations have gained competence in protecting their networks (e.g. using firewalls and intrusion prevention and detection systems); today's attackers are instead turning their attention to what they perceive as the weakest point in these systems, i.e. client machines. That is, attacks are increasingly being directed at client machines and devices, since end users are usually not particularly skilled at defending against security threats and vulnerabilities.

Challener et al. [35] identify six of the most significant security threats that are affecting end users and their devices today:

1. Vulnerable programs: such as programs allowing buffer overflows and parsing errors;
2. Malicious programs: including viruses, spyware and adware;
3. Misconfigured programs: i.e. programs that may have adequate security features, but which are not turned on or configured properly;

[1]http://www.cert.org/
[2]http://www.symantec.com/
[3]http://www.mcafee.com/

4. Social engineering: phishing and pharming attacks;
5. Physical theft of user devices: e.g. laptops, PDAs, mobile phones, etc.;
6. Electronic eavesdropping: e.g. capturing email.

Challener et al. [35] attribute the proliferation of such security threats to the fact that it is extremely difficult to develop software that is completely secure. Hence software bugs simply remain dormant until they are discovered and exploited. Three main reasons have been identified for this phenomenon. Firstly, modern software systems are extremely complex, usually consisting of millions of lines of source code. It is therefore highly likely that there will be bugs in the code. Secondly, it is non-trivial to replace older systems which do not have the necessary security features with those which can handle modern threats. Hence new systems are integrated with these older systems. Compatibility issues often arise as a result of integration. Finally, most existing defence mechanisms are software-based. Challener et al. argue that a purely software-based approach may not be adequate, and advocate a hardware-based approach. Schell and Thompson [136] express a similar view, and were amongst the first to recognise that the lack of (hardware-based) platform defences could impede the growth and adoption of potential applications.

This was exactly the mission of the Trusted Computing Platform Alliance[4] (TCPA) and its successor the Trusted Computing Group[5] (TCG), i.e. enhancing the security of computing platforms using a hardware-based approach.

4.1.1 The Trusted Computing Group

Trusted computing platform specifications were first published by the TCPA (which included IBM, Intel, HP and Microsoft) in January 1999. The alliance expanded rapidly, and by 2002 it had more than 150 members. In April 2003, the TCPA was succeeded by the Trusted Computing Group (TCG), who adopted the work and the specifications of the TCPA. The TCG is a not-for-profit industry standards organisation, formed to develop, define, and promote open standards for hardware-enabled

[4]http://www.trustedcomputing.org/
[5]http://www.trustedcomputinggroup.org/

trusted computing and security technologies, across multiple platforms, peripherals, and devices. The TCG specifications are intended to enable the construction of more secure computing environments without compromising functional integrity, privacy, or individual rights. One of the primary goals of the TCG is to help users protect their information assets (data, passwords, keys, etc.) from compromise arising from external software attacks or physical theft.

4.1.2 What is a Trusted Platform?

In their seminal paper [96], McKnight and Chervany established that *trust* is a very complex notion which means different things to different people, in different contexts, and in different environments. We first look at the definition of "trust" or "trusted" used by the TCG. According to the TCG:

"A trusted system or component is one that behaves in the expected manner for a particular or intended purpose."

Using the notions of behavioural trust and social trust, Balacheff et al. [11] distinguish between whether a platform *can* be trusted and whether a platform *should* be trusted (we refer the interested reader to Chapter 1 of [11] for further details).

Trusted Computing, as developed by the Trusted Computing Group, is a technology designed to enhance the security of computing platforms. It involves incorporating trusted hardware functionality, or so called "roots of trust", into platforms. Users can thereby gain greater assurance that a platform is behaving in the expected manner [11, 100, 151]. Trusted hardware, in the form of a hardware component called the Trusted Platform Module (TPM), is built into a host platform. The TPM provides the platform with a foundation of trust, as well as the basis on which a suite of Trusted Computing security functionality is built. The TPM and its host are collectively referred to as a *Trusted Platform*. A trusted platform should support the following functionalities (described below): Attestation, Integrity Measurement, Storage and Reporting, Protected Capabilities, and Protected Message Exchange.

4.2 Trusted Platform Architecture

In this section we introduce the functional components of a Trusted Platform, focussing in particular on the TPM. This provides the basis for understanding the functionality provided by a TPM, discussed in the following section.

4.2.1 Roots of Trust

Trust in any computing platform must be based on a sound foundation, or ‘root of trust’. In other words, any platform which might be called a ‘trusted platform’ must contain some trustworthy functionality that acts as a starting point for building trust in its operation. As described below, a TCG-compliant trusted platform has three roots of trust, namely the root of trust for measurement (RTM), the root of trust for storage (RTS), and the root of trust for reporting (RTR).

4.2.1.1 The RTM

The RTM is a computing engine that is trusted to provide accurate integrity measurements [151], i.e. measurements of software that has been executed on the platform. On a PC, the RTM is actually the platform itself (i.e. the normal computing engine in the platform, and not the TPM). Since, the measurement process must commence as soon as a platform is powered on (i.e. the RTM must be the first program to execute on the platform), it is logical to place the RTM instruction set in the BIOS, or, more specifically, in the BIOS Boot Block (BBB). This set of instructions is often referred to as the *Core Root of Trust for Measurement* (CRTM).

It is important to note that the CRTM resides in the BBB, and it may be infeasible to make the BIOS resistant to physical attacks. However, the TCG specifications do not require the CRTM to be tamper-resistant, but only tamper-evident [11].

4.2.1.2 The RTS

At an abstract level, and as noted by Gallery in Chapter 3 of [100], the TPM is basically the realisation of the RTS and the RTR. The RTS has the following three main functions.

- It must accurately (i) store the integrity measurements made by the RTM in a log file, (ii) condense the integrity measurements made by the RTM in a way that preserves the values and order of measurements made, and (iii) store the results in registers residing in the TPM (details of exactly how this is done are given in Section 4.3.1).
- It must protect a variety of types of keys and data. Certain of these keys are used to support the Protected Message Exchange Mechanism, described in Section 4.3.2.
- It must support the migration (allowing the transfer of migratable objects from one TPM to another), and maintenance (allowing the transfer of non-migratable objects from one TPM to another) capabilities of a Trusted Platform.

4.2.1.3 The RTR

The RTR is primarily responsible for reporting the current integrity measurements for a trusted platform (as stored by the RTS) to external ‘challengers’. The RTR is used to support the Integrity Measurement, Reporting, and Storage functionality (described in Section 4.3.1).

4.2.2 TPM Functional Components

As discussed in the previous section, the TPM incorporates the RTS and the RTR. In this section we introduce the components that make up the TPM, and briefly explain the function of each component. A TPM which conforms to the TCG TPM specifications should contain all the components shown in Figure 4.1. This discussion will also help to explain the functionality that is provided by a TPM.

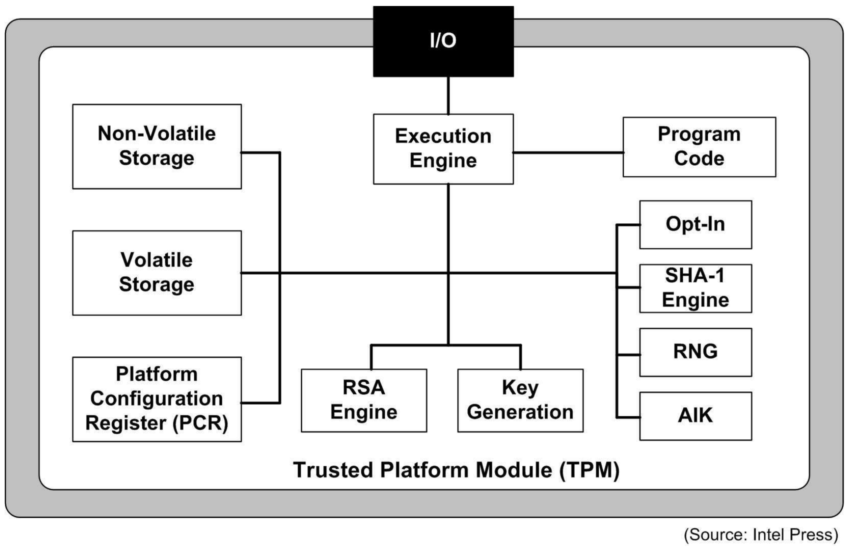


Figure 4.1: The TPM Design

1. **Input/Output (I/O):** The I/O component provides an interface between a TPM and the other components of a host platform. It manages all information flows (e.g. the TPM commands) over the communications bus, and is responsible for the following [100, 151]:

   (a) encoding/decoding of communications over internal and external buses,

   (b) routing messages to the appropriate TPM component,

   (c) enforcing access control policies within a TPM.

2. **Execution Engine:** Upon receipt of a TPM command, the execution engine parses the command, locates the program associated with the command (in the program code component), and then executes it. It ensures that (i) operations are segregated, and (ii) shielded locations are protected [100].

3. **Program Code:** Program code contains firmware for measuring platform devices [151]. It is also responsible for the validation of (i) the entire command bit stream, (ii) the parameters of the code, and (iii) the command authorisation information (further information can be found in Chapter 8 of Grawrock [66]).

4. **Non-Volatile Storage:** Certain data and information must be available to a TPM upon power up, and not be affected by power cycles. These must reside in the non-volatile storage, which is used to store persistent data such as the Endorsement Key (EK) and the Storage Root Key (SRK) (the EK and SRK are described in the next section).

5. **Volatile Storage:** The volatile storage area is used by the TPM to store the keys that are in use by the TPM (but not the EK or the SRK).

6. **Platform Configuration Registers (PCRs):** PCRs are registers in the TPM used primarily to store integrity measurements. Each PCR is 20 bytes long, and the TCG TPM specification requires that a TPM possesses a minimum of sixteen PCRs. Registers 0-7 are reserved for TPM use, while registers 8-15 are available for use by the operating system and applications [151]. In Section 4.3.1 we describe how PCRs are used to support the Integrity Measurement, Storage and Reporting mechanism.

7. **RSA Engine:** The RSA Engine is used for:

   (a) signing data using private signing keys,

   (b) encryption/decryption of data using storage keys,

   (c) decryption using the Endorsement Key.

   The implementation of the RSA algorithm must comply with the PKCS #1 v2.1 standards [131]. In most cases it is recommended (and in some instances mandated) that 2048-bit RSA keys are used for the above operations. However, for backward compatibility and other reasons, other key lengths (e.g. 512, 768 and 1024 bits) are also supported by the TPM.

8. **RSA Key Generation:** The RSA Key Generation engine is capable of generating signing keys and storage keys of up to 2048 bits in length for use by the RSA Engine.

9. **Opt-In:** A TPM is an opt-in device, meaning that a platform owner must take specific steps in order to activate it. The opt-in component must incorporate a mechanism to detect physical presence (of a human, usually the owner of the platform) [66]. This may be in the form of a simple button or switch.

10. **SHA-1 Engine:** A TPM must incorporate a SHA-1 hash engine where, as described in Section 2.4.2, SHA-1 takes as input an arbitrary length data string and gives as output a 20-byte hash code.

11. **Random Number Generator (RNG):** A TPM must incorporate a true random bit generator, which is used to seed a pseudorandom number generator. The RNG is also used for key generation, nonce creation, and to strengthen pass phrase entropy [151]. Further details of the TPM RNG can be found in Chapter 3 of [100].

12. **Attestation Identity Key (AIK):** AIKs must be persistent. Since a TPM may generate an arbitrary number of AIKs (as described below), it may not be feasible to store all of them inside the TPM. It is therefore recommended that AIKs be stored as Blobs (i.e. bit-strings in the format output by the TPM) in persistent external storage (i.e. outside the TPM). AIKs are described in greater detail in Section 4.2.3.

4.2.3 TPM Keys and Identities

In this section we describe the various types of key used by a TPM.

4.2.3.1 Endorsement Key (EK)

Every TPM has a unique 2048-bit RSA key pair called the *Endorsement Key* (EK). The EK is likely to be generated by the TPM manufacturer before the platform is shipped to end users. The EK private key, together with a certificate for the corresponding public key, can be used to prove that a genuine TPM is contained in a platform. As discussed in the previous section, the EK private key resides in the non-volatile storage of a TPM, and is never revealed outside of the TPM. Although the EK public key may be exposed outside of the TPM, it is only used to interact with external entities in special circumstances, because a TPM can be uniquely identified by its EK (as a TPM only has one such key pair).

4.2.3.2 Attestation Identity Keys (AIKs)

A TPM can generate an arbitrary number of 2048-bit RSA *Attestation Identity Key* (AIK) key pairs, which are used for interacting with other entities. AIKs function as identities or pseudonyms for a trusted platform, and platform privacy can be achieved by using a different AIK to interact with different entities. AIKs are used to:

1. sign certain data, to prove that they are stored in, or originate from, a TPM;
2. certify other keys originating from the TPM.

Since AIKs (instead of an EK) are used to interact with external entities, a TPM must therefore convince an external entity that a particular AIK has originated from a genuine TPM. This is addressed by assocating an AIK with an EK in such a way that an external entity is unable to learn the exact EK. The TCG has specified two approaches to support this association, and we examine them in greater detail in Section 4.4.

4.2.3.3 Storage Root Key (SRK)

The Storage Root Key (SRK) is a 2048-bit RSA key pair which is generated by a TPM. The primary function of an SRK is to protect (i.e. by encrypting) other keys that are stored externally to the TPM. For example, if an AIK private key is to be stored outside of the TPM, it must be encrypted using the SRK before it leaves the TPM. The SRK resides in the non-volatile storage in a TPM, and functions as a master key for all secret information stored externally to a TPM. When a TPM owner is established, an SRK pass phrase is also established, so that no one else (apart from the owner) has access to the SRK. This pass phrase is encrypted using the EK.

4.2.3.4 Key Attributes

Every key associated with a TPM can be defined as either *Migratable* or *Non-Migratable*. A non-migratable key (usually the private portion of it) cannot leave the TPM in cleartext form. Migration of a non-migratable key would have the serious disadvantage of allowing one platform to masquerade as another [151]. For this reason, AIKs must always be non-migratable.

On the other hand, there may be cases where a user wishes his/her data or keys to be accessible on other platforms. The TPM supports this through the use of migratable keys.

4.2.3.5 Other Key Types

The following additional key types are defined [11]:

1. **Storage keys** are used to wrap (i.e. protect using encryption) externally stored keys and data, and must be RSA keys of length at least 2048 bits.
2. **Signature keys** are used to sign application data and messages. Such keys can be either migratable or non-migratable. These restrictions can be imposed either by the TPM or its owner.
3. **Bind keys** are keys that are intended to be used to encrypt small strings of data (e.g. a secret key for a symmetric algorithm).
4. **Legacy keys** are keys created outside the TPM. Such keys are supported for backwards compatibility, e.g. to enable data from legacy applications to be decrypted.

4.2.4 TPM Credentials

There are four defined types of credential [151].

1. **Endorsement Credential:** An Endorsement Credential is issued to a TPM by an entity known as the Trusted Platform Module Entity (TPME). The TPME is the entity (likely to be the TPM manufacturer) that generates and embeds the EK pair into the TPM. This credential is a digitally signed statement containing the following information: (i) a statement that it is an endorsement credential, (ii) the public EK, (iii) the TPM type and its properties, and (iv) the name of the TPME. It is important to note that an endorsement credential uniquely identifies a particular platform for the reasons discussed in the previous section.

2. **Conformance Credential:** A Conformance Credential can be issued by any entity which has the standing to make credible statements about TPMs and the platforms containing them. The credential vouches for the fact that a particular TPM conforms to the TCG specifications, and that the TPM is properly incorporated into the platform. A conformance credential typically contains the following information: (i) evaluator name, (ii) platform manufacturer name, (iii) platform model number and version, (iv) TPM manufacturer name, and (v) TPM model number and version. Note that a conformance credential does not contain information that could be used to uniquely identify a platform. Also a platform may have multiple conformance credentials attesting to the various components used to build a platform.

3. **Platform Credential:** A Platform Credential is issued by the platform manufacturer; apart from describing the properties of a platform, it asserts that a particular platform contains a TPM of the type described in the Endorsement Credential. It also makes references to the Endorsement Credential and the Conformance Credential(s), and, as a result, a Platform Credential uniquely identifies a platform. A Platform Credential contains: (i) the name of the platform manufacturer, (ii) the platform model and version, (iii) a reference to the endorsement credential, and (iv) a reference to the conformance credential(s).

4. **Identity or AIK Credential:** AIK credentials are issued by Privacy CAs. Such a credential is used to certify that an AIK public key belongs to a TPM with particular properties, without identifying that platform. An AIK credential contains: (i) the AIK public key, (ii) the TPM model number, (iii) the name of the TPM manufacturer, (iv) the platform type, (v) the name of the platform manufacturer, and (vi) a reference to the conformance credential.

For further information on TPM Credentials, see Chapter 3 of [100] and the TCG Architecture Overview Specification [151].

4.3 Trusted Computing Functionality

In this section we describe four key features of trusted computing, namely *Integrity Measurement, Storage and Reporting* (IMSR), *Protected Message Exchange* (sometimes also referred to as *Protected Storage*), *Authenticated Boot*, and *Virtualisation.*

4.3.1 Integrity Measurement, Storage and Reporting (IMSR)

Integrity Measurement, Storage and Reporting (IMSR) is a key feature of Trusted Computing that builds on the three Roots of Trust in a trusted platform (as described in Section 4.2.1). Together, these roots of trust allow a verifier to learn the operational state of a platform, and hence obtain evidence of a platform's behaviour. This functionality is extremely important, as a platform may potentially enter a wide range of operational states, including those that are insecure and undesirable.

4.3.1.1 Integrity Measurement

Integrity measurement involves the RTM measuring a platform's operational state and characteristics. The measured values, known as integrity metrics, convey information about the platform's current state (and hence trustworthiness).

4.3.1.2 Integrity Storage

Details of exactly what measurements have been performed are stored in a file called the *Stored Measurement Log* (SML). Using the RTS, a digest (i.e. a cryptographic hash computed using SHA-1) of the integrity metrics is saved in one of the TPM's PCRs. The SML contains sequences of measured events, and each sequence shares a common measurement digest. Since an SML may become fairly large, it does not reside in the TPM. Integrity protection for the SML is not necessary, since it functions as a means to interpret the integrity measurements in the PCRs, and any modifications to the SML will cause subsequent PCR verifications to fail.

There are only a limited number of PCRs in the TPM. Thus, in order to ensure that previous and related measured values are not ignored/discarded, and the order of operations is preserved, new measurements are appended to a previous measurement digest, re-hashed, and then put back into the relevant PCR. This technique is known as *extending* the digest, and operates as follows:

$$PCR_i[n] \leftarrow \textit{SHA-1}(PCR_{i-1}[n] \,||\, \text{New integrity metric}),$$

where $PCR_i[n]$ denotes the content of the nth PCR after i extension operations, and $||$ denotes the concatenation of bit strings.

4.3.1.3 Integrity Reporting

The final phase of the IMSR process is Integrity Reporting. The RTR has two main responsibilities during Integrity Reporting:

1. to retrieve and provide a challenger with the requested integrity metrics (i.e. the relevant part of the SML and the corresponding PCR values); and
2. to *attest to* (prove) the authenticity of the integrity metrics to a challenger by signing the PCR values using one of the TPM's AIK private keys.

To verify the integrity measurements, the verifier computes the measurement digest (using the relevant portion of the SML), compares it with the corresponding PCR values, and checks the signature on the PCR values. The process of integrity reporting is also often referred to as *Platform Attestation.*

4.3.2 Protected Message Exchange and Storage

The TCG specification include three types of *Protected Message Exchange* mechanisms, namely *Binding*, *Signing*, *Sealing*, which we now describe. We also describe how the TPM supports the Protected Storage mechanism.

4.3.2.1 Binding

Binding uses asymmetric encryption, as introduced in Chapter 2. If a private decryption key is labelled as non-migratable, then only the TPM that owns it will ever have access to this key. Hence, if a message is encrypted using a public key corresponding to a non-migratable private key, then the message is deemed to be ‘bound’ to the TPM that owns the key pair, since no other platform can decrypt it.

On the other hand, if a message is encrypted with a public key whose corresponding private key is migratable, then the encrypted message is not bound to a platform, since it might be possible to decrypt it on other platforms.

4.3.2.2 Signing

Keys that are labelled as signing keys (discussed in Section 4.2.3.5) can only be used for the purpose of signing data, i.e. they are not permitted to be used for encryption.

4.3.2.3 Sealing

Sealing, an enhanced form of binding, is one of the key features of Trusted Computing. A sealed message, apart from being bound to a platform, is also associated with a set of platform metrics specified by the message sender. The platform metrics serve as additional requirements that must be satisfied before the decryption of a sealed message can take place.

The sealing process operates as follows:

1. The sender symmetrically encrypts the message M to be sealed using a key K to produce $E_K(M)$.

2. The encrypted message $E_K(M)$ is sealed to a particular platform state by computing $E_{PK}(K||PCR)$, where PCR contains PCR values specified by the sender and PK is a public key whose corresponding private key SK is non-migratable.

3. The encrypted message $E_K(M)$ can only be unsealed on a TPM which has the decryption key SK corresponding to PK, and the TPM will only do so if the platform configuration matches the PCR values specified by the sender.

Sealing associates a message with a set of PCR values and a non-migratable asymmetric key. Sealing is a powerful feature of the TPM, as it gives assurance that sealed data can only be decrypted when the platform is in a particular state. Arbitrary data, including keys, can be sealed. If a private signature key is sealed, i.e. so that the process of signing using this key is linked to a particular set of PCR values, then the signing process is known as *Sealed-Signing*.

4.3.2.4 Protected Storage

The TPM can protect stored data of any form, including keys. The TPM achieves this not by performing bulk encryption of data or keys on the TPM itself, but through key management via the SRK (described in Section 4.2.3.3). The SRK resides in the TPM, and is used to protect (i.e. encrypt) child keys, including identity keys, bulk encryption keys (also referred to as storage keys), and signing keys. These encrypted 'key blobs' are stored outside the TPM. The function of each type of key is described in Section 4.2.3.5. A key hierarchy is thereby created, with the SRK as the root key. For further details of the TPM key hierarchy and protected storage in general, see Chapter 7 of [11].

4.3.3 Authenticated Boot and Secure Boot

Trusted computing functionality can be used to support two distinct types of security-enhanced platform boot process, known as *Authenticated boot* and *Secure boot*.

4.3.3.1 Authenticated Boot

In an authenticated boot process, the RTM, the measurement agents, and the TPM cooperate to measure and record the boot sequence in the PCRs and the measurement log (i.e. the SML). We briefly describe the sequence of events during a typical authenticated boot process (see also Figure 4.2):

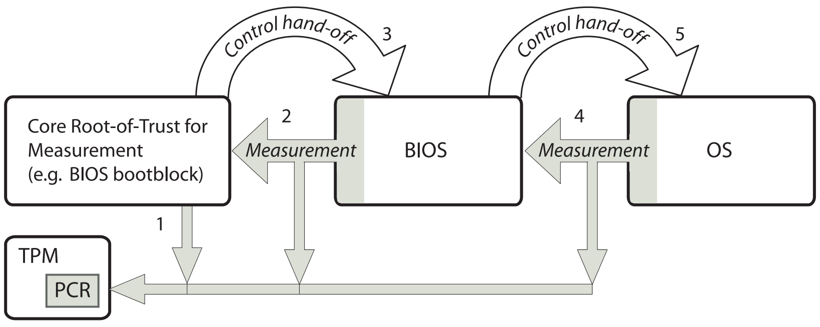


Figure 4.2: An Authenticated Boot Process

1. The CRTM, which resides in the BIOS Boot Block (BBB), measures itself and the rest of the BIOS, computes a cryptographic hash (using SHA-1) of the measurements, and stores the output integrity metric in the first PCR (i.e. PCR 0). The CRTM passes control to the measured entity, i.e., the BIOS.

2. The BIOS measures the next component, i.e. the Operating System, computes a cryptographic hash of the measurements, stores the output integrity metrics in the next PCR (i.e. PCR 1), and then passes control to the Operating System.

3. The Operating System measures the next component, e.g., the code of an application, computes a cryptographic hash of the measurements, stores the integrity metrics in the next PCR, and then passes control to the application.

4. This process of measuring the next entity, storing the measurement, and passing control to the measured entity continues indefinitely, enabling the platform at any time to attest to its current configuration.

The authenticated boot process is sometimes also known as the process for *establishing transitive trust* [66].

4.3.3.2 Secure Boot

Whilst the authenticated boot process enables the boot process to be measured, it does not control what is booted; secure boot takes this one step further by both measuring and controlling the boot process. That is, a secure boot process will be aborted if, at any stage, the actual boot process differs from the expected process. Secure boot involves the RTM, the measurement agents, the PCRs, and another type of internal register in the TPM known as a data integrity register (DIR). Before a trusted platform boots, the platform owner writes integrity metric values (the reference or expected values) to the DIRs. When the platform boots, the RTM measures the operational state of the platform, hashes the measurements, and stores the results in the PCRs. At each stage, the PCR values are compared to the values in the DIRs, and, if there is a discrepancy, the boot process is suspended.

If the secure boot process completes successfully, then the platform must have booted into the desired state.

4.3.4 Isolated Execution Environments

It may be necessary for a sensitive application (e.g. a banking application) to be protected from external interference (e.g. eavesdropping or even active manipulation) or from other processes that are running on the same platform. In order to achieve this, a number of logical compartments or execution environments can be created within a single platform. As discussed by Peinado, England and Chen [115], an isolated execution environment must possess the following properties:

- No interference: the program/software in the isolated execution environment must be isolated from external interference.
- No observation: the program's data and computation should not be observable by other entities.
- Trusted paths: a secure channel must exist between the program and its input (i.e. keyboard and mouse) and output devices (i.e. video).
- Secure communication: the program should be able to securely exchange data with other programs in a way that ensures the data's confidentiality and integrity.

These requirements can be supported through the use of virtualisation technology. Virtualisation provides a platform with the ability to run multiple 'virtual machines' on one physical platform, managed by a 'hypervisor' or Virtual Machine Monitor (VMM), as shown in Figure 4.3.

Virtual machines (VMs) are isolated from each other, and one VM cannot access memory space allocated to another VM — the VMM ensures this with assistance from the hardware. For example, Intel's Trusted Execution Technology (TXT)[6] [66] provides hardware support to help enable the creation of secure (and measurable) compartments in which virtual machines can operate.

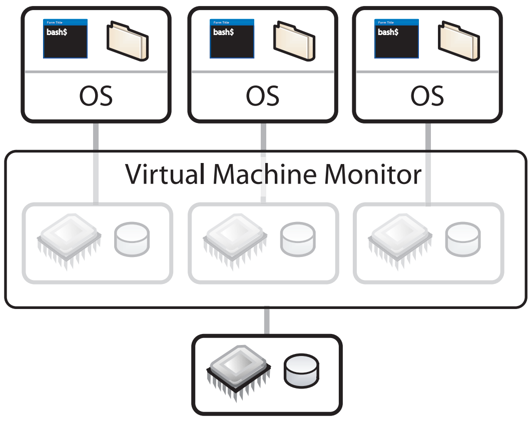


Figure 4.3: A Virtualisation Scenario

[6]formerly known as Intel LaGrande Technology

The use of virtualisation slightly alters the authenticated boot process, as the VMM will also need to be attested to. This may be advantageous, because it could make the use of attestation more manageable. A detailed description of virtualisation technologies is beyond the scope of this thesis; the interested reader is referred to [66, 115].

4.4 Trusted Computing and Privacy

Trusted computing can help to protect the privacy of platform users by hiding the identity of the platform. The TCG adheres closely to the guiding principles of the World Wide Web Consortium (W3C) Platform for Privacy Preference (P3P)[7] working group [121]. One of these principles addresses the confidentiality of users' personal information, i.e. that personally identifiable information (PII) should always be protected with appropriate security safeguards.

One of the goals of trusted computing is to allow an external entity which is interacting with a trusted platform to recognise that the platform is indeed a trusted platform (i.e. it contains a genuine TPM), but not to uniquely identify the platform. The Endorsement Key (along with the EK Credential described in Section 4.2.4) could, in theory at least, be used to prove that a platform contains a genuine TPM. However, since there is only one EK key pair per TPM, it could also be used to identify it. The EK is therefore considered as PII, and its use is hence restricted. To further protect the EK, a TPM owner may also associate authorisation data with the EK, so that the release of the EK public key or use of the EK private key becomes an authorised command.

As mentioned in Section 4.2.3.2, instead of using the EK, a trusted platform uses AIK pairs to interact with external entities, although an external entity must be able to verify that a particular AIK has originated from a genuine TPM. This is solved by associating an AIK with an EK is such a way that the external entity is unable to learn the exact EK. The TCG specifications support two ways of managing such associations, namely the use of a Privacy CA and the DAA protocol. We describe both approaches in greater detail below.

[7]See http://www.w3.org/TR/P3P/

4.4.1 Privacy CA

A Privacy Certification Authority, or Privacy CA, is a trusted computing specific trusted third party. Its primary role is to create an Identity Credential, or AIK Credential, which attests to the fact that a particular AIK public key originates from a platform containing a TPM of a particular type. A Privacy CA is able to ascertain whether a platform contains a genuine TPM by inspecting its platform, conformance and endorsement credentials. If the credentials are accepted by the Privacy CA, it issues the TPM with an AIK Credential. We now describe the steps involved in obtaining an AIK Credential from a Privacy CA [11].

1. The TPM owner instructs the TPM to generate a new identity (i.e. a new AIK). The TPM then creates an *identity-binding*, i.e. a signature that links together the following:
   (a) the newly generated AIK public key,
   (b) a name (for the AIK key) chosen by the TPM owner, and
   (c) the identifier of the Privacy CA chosen by the TPM owner to attest to the new identity.
2. The TPM assembles the data required by the Privacy CA in order to issue the AIK Credential, including the *identity-binding* in step 1, and the endorsement, platform and conformance credentials.
3. The platform owner encrypts the data assembled in step 2 using the Privacy CA's public key, and sends the encrypted data to the Privacy CA.
4. On receiving the above message, the Privacy CA decrypts it, and examines the credentials to determine whether they meet the requirements of the Privacy CA's issuing policy.
5. The Privacy CA now performs the following steps:
   (a) creates an AIK credential,
   (b) encrypts the AIK credential with a newly generated secret key K,
   (c) encrypts K using the EK public key specified in the Endorsement Credential, and

(d) sends the encrypted AIK credential and the encrypted key to the TPM.

6. On receiving the above encrypted messages, the TPM uses its EK private key to decrypt K. TPM then uses the key K to extract the AIK credential.

In the above process, the EK is divulged to the Privacy CA, and the Privacy CA therefore has full knowledge of EK-AIK associations. The privacy of platform users therefore rests on the trustworthiness of Privacy CAs, and this may not be desirable. The Direct Anonymous Attestation protocol, discussed next, has been proposed to avoid this dependency.

4.4.2 Direct Anonymous Attestation

Direct Anonymous Attestation (DAA) [27, 100] is a special type of group signature scheme that can be used to anonymously authenticate a TCG v1.2 compliant platform to a remote verifier. DAA has been adopted by the Trusted Computing Group in version 1.2 of the Trusted Computing Trusted Platform Module (TPM) Specifications [151]. The key features that DAA provides are the capability for a TPM (a prover) to anonymously convince a remote or external verifier that:

- it is in possession of a DAA Certificate obtained from a specific DAA Issuer, without having to reveal the DAA Certificate (or any unique identifiers, e.g. the EK) to a verifier, and hence it is indeed a genuine TPM (and that it will therefore behave in a trustworthy manner);
- a DAA Signature computed by a prover on a message m has been generated using a valid DAA Certificate issued by a specific DAA Issuer; colluding verifiers are unable to link two different DAA Signatures created by the same prover, and, in particular, verifiers are not given the DAA Certificate.

Therefore, if an AIK public key is signed using DAA, a verifer can be convinced that it is held by a TPM with the specified properties. The DAA scheme also provides a flexible way of achieving a number of different levels of 'linkability'. Subject to agreement between the prover and verifier, DAA Signatures can be either 'random-base' or 'name-base'. Two random-base signatures signed by the same prover (TPM) for the same verifier cannot be linked. However, name-base signatures are associated with the verifier's name; as a result, two name-base signatures signed by the same prover (TPM) for the same verifier can be linked.

These features help to protect the privacy of a TPM user. Another important privacy-preserving feature of DAA is that the powers of the supporting Trusted Third Party (the DAA Issuer) are minimised, as it cannot link the actions of users even if it colludes with a verifier.

DAA allows a prover to anonymously convince a remote verifier that it has obtained an anonymous attestation credential, or DAA Certificate (a Camenisch-Lysyanskaya (CL) signature [32]), from a specific DAA Issuer (Attester). The DAA Certificate can also be used to provide an implicit "link" between an EK and an AIK.

We first provide a high level description of the DAA protocol. We then describe the two sub-protocols: *DAA Join* and *DAA Sign*. After which we explain how the variable anonymity capability is achieved. We conclude our discussion by briefly reviewing some proposed variants of DAA; we also mentioned two possible shortcomings of the protocol.

4.4.2.1 High Level Overview

We first introduce the entities involved in the DAA protocol and the roles they play.

- The *Certification Authority (CA)* acts as a trusted third party. Its role is to certify the authenticity of the DAA Issuer's longer-term public key, CK_I. It does not directly participate in the DAA protocol.
- The *DAA Issuer* (or just the Issuer) is a third party that issues DAA Certificates (i.e. anonymous credentials) to provers. It must be trusted to perform its role in a trustworthy manner by the other protocol participants. A DAA Issuer must possess two types of public key: a longer-term public key CK_I, and a shorter-term public key PK_I. The reason for this multi-level key hierarchy is to provide an issuer with flexibility in choosing the frequency with which it changes its (shorter term) key pair, and also to simplify key update in the

event that its shorter term key pair is compromised. Further discussion of this feature can be found in Chapter 5 of [100].

- The *Prover* (or the User) generates DAA Signatures. In the context of Trusted Computing, the prover is the TPM.

- The *Verifier* verifies DAA Signatures computed by provers.

Note that, apart from the CA, all the entities listed above take direct part in the DAA protocol. Also, in normal circumstances, the numbers of CAs and DAA Issuers are likely to be very small by comparison with the number of provers.

The DAA Scheme consists of two sub-protocols (or phases), namely the *DAA Join Protocol* and the *DAA Sign Protocol*. In the Join Protocol (shown in Figure 4.4), a prover interacts with a DAA Issuer I in order to obtain an anonymous credential on a secret value f (also referred to as the DAA Secret), known only to the prover. This anonymous credential, known as a DAA Certificate, is jointly computed by the DAA Issuer and the prover as a function of a blinded value of f, the shorter-term public key of I, PK_I, and other parameters. The DAA Certificate is later used by a prover during the DAA Sign Phase to compute a DAA Signature. As part of the DAA Join protocol, the prover is authenticated to the DAA Issuer using its Endorsement Key. The Issuer authenticates itself to the prover using its shorter-term public key PK_I, which the prover verifies using a certificate signed by the Issuer with its longer-term public key CK_I, which is in turn certified by the CA.

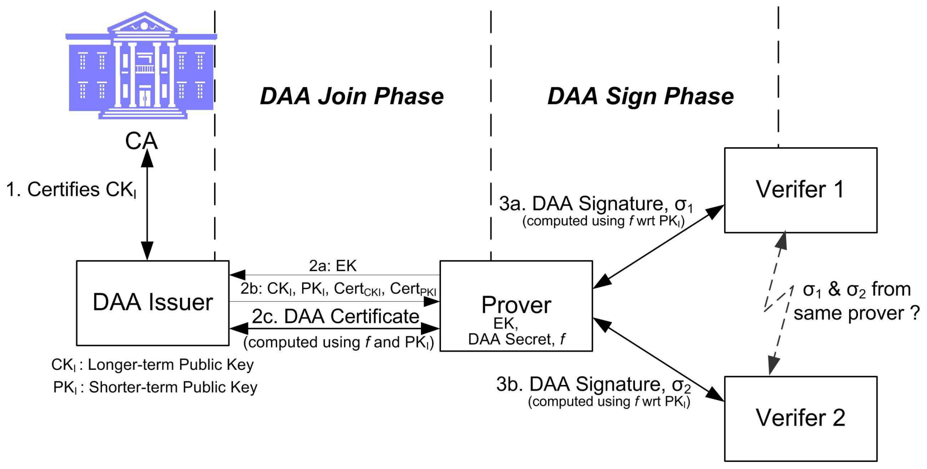

DAA Join Phase
DAA Sign Phase
CA
1. Certifies CK_I
DAA Issuer
CK_I : Longer-term Public Key
PK_I : Shorter-term Public Key
2a: EK
2b: CK_I, PK_I, Cert_CKI, Cert_PKI
2c. DAA Certificate
(computed using f and PK_I)
Prover
EK,
DAA Secret, f
3a. DAA Signature, σ1
(computed using f wrt PK_I)
3b. DAA Signature, σ2
(computed using f wrt PK_I)
Verifer 1
Verifer 2
σ1 & σ2 from same prover ?

Figure 4.4: The DAA Scheme

In the DAA Sign Phase, the prover DAA-signs a message m, using its DAA Secret f, the DAA Certificate, and other public parameters. The output of the DAA-signing process is known as the DAA Signature. This DAA Signature enables the prover to prove to a verifier (using a signature-based proof of knowledge) that (i) it is in possession of a DAA Certificate, and (ii) the DAA Signature on message m was computed using its DAA Secret f, the DAA Certificate in (i), and other public parameters. Verifying a DAA Signature requires knowledge of the DAA Issuer's public key PK_I (i.e. the public key of the DAA Issuer that was used to create the DAA Certificate). Hence, prior to running the DAA Sign protocol, a verifier must have obtained an authentic copy of PK_I.

The DAA Sign Protocol has the property that colluding verifiers are unable to link different DAA Signatures originating from the same prover (as shown in Figure 4.4). This property applies even if a DAA Issuer colludes with a verifier. If the DAA Issuer and the verifier is the same entity, then an attack is possible [140].

4.4.2.2 DAA Join Protocol

As described above, the Join protocol enables the TPM to obtain a DAA Certificate (also known as an anonymous attestation credential) from the DAA Issuer. The Join protocol is based on the CL signature scheme [32].

Let (n, S, Z, R) be the DAA Issuer public key, where n is an RSA modulus and S, Z, and R are integers modulo n. We assume that the platform (TPM) is already authenticated to the DAA Issuer via its Endorsement Credential (which contains the EK public key).

The platform (TPM) first generates a DAA secret value f and makes a commitment to it by computing $U = R^f S^{v'} \bmod n$, where v' is a value chosen randomly to "blind" f. The platform (TPM) also computes $N_I = \zeta_I^f \bmod \Gamma$, where ζ_I is derived from the DAA Issuer's name and Γ is a large prime. The platform then sends (U, N_I) to the DAA Issuer, and convinces the DAA Issuer that U and N_I are correctly formed (using a zero knowledge proof [61, 64]). If the DAA Issuer accepts the proof, it signs the hidden message U by computing $A = (\frac{Z}{US^{v''}})^{1/e} \bmod n$, where v'' is a random integer and e is a random prime. The DAA Issuer then sends the platform (i.e. the TPM) the triple (A, e, v''), and proves that A was computed correctly. The DAA Certificate is then $(A, e, v = v' + v'')$.

4.4.2.3 DAA Sign Protocol

As described above, the Sign protocol allows a platform to prove to a verifier that it is in possession of a DAA Certificate, and, at the same time, to sign a message. The platform signs a message m using its DAA Secret f, its DAA Certificate, and the public parameters of the system. The message m may be an AIK public key generated by the TPM, or an arbitrary message. The platform also computes $N_V = \zeta^f \bmod \Gamma$ as part of the signature computation (the selection of ζ is discussed in the next section). The output of the Sign protocol is known as the DAA Signature, σ.

The verifier verifies the DAA Signature σ, and, if the verification succeeds, is convinced that:

1. the platform has a DAA Certificate (A, e, v) issued by a specific DAA Issuer, and hence it incorporates a genuine TPM containing a legitimate EK; this is accomplished by a zero-knowledge proof of knowledge of a set of values f, A, e, and v such that $A^e R^f S^v \equiv Z \pmod{n}$;
2. a message m was signed by the TPM using its DAA secret f, where f is the same as the value in the DAA Certificate (used in step 1); if m includes an AIK public key, then it originates from a genuine TPM.

In summary, once a platform (TPM) has obtained a DAA Certificate (which only needs to be done once), it is able to subsequently DAA-Sign as many AIKs as it wishes, without involving the DAA Issuer.

4.4.2.4 Variable Anonymity

Anonymity and unlinkability are provided to a user through the use of two parameters: ζ, also referred to as the *Base*, and the AIK. The choice of the base directly affects the degree of anonymity afforded to a TPM user. If perfect anonymity is desired, then a different, random, base value should be used for every interaction with a verifier. Conversely, if the same base value is used for every interaction with a verifier, then the verifier can link these interactions. In addition, if the same base value is used to interact with different verifiers, then they are able to correlate the activities of a particular TPM. (A more detailed discussion of the effects of choices of base values is given in [150]).

As discussed in Section 4.2.3.2, a TPM is capable of generating multiple platform identities by generating different AIKs. This unlinkability will be preserved if a TPM uses different AIKs to interact with different verifiers (provided the base is also different).

4.4.2.5 Developments of and Issues with DAA

DAA [151] is based on a protocol proposed by Brickell, Camenisch and Chen [27] in 2004. A number of variants have since been proposed, three of which we now briefly introduce.

1. Revoking a DAA credential issued to a compromised TPM is problematic precisely because of the anonymity features provided by DAA. Brickell and Li [30] extended the DAA Scheme to include an enhanced revocation capability, which aims to overcome this limitation whilst still maintaining full unlinkability. One limitation of this scheme is the requirement for an extra trusted third party to act as an revocation manager.
2. Ge and Tate [58] recently proposed a more efficient variant of DAA which retains its security and privacy features. This scheme has sign and verify protocols that are more efficient than DAA's, and is thus more suited for deployment on resource-constrained platforms, such as mobile devices. It is unclear whether the scheme will be incorporated into future TCG specifications.
3. A new protocol with the same properties as DAA but using elliptic curve cryptography and bilinear maps, has been proposed by Brickell, Chen and Li [28, 29]. This scheme has the advantage of having significantly shorter key

and signature lengths than DAA, whilst giving a similar level of security and computational complexity.

Despite the advantages offered by the above schemes, they have yet to have an impact on the TCG standards, and so they are not yet relevant in practice.

The privacy features of DAA have a wide range of potential uses. For example, Balfe et al. [17] have proposed using DAA to provide stable identities in peer-to-peer networks, thereby preventing Sybil attacks [48]. In chapters 7 and 9 of this thesis we show how DAA can be used as the basis of security schemes supporting mobile ubiquitous computing.

Finally we note that two possible security issues with DAA have recently been described. Smyth, Ryan and Chen [140] point out that the unlinkability property may be compromised if the DAA Issuer and the Verifier are the same entity. They also propose a simple modification that prevents the attack. Rudolph [132] shows that a DAA Issuer is able to embed covert identifying information into its public keys, enabling user anonymity to be compromised. The feasibility of this latter attack is analysed in Chapter 5, and possible countermeasures are also proposed. It should be noted that neither of these attacks are attacks on the DAA protocol itself, but rather on the specific use of DAA in the TCG specifications.

4.5 Applications of Trusted Computing

Applications of trusted computing have been proposed in a wide variety of contexts and settings, either to enhance the security of existing systems or to provide new security features. In this section we briefly review a range of such applications.

4.5.1 Commercial Applications

One of the relatively few applications of trusted computing technology in existing commercial products is the BitLocker disk drive encryption feature in Microsoft's Windows Vista Operating System and Windows Server 2008. Bitlocker is a full volume encryption feature designed to: (i) protect against the compromise of data on machines that are stolen or lost; and (ii) to provide secure data deletion when BitLocker-protected computers are decommissioned [55]. Furthermore, Bitlocker ensures that stored data is accessible only if the computer's boot components are unaltered and the encrypted disk resides in the original computer. Bitlocker supports three modes of operation (for further details see, for example, Gallery and Mitchell [55]).

Hewlett-Packard[8] PCs come equipped with a suite of tools called *ProtectTools Embedded Security.* These tools can be used to provide authentication of a TPM-enabled PC to an enterprise network, and to support various file and folder encryption capabilities. Other PC manufacturers which support use of trusted computing technology include Dell[9], Sony[10] and IBM[11]. More generally, it seems that there is a dearth of commercial off the shelf applications of trusted computing technology. This contrasts with the wide range of published proposals for applications of the technology, a selection of which we now describe.

4.5.2 Client Applications

The use of trusted computing has been proposed to enhance the security of a range of client applications. We discuss some of these proposals below.

E-commerce. Balfe and Paterson [19] described how trusted computing can be used to enhance the security of Internet-based Card Not Present (CNP) transactions. Their proposals rely on using the TPM to bind a platform, and hence its owner, to a particular card. Balfe and Paterson [20] also proposed a system (called e-EMV) that emulates EMV smart cards on trusted computing enabled platforms. The e-EMV scheme supports both the demonstration of card ownership and cardholder authentication, as well as defending against threats posed by malware (in particular, transaction generators). Using a combination of virtualisation and attestation mechanisms, Stumpf, Eckert and Balfe [143], proposed an architecture and

[8]http://www.hp.com/
[9]http://www.dell.com/
[10]http://www.sony.com/
[11]http://www.ibm.com/

client application for securing e-commerce transactions. Their scheme is resistant to client compromise and man-in-the-middle attacks on SSL.

Gaming. Final Fantasy, proposed by Balfe and Mohammed [18], is a trusted computing based security framework designed to be incorporated into gaming consoles, with the objective of preventing players from cheating during network gaming. Final Fantasy also provides a secure platform for players to engage in online auctions of in-game items.

Preventing Phishing Attacks and Crimeware. Gajek et al. [52] and Alsaid and Mitchell [9] investigated how trusted computing can be used to prevent various forms of phishing attacks. Balfe et al. [14] also examine how trusted computing can be used to address the threats that are posed by various forms of crimeware (e.g. keystroke loggers, viruses, worms, rootkits, and trojan horses).

Privacy Enhancing Applications. Chen, Pearson and Vamvakas [37] describe how trusted computing could be used to enhance the security and privacy of a biometric authentication process. Trusted computing functionality can be used to give a user assurance that the biometric authentication system will not compromise biometric information, by attesting to the trustworthiness (i.e. state) of the system to the user. Gajparia and Mitchell [53], proposed a scheme to preserve user privacy by allowing a user to determine the state of a destination platform before he/she decides whether to send it private data.

Client-Server Applications. Single Sign-On (SSO) systems provide a user with the ability to log on to a variety of different applications without the user having to maintain separate authentication credentials for each application [112]. Pashalidis and Mitchell [112] proposed a way of using trusted computing to enhance existing SSO systems. Sevinç, Strasser and Basin [138] describe how trusted computing could be used to secure the access, distribution and storage of confidential documents in a networked enterprise environment for mobile workers.

4.5.3 Distributed Computing Environments

In this section we review possible applications of trusted computing in three different distributed computing environments, namely ad hoc and sensor networks, grid computing, and peer-to-peer (P2P) networks.

Ad Hoc and Sensor Networks. Jarrett and Ward [78] proposed using trusted computing to prevent selfish or malicious nodes from participating in network activities (e.g. routing) by incorporating remote attestation mechanisms into the Ad Hoc On Demand Distance Vector (AODV) routing protocol [116]. Yan and Cofta [160] explored the possibility of using trusted computing to sustain the trust relationships established between entities, and hence build a trusted community over a period of time. This model is especially suited for an ad hoc network because a node would have assurance that the other nodes are carrying out their assigned tasks in a trustworthy manner. In sensor networks, nodes are often organised in clusters, and the nodes acting as cluster heads can be a target for attacks. Using trusted computing attestation mechanisms, Krauß, Stumpf and Eckert [82] describe two techniques whereby cluster nodes can check the integrity of a cluster head and thereby detect whether it has been compromised by an adversary. A working prototype of a secure mobile tele-therapeutic system for deployment in ad hoc environments was developed and implemented by Grossmann et al. [67]. Atmel TPMs were incorporated into mobile devices, which were built into mobile infusion pumps (for pain therapy) to allow secure remote control. The TPMs were also used to secure communications, e.g. between a patient's medical device and a hospital's webserver. Grossman et al. claim that the use of TPMs did not impose significant computational overheads on the system.

Grid Computing. Cooper and Martin [40], and Löhr et al. [88] both propose running grid services and grid user jobs in Virtual Machine (VM) compartments, and having the state of the virtualisation layer attested to the users. Users can thereby gain assurance regarding the security properties of the services and the execution environment. Löhr et al. [88] also propose an offline attestation scheme whereby grid resource providers distribute attestation tokens. An attestation token contains the provider's public key and the platform states to which the corresponding private key have been sealed. By examining the configuration information in attestation tokens, grid users can select a provider with which to interact. When a user decides to interact with a resource provider, the provider will only be able to unseal the private key if it is in the state specified in the attestation token. Sharing the philosophy that trust should flow in both directions between grid user and grid resource, Yau and Tomlinson [161] describe how trusted computing can be used to establish trust between a grid user and a grid resource in a variety of environments.

A concrete framework for securing grid workflows was proposed by Yau et al. [162]. This framework supports the following security services: (i) trusted resource provider selection; (ii) confidentiality and integrity of job information; and (iii) data auditing for process provenance. In a grid, it is often necessary to share resources and content amongst virtual organisation (VO) members. If the content is encrypted, then it may not be desirable to share the content decryption key amongst multiple members, given that some of them may not be trustworthy. This problem is addressed in the Daonity system, developed by Mao et al. [36, 90], in which a relocatable key is established to allow controlled group sharing of encrypted content. This is achieved using the credential migration capability of TPMs, allowing a group key to be securely transferred to only those group members that can attest to their platform states, and hence prove that they are trustworthy.

For a more detailed discussion of the application of trusted computing to grid computing, see Martin and Yau [91].

Peer-to-Peer (P2P) Networks. As discussed in section 4.4.2, Balfe, Lakhani and Paterson [16, 17] describe the possible use of trusted computing, in particular DAA, to provide stable addresses and pseudonymous authentication between nodes in a peer-to-peer environement, hence avoiding Sybil attacks. Kinateder and Pearson [80] presented a trusted computing based distributed reputation system, with enhanced privacy and security features. This system uses trusted computing to allow a peer node to derive confidence in the recommendations provided by other peer nodes. Using trusted computing functionality, Sandhu and Zhang [135] proposed a peer-to-peer based access control architecture, which uses the trustworthiness of user platforms and applications as part of its access control decision-making process. In this architecture, access to an object can be restricted to platforms with applications in specified valid states.

4.5.4 Other Applications

Abbadi [5, 6, 7] has proposed several schemes which use trusted computing for Digital Rights Management (DRM), i.e. protecting proprietary digital content against unauthorised access. Gallery [54] has also investigated how trusted computing can be used to help meet the DRM requirements for a mobile environment. Using trusted computing functionality, Gallery and Tomlinson [56], proposed two protocols to enable the secure delivery of conditional access applications to mobile receivers.

The use of Radio Frequency Identification (RFID) technology poses a number of privacy issues [57]; for example, RFID tagged items and products can easily be tracked. As a result, individuals in possession of such items could be tracked without their knowledge. Molnar, Soppera and Wagner [102] introduce a trusted computing based RFID architecture that addresses some of these privacy concerns. Remote attestation mechanisms are used in RFID readers to prove that they are running specific trustworthy software. Individuals interacting with the readers can thereby verify both that the readers comply with privacy regulations and also that they have not been compromised.

The use of trusted computing to provide privacy and security for mobile agents has been proposed by a number of authors [13, 42, 113, 114, 123]. Dietrich et al. [46] described a generic approach for establishing trust relationships between remote platforms using trusted computing. Finally, Li et al. [86, 87] have also provided a glimpse of how trusted computing can be used in pervasive computing environments to enforce trust, and hence protect the services and data of a critical information infrastructure.

4.6 Summary

In this chapter we introduced trusted computing technology. We introduced the concept of a trusted platform and described some of the capabilities of such a platform. We also looked at how end user privacy can be supported using Trusted Computing. We then reviewed some proposed applications of Trusted Computing.

Trusted computing technology is still evolving, and open issues and challenges remain. One such issue, highlighted by McCune et al. [95], is the turtle problem, i.e. the problem of establishing the first point of trust for user-based attestation in a network environment. Another area of considerable recent interest involves possible means of generalising the attestation mechanism. Such work includes Shi et al.'s [139] fine grained code attestation, and property-based attestation [70, 133]. Balfe et al. [15] highlighted some of the challenges that need to be overcome in order for the potential of Trusted Computing to be fully realised.

Indeed, trusted computing could become a ubiquitous security infrastructure, used routinely in the provision of almost every security service; indeed, this would appear to be part of the vision of the TCG (see, for example, [124]). It therefore would seem prudent to further explore ways in which such technology can be exploited to solve some of the most difficult problems facing users and designers of ubiquitous computing systems, not least in addressing such issues as user privacy and trust establishment. In this thesis, we examine a number of ways in which trusted computing could be used to secure mobile ubiquitous services.

Chapter 5

A Possible Privacy Flaw in DAA

Contents

A possible privacy flaw in the TCG implementation of the Direct Anonymous Attestation (DAA) protocol has recently been discovered by Rudolph. This flaw allows a DAA Issuer to covertly include identifying information within DAA Certificates, enabling a DAA Issuer that colludes with one or more verifiers to link and uniquely identify users, compromising user privacy and thereby invalidating the key feature provided by DAA. In this chapter we argue that, in typical usage scenarios, the weakness identified by Rudolph is not likely to lead to a feasible attack; specifically we argue that the attack is only likely to be feasible if honest DAA signers and verifiers never check the behaviour of issuers. We also suggest possible ways of avoiding the threat posed by Rudolph's observation.

5.1 Introduction

As discussed in Chapter 4, Direct Anonymous Attestation (DAA) is a special type of group signature scheme that can be used to anonymously authenticate a principal, also referred to as a prover, to a remote verifier. DAA was adopted by the Trusted Computing Group in version 1.2 of the Trusted Computing Trusted Platform Module (TPM) Specifications [151]. The key features provided by DAA are the capability for a prover (a group member) to anonymously convince a remote verifier that:

- it is in possession of a DAA Certificate obtained from a specific DAA Issuer, without having to reveal the DAA Certificate (as would be necessary for a signature-based proof of knowledge);
- a DAA Signature on a message m has been generated using a valid DAA Certificate issued by a specific DAA Issuer; colluding verifiers are unable to link two different DAA Signatures created by the same prover, and, in particular, verifiers are not given the DAA Certificate.

The DAA scheme also provides a flexible way of achieving a number of different levels of 'linkability'. Subject to agreement between the prover and verifier, DAA Signatures can be either 'random-base' or 'name-base'. Two random-base signatures signed by the same prover (TPM) for the same verifier cannot be linked. However, name-base signatures are associated with the verifier's name; as a result, two name-base signatures signed by the same prover (TPM) for the same verifier can be linked.

These features help to protect the privacy of a prover. Another important feature of DAA (distinguishing it from other types of group or ring signature schemes) is that the powers of the supporting Trusted Third Party (i.e. the DAA Issuer in its role as the group manager) are minimised, as it cannot link the actions of users (i.e. provers) and hence compromise user privacy, even if it colludes with a verifier. This unlinkability property is the key feature of DAA.

However, an attack was recently discovered by Rudolph [132] which potentially compromises the unlinkability property of DAA. The attack could allow a DAA Issuer to embed covert identifying information into DAA Certificates (of provers) and to subsequently link the transactions of the users/provers to whom the DAA Certificates belong [132]. As a result, DAA Signatures originating from the same users would become linkable, and user devices could thereby be uniquely identified. In this chapter, we argue that Rudolph's attack may be infeasible in practice, and we discuss why an attempt to launch such an attack could easily be discovered in an environment where there is at least one honest verifier. We also propose approaches which could prevent the attack from taking place.

The remainder of this chapter is organised as follows. In Section 5.2 we briefly outline the privacy attack. In Section 5.3 we explain why the attack is likely to be unrealistic in many practical scenarios, and, in Section 5.4, we discuss possible modifications to the use of DAA in the TCG specifications that can prevent the attack. Finally, conclusions are drawn in Section 5.5.

5.2 A Privacy Attack on DAA

In this section, we briefly describe how and under what assumptions the Rudolph attack works.

The privacy breaching attack on DAA proposed by Rudolph [132] operates under an assumption about the use of DAA, which we first describe. Specifically, it is assumed that the DAA Issuer's longer-term public key CK_I, as well as the (shorter-term) public key PK_I (along with its certificate chain) used to compute the DAA Certificate, are communicated to the verifier via the prover during the DAA Sign Phase (as shown in Figure 5.1). Whether or not this is a realistic assumption is not clear; other possibilities include use of a publicly accessible certificate repository. In any event, as we describe below, the verifier will need to know which of the DAA Issuer's shorter-term keys has been used to create the DAA Certificate on which the DAA Signature is based.

The attack works as follows. As shown in Figure 4.4, during the DAA Join Phase the DAA Certificate is computed using the DAA Issuer's public key PK_I and other parameters. The key PK_I is a shorter-term public key which is certified using the longer-term public key CK_I, which in turn is certified by a trusted CA (as shown

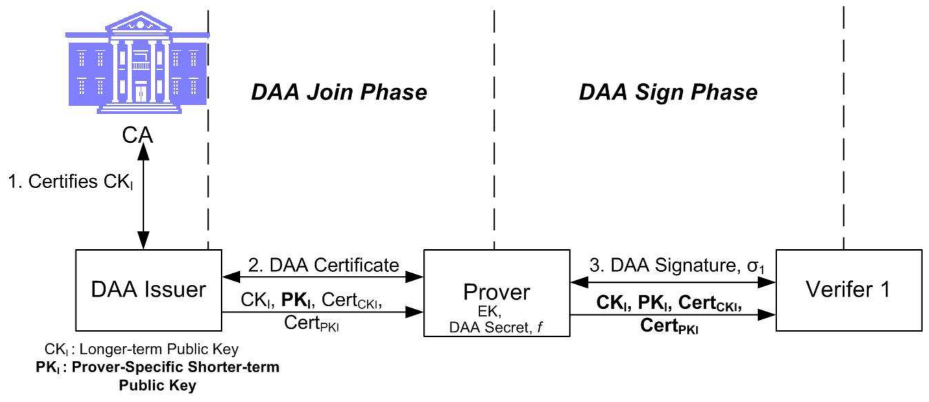

CA
1. Certifies CK_I
DAA Join Phase
DAA Sign Phase
DAA Issuer
2. DAA Certificate
CK_I, PK_I, Cert_CKI, Cert_PKI
Prover
EK, DAA Secret, f
3. DAA Signature, σ1
CK_I, PK_I, Cert_CKI, Cert_PKI
Verifer 1
CK_I : Longer-term Public Key
PK_I : Prover-Specific Shorter-term Public Key

Figure 5.1: The Rudolph Attack

in Figure 4.4). This key hierarchy is an intentional design feature of DAA, chosen to make the TPM and Issuer key life cycles independent. This is because the TPM computes its DAA private key as a digest of a secret seed and the Issuer's longer-term public key. This ensures that the TPM uses the Issuer key that matches its DAA private key. If the Issuer had only a single key, then, when the Issuer changed its key, every TPM would also have to update its key. To avoid this problem, the Issuer is given the two types of key described above.

Unfortunately, this flexibility can potentially be exploited by a curious DAA Issuer to compromise the privacy properties of DAA. As observed by Rudolph, a DAA Issuer could embed covert identifying information into a public key without the knowledge of an honest prover. The Issuer simply uses a different public key PK_I to generate DAA Certificates for each prover with which it interacts. As a result, the Issuer will be able to compile a table mapping between a prover's public EK (its unique key) and the public key used to generate the DAA Certificate for this prover.

Suppose a verifer has obtained the Issuer's public key PK_I from the prover. Whenever a prover executes the DAA Sign protocol with a colluding verifier (i.e. one that colludes with the Issuer to identify a prover), and assuming that the public key PK_I used to generate its DAA Certificate is communicated to a verifier via the prover, the DAA Issuer and the colluding verifier can easily link the transactions of a prover and uniquely identify it. A verifier needs only inform the DAA Issuer of the value of PK_I. The DAA Issuer can then consult the EK-PK_I mapping it has compiled, and determine the prover's EK. Similarly, with the aid of the Issuer, colluding verifiers are able to uniquely identify a particular prover.

5.3 How Realistic is the Rudolph Attack?

We now consider how the Rudolph attack might work in practice, and in particular we examine two possible attack scenarios.

5.3.1 Scenario 1: Linking large numbers of users

In this scenario, the aim of the DAA Issuer is to identify large numbers of provers. We believe that this attack scenario is infeasible in practice. We demonstrate (i) why this attack scenario is unrealistic (even if all the verifiers collude) because of the communications and computation burden involved in performing such an attack; and (ii) how the attack can easily be detected if there is at least one honest verifier:

(i) Suppose we have a scenario in which there is one DAA Issuer, n provers (all joining the network or system at different times), and k verifiers. Suppose all the k verifiers are colluding with the Issuer in an attempt to link or uniquely identify the provers. For the attack to work, the colluding verifiers have to shoulder an additional communication and computational burden (see Table 5.1 for a summary of the communication and computational overheads). First and foremost, to be able to link all the n provers, the DAA Issuer needs to use n different public keys PK_I, one for each prover. These n public keys will also need to be communicated to each of the verifiers. If a trusted directory is used to hold copies of the public keys, then the Issuer would need to upload a total of n different public keys to this directory (if the provers join at different times then this may involve sending up to n separate upload messages). If the verifiers obtain the public keys from the Issuer directly, then the total communications overhead for an Issuer may be up to nk messages (as compared to k messages if the Issuer only uses one key).

A verifier, regardless of the mechanism used to retrieve Issuer public keys, will need to obtain up to n Issuer public keys for the n provers. This means that the communications overhead for a verifier may be up to n messages. Furthermore, whenever there is a new prover, the verifiers might need to obtain the new public key for this prover.

We now point out why launching the Rudolph attack also has a significant computational overhead. Firstly, the generation of the DAA Issuer public key PK_I involves performing a non-interactive zero knowledge proof (using the Fiat-Shamir heuristic) [27]. This potentially involves the DAA Issuer performing at least 160 modular exponentiations (which could go up to 6×160, one for each of the public key components g, h, S, Z, R_0 and R_1). This contradicts Rudolph's claim that the process of generating a large number of public keys can be performed efficiently [132].

Secondly, the work to be performed by the verifier in trying to identify the prover may become infeasibly large, depending on how Issuer public keys are distributed. If the prover sends the Issuer-signed certificate for the Issuer public key PK_I to the verifier as part of the DAA Sign protocol, as assumed by Rudolph, then there is no problem. However, if the key is obtained from a directory, then significant computational problems arise. This is because the verifier will have no way of knowing which of the n Issuer public keys have been used to create the DAA certificate, and hence which of them should be used to attempt to verify the DAA Signature. The only solution would be for the verifier to attempt to verify the signature using every possible key, which would involve up to n verifications. Given the ubiquity of trusted computing hardware, a typical value of n might be 10^5 or 10^6, which would make such a process computationally infeasible.

(ii) The attack will easily be discovered if there is at least one honest verifier or if Issuer public keys are stored in public directories, as we now describe.

- Consider first an environment in which there is at least one honest verifier. When the honest verifier attempts to verify a DAA Signature, it first needs to retrieve the Issuer's public key, either directly from the Issuer or from a trusted directory. If the Issuer (or the trusted directory) submits a large number of public keys to the verifier, then suspicions about its trustworthiness will immediately be aroused. Even if the honest verifier is given the Issuer public key by the prover rather than retrieving it from a directory, then it could still detect misbehaviour if it keeps a log of all

Table 5.1: Communications and computation costs for honest and colluding entities.

Type of Costs	Costs Incurred By			
	Honest Issuer	Honest Verifier	Colluding Issuer	Colluding Verifier
Communication (no. of messages)	1	1	nk	n
Computation (no. of DAA Sign Verifications)	-	1	-	n

the Issuer public keys that it has been passed. If one particular Issuer is using large numbers of different public keys then this will quickly become obvious to such a verifier.

- Suppose an Issuer uploads multiple public keys to a directory or other repository. This will immediately be obvious both to the operator of the directory (which may report suspicious behaviour) as well as to any user of the directory.

5.3.2 Scenario 2: Linking a small set of users

If the aim of the DAA Issuer is to link all transactions involving a single user (e.g. a high-profile user or one that makes high value transactions), or a very small set of users, then the attack is much more likely to succeed in a way that is hard to detect. For example, if a DAA Issuer only wants to distinguish transactions involving one user, then the Issuer only needs to have two public keys PK_I. In such a case, the communication and computation problems discussed in the previous section would not be an issue, nor would there be a large number of Issuer public keys in circulation to arouse suspicions.

Nevertheless, if a verifier deals with many provers who are clients of the same Issuer, then the suspicions of an honest verifier might be aroused if one Issuer public key, or some small set of such keys, is used much less than others. In particular, if the DAA

Signatures are of the name-base type, allowing a verifier to link DAA Signatures signed by the same prover (TPM) for the same verifier, then the verifier will be able to observe significant differences in the numbers of clients for an Issuer's public keys.

5.4 Preventing the Rudolph Attack

Despite the issues raised in the previous section, the Rudolph attack will work if a verifier obtains the Issuer's public key from a prover (as described in [132]), and if no honest verifier keeps track of the Issuer public keys it has received or if the goal of the attackers is only to track a few users. Even worse, a prover would be completely oblivious to such an attack, as there is no way for a prover to tell if a DAA Issuer is embedding covert identity information into the public key that is used to generate its DAA Certificate (e.g. by using a different public key for each prover).

We now examine a number of possible ways of preventing the Rudolph attack. We also discuss the limitations of these approaches.

5.4.1 Modifying the TCG Specifications

We first observe that addressing the root cause of the problem would involve changing the TCG specifications to prevent a DAA Issuer from self-certifying an arbitrary number of public keys. This could be achieved by requiring the Issuer to use a private key for which the public key has been directly certified by a third party CA (in the notation used above, this would mean that the issuer key CK_I would be used to generate DAA certificates).

There are two problems with such a approach. Firstly, the CA would then need to be trusted not to generate large numbers of certificates for an Issuer. Whilst this could be supported by requiring any CA that generates Issuer certificates to adhere to an appropriate Certification Practice Statement, it still means that a significant amount of trust is placed in a single third party, a situation which the design of DAA seeks to avoid.

Secondly, as explained above, this means that the key life cycle of the TPM and the Issuer become linked. Addressing this issue would require further changes in the operation of the TPM.

An alternative approach would retain the two levels of Issuer public keys, but would require both types to be certified by a CA. As in the existing scheme, the first level key would be used to compute the DAA secret, and the second level key would be used to create the DAA Certificate. Certificates for second level keys could reference the first level key to link the two together. This could address the Rudolph attack without causing a key life cycle issue.

5.4.2 Using a Trusted Auditor

We next explore another possible approach which does not involve modifying the TCG specifications too much (or at all). We propose that a prover obtains DAA Certificates only from DAA Issuers that use the same public key for a very large set of users. Thus the challenge is to enable a prover to determine the key usage behaviour of a DAA Issuer.

If a prover is able to obtain assurance that a specific Issuer's public key has being used more than a certain number of times (i.e. to generate a certain number of DAA Certificates), then it is immediately able to derive confidence that it will not be uniquely identified, and at the same time gain information about the level of anonymity that it is being afforded. For example, if a prover knows that the public key used to generate its DAA Certificate has been used to generate a thousand other DAA Certificates, then it knows that it cannot be distinguished from a thousand other entities. On the other hand, if it knows that a particular public key has only been used to generate three other DAA Certificates, then the level of anonymity afforded to it is potentially very limited.

We now suggest two possible approaches designed to give a prover this type of assurance. We also discuss the possible limitations of the suggested approaches.

5.4.2.1 A modification to the use of DAA.

This approach requires the introduction of a new type of trusted third party, which we refer to as a *Trusted Auditor* (TA). We make use of the TA (which is not necessarily unique) to give provers assertions about the "trustworthiness" of individual DAA Issuers. Since the CA needs to be trusted by the protocol participants, and since it is already employed to certify the longer-term public key CK_I of the Issuer, a CA could act as a TA, although this does need to be the case.

We propose that the following additional steps be performed by a prover during the DAA Join protocol. During the Join phase, and after a DAA Certificate has been successfully created, the prover establishes a secure channel with the TA. This can be achieved by the prover first establishing a unilaterally authenticated secure channel with the TA using a public key certificate for the TA, e.g. using SSL/TLS. The prover can then send its EK credential to the TA via this channel, and use this credential to authenticate itself, e.g. by decrypting data sent to it encrypted using the EK. Finally, the prover uses this channel to send a statement that a specific DAA Issuer has used a particular public key PK_I to derive its DAA Certificate, i.e.

$$Prover \rightarrow TA : ID_{Issuer}, CK_I, PK_I$$

Using these messages, the TA can compile a list of the form given in Table 5.2. A copy of this list signed by the TA (to prove authenticity) can then be communicated to a prover prior to the Join Phase. Since it is desirable not to publicise the public EK values of individual trusted platforms, this information can be removed from the list before it is distributed. The list of public EKs can be replaced by the total number of different EKs for which credentials have been generated using a particular Issuer public key.

This approach suffers from one problem. The public part of the Endorsement Key (EK) of every prover is revealed to the TA. Since the EK pair for a trusted platform is fixed, the public EK functions as a fixed identifier for a platform, and hence revealing it is not desirable. Indeed, DAA was introduced to avoid the need to reveal the link between the public EK and other prover public keys to a Privacy CA. Nevertheless, this scheme does not pose the same threat to user privacy as the

Table 5.2: Mapping of EKs to a PK_I for individual Issuers

No.	Issuer Name	Longer-term CK_I	Shorter-term PK_I	Users
1.	Alice	CK_I^A	PK_I^{A1}	EK_1 $\vdots$ EK_{2000}
			PK_I^{A2}	EK_1 $\vdots$ EK_{10}
2.	Bob	CK_I^B	PK_I^{B1}	EK_1 $\vdots$ EK_{200}

use of a Privacy CA, since the TA does not learn the link between the EK and any other prover keys.

5.4.2.2 An alternative approach.

A possible alternative to the above approach avoids revealing the EK to a third party, although it still relies on a TA to provide assertions regarding the trustworthiness of the DAA Issuer (i.e. reporting on the number of DAA Certificates generated for a particular public key for a specific Issuer). A prover and DAA Issuer run the DAA Join protocol as normal, thereby obtaining a DAA Certificate (for the prover's unique DAA Secret f). The prover then conducts an instance of the DAA Sign protocol with the TA, which acts as the verifier. The prover DAA-signs the following information sent to the TA (acting as verifier), so that the TA can compile a table similar to that shown in Figure 5.2.

$$Prover \rightarrow TA : ID_{Issuer}, CK_I, PK_I$$

One possible problem with this approach is that, because no use is made of the EK, a malicious Issuer could fabricate DAA-signed messages of the above form, and send them to the TA. The DAA signature signed for the TA could be of the name-base type, which will guarantee that each DAA secret can only provide one piece of evidence. However, the messages could be based on any number of 'fake' DAA Certificates, that are valid in that they have been created by the DAA Issuer, but have never been sent to a genuine prover. Such messages will be indistinguishable from messages sent from genuine provers, and hence the number of uses of a key can be artificially inflated.

Nevertheless, a table created in this way will still reveal if an Issuer has created more public keys than would be expected in ‘normal’ behaviour; this may be sufficient to deter an Issuer from using the Rudolph attack on a large scale.

5.4.3 A User-Centric Approach

To determine the trustworthiness of an Issuer, two or more users could collaborate to compare the PK_I values that they have obtained from a particular Issuer. If all of them have the same key PK_I, then there is good chance that the Issuer is using the same PK_I for a large set of users. However, if the users find that two or more different keys PK_I have been used, then the trustworthiness of the issuer is immediately called into question. This approach is suited to a distributed or peer-to-peer environment, and does not require the involvement of a trusted third party. Clearly, the effectiveness of the technique will depend on the number of cooperating users.

5.5 Summary

A privacy flaw in DAA was recently pointed out by Rudolph [132]. In this chapter we have analysed the feasibility of attacks exploiting this property. We then examined possible approaches which could be used to prevent (or reveal) the attack as well as the limitations of these approaches. These approaches could make a successful attack very difficult to perform; however, all of the suggestions have certain drawbacks. It remains an open problem to find a ‘perfect’ solution to the Rudolph attack. Indeed, the DAA scheme itself cannot stop an issuer from using a different key with each TPM, no matter whether the key is certified by the Issuer’s longer-term key or by another CA. It is a very tough challenge for any application to completely avoid such a threat.

It should be noted that the Rudolph Attack is not an attack on the DAA protocol itself, but is rather a weakness introduced in the particular use of DAA. In fact, an implementation of an arbitrary group signature scheme might have a similar problem if the size of a group is very small, e.g. for groups containing just a single member. However, in other implementations of group signatures, a group manager might not have any motivation to break the anonymity of its members, because the manager has the ability to determine the identity of a signer from its signature.

Chapter 6

Analysis of a Secret Distribution and Storage Scheme

Contents

In this chapter we analyse a protocol proposed by Sevinç, Strasser and Basin, which uses trusted Computing functionality to secure both the storage of secrets and their distribution from a server to a client. We identify two security weaknesses in the protocol design, namely the absence of server-to-client authentication and the unauthenticated encryption of secrets sent from the server to the client. As a result of these weaknesses, we show how the TPM could be exploited as an oracle, hence undermining the security of the scheme. We also propose modifications designed to make the protocol more secure.

6.1 Introduction

As was discussed in Chapter 4, trusted computing provides many innovative security features such as the protected message exchange mechanism and the remote attestation service. These features have attracted the attention of a number of security protocol designers, who have used them to help to address a variety of security challenges and issues (some of these applications have been described in Section 4.5). Like any other security technology, trusted computing does not remove every security issue. Best practices for protocol design must still be followed to minimise the risk that vulnerabilities are present.

In this chapter we consider a protocol proposed by Sevinç, Strasser and Basin [138], which uses trusted computing functionality to secure both the storage of secrets and their distribution from a server to a client. We describe security weaknesses in this protocol. Specifically, because messages are not authenticated, attackers could exploit the TPM as an oracle for signing arbitrary messages. We go on to suggest possible protocol enhancements which will both prevent the attack and improve the overall security of the scheme.

The remainder of the chapter is organised as follows. In Section 6.2 we introduce the notation used in this chapter, and also describe the commands employed in the Sevinç-Strasser-Basin (SSB) Scheme. Section 6.3 describes the operation of the TPM-based secret distribution and storage scheme, and in Section 6.4 we analyse its security. In the penultimate section we propose protocol modifications designed to improve the security of the scheme. Finally, we provide concluding remarks in Section 6.6.

6.2 Preliminaries

In this section we briefly introduce certain notation, and review the TPM commands used in the SSB protocol. We also state the assumptions upon which the scheme is based.

6.2.1 Notation

The notation used in the protocol descriptions is summarised in Table 6.1.

Table 6.1: Notation

Notation	Description
C	The Client
S	The Server
d_s	Secret data generated by Server S which is to be transferred to the client C
f_a	A flag, which may be set or cleared, used to specify whether a particular key is an AIK key
f_b	A flag, which may be set or cleared, used to specify whether a particular key is a binding key
f_n	A flag, which may be set or cleared, used to specify whether a particular key is a non-migratable key
h_K	A Handle to key K
$Cred_{AIK_{pk}}$	An AIK Identity Credential (as described in Section 4.2.4)
$PCRInfo$	$PCRInfo$ consists (i) a set of PCR indices; and (ii) their respective PCR values.
$SeReq$	Secret Request Message
(A_{pk}, A_{sk})	The public/private key pair of a principal A
(BK_{pk}, BK_{sk})	A pair of public/private Binding Keys
(CA_{pk}, CA_{sk})	The public/private key pair of a CA
(MK_{pk}, MK_{sk})	The pair of public/private Master Storage Keys

6.2.2 TPM Commands

The SSB Scheme uses the following TPM commands.

- $(BK_{pk}, \widetilde{BK}_{sk}) \leftarrow$ `TPM_CreateWrapKey`$(h_{MK}, f_b, f_n, PCRInfo)$: This command generates and outputs an encryption/decryption key pair (BK_{pk}, BK_{sk}). It takes as input (i) a key handle h_{MK} to the master storage key MK that is used to encrypt the newly generated secret key, (ii) a flag that specifies whether the newly generated key is to be used for signing or binding (encrypting), (iii) a flag that specifies whether the key is migratable or non-migratable, and (iv) if it is a binding key, whether the key is to be sealed to a specified set of PCR values. The public key BK_{pk} is returned unencrypted, while the private key

BK_{sk} is returned in encrypted form, i.e.

$$\widetilde{BK}_{sk} = E_{MK_{pk}}(f_b, f_n, PCRInfo, BK_{sk})$$

where $E_{MK_{pk}}(f_b, f_n, PCRInfo, BK_{sk})$ denotes the encryption of the private binding key BK_{sk} (and associated information) using the (non-migratable) master storage key MK_{pk}, and where BK_{sk} is a non-migratable, binding key sealed to the PCR values specified in *PCRInfo*.

- $h_{BK_{sk}} \leftarrow$ `TPM_LoadKey2`$(BK_{pk}, \widetilde{BK}_{sk}, h_{MK})$: This command loads a private decryption key BK_{sk} into the TPM for subsequent use within the TPM. It takes as input (i) the public (encryption) key BK_{pk}, (ii) the corresponding private key in encrypted form $\widetilde{BK}_{sk}$, and (iii) a handle h_{MK} to the master storage key MK. A handle h_{BK} pointing to the loaded key BK_{sk} is returned by the TPM.

- $Cert \leftarrow$ `TPM_CertifyKey`(h_{BK}, h_{AIK}, N): This command produces a certificate for a TPM public key, where the certificate specifies the properties of the key. It takes as input (i) a handle to the loaded private key BK_{sk} (assuming the previous command has previously been executed), (ii) a handle to the certifying/signing key AIK, and (iii) a nonce N. The command then returns a certificate

$$Cert_{BK_{pk}} = Sig_{AIK_{sk}}(f_b, f_n, PCRInfo, N, BK_{pk})$$

The certificate states that the public key BK_{pk} is the public part of a non-migratable binding key pair.

- $P \leftarrow$ `TPM_UnBind`(T, h_{BK}): This command decrypts the input ciphertext. It takes as input (i) ciphertext T, and (ii) a handle h_{BK} which points to the decryption key BK_{sk}. The command then outputs P, the decrypted version of T.

6.2.3 Assumptions

Use of the SSB scheme has the following prerequisites:

1. The server has prior knowledge of:

- the secret data d_s which is possessed by the server prior to the protocol run (only the server has knowledge of d_s prior to the protocol run) and which is to be transferred to C;
- the certification authority's public key CA_{pk}, that can be used to verify the AIK identity credential;
- a set of PCR values (and associated indices) that correspond to a trusted state on the client platform.

2. The client TPM has:

   - generated an AIK key pair and has obtained an identity credential (AIK public key certificate) for AIK_{pk}, the public part of the AIK, from a certification authority trusted by the server;
   - generated a master storage key pair (MK_{pk}, MK_{sk});
   - loaded the private key MK_{sk} of the non-migratable master storage key pair MK;
   - loaded the AIK private key, AIK_{sk}.

3. The client is in possession of:

   - the handle h_{MK} to, and the authorisation data for, the non-migratable master storage key MK for the client TPM;
   - the handle h_{AIK} to, and the authorisation data for, an AIK;
   - an AIK identity credential $Cred_{AIK_{pk}}$, which enables external entities to verify that an AIK is authentic and belongs to a genuine TPM with the specified properties.

4. The operator of the client platform has the authorisation data for the AIK key pair.

6.3 A TPM-based Secret Distribution and Storage Scheme

The scheme proposed by Sevinç, Strasser and Basin (SSB) [138] uses the binding and sealing functions of trusted computing to help securely distribute secrets from a server to a client. The server is not required to trust the client platform, although the server must trust the trusted platform module (TPM) in the client machine to behave in the expected manner. With the aid of this TPM, the server is able to learn the operational state of the platform, and gain assurance that a transferred secret will only be released (decrypted) if the operational state of the client platform is a particular specified trustworthy state. The server is also given assurance that the platform is not in a compromised state before the secret is released to the platform. We now describe this scheme in greater detail (see also Figure 6.1).

1. The client C sends a secret request message $SeReq$ to the server S requesting the transfer of secret data.

2. On receiving the above message, S replies with $PCRInfo$ and a nonce N. $PCRInfo$ specifies the operational state in which the client platform must be, in order for the secret data to be released.

3. C generates a non-migratable encryption/decryption binding key pair (BK_{pk}, BK_{sk}) by invoking the `TPM_CreateWrapKey` command. The command takes as input the handle to the master storage key h_{MK} and $PCRInfo$ (received from step 2), and outputs $(BK_{pk}, \widetilde{BK}_{sk})$. Note that the output private key $\widetilde{BK}_{sk}$ is sealed to the PCR values specified in $PCRInfo$ and is also encrypted under the master storage key MK.

4. C loads the sealed private key BK_{sk} into the TPM so that it is available within the TPM. This is achieved using the `TPM_LoadKey2` command, taking as input (i) the encrypted sealed private key $\widetilde{BK}_{sk}$, (ii) its corresponding public key BK_{pk}, and (iii) the handle to the master storage key h_{MK}. The TPM returns a handle $h_{BK_{sk}}$ to the loaded private key.

5. C creates a certificate for the non-migratable binding public key BK_{pk}, using the `TPM_CertifyKey` command. This command takes as input (i) the handle $h_{BK_{sk}}$ to the loaded private key BK_{sk}, (ii) a handle h_{AIK} to the AIK key, and (iii) a freshly generated nonce N. The command then returns a certificate $Cert_{BK_{pk}}$. The certificate contains (i) the public key BK_{pk}, and (ii) a description of the properties of BK (including whether the key is a binding key and/or a non-migratable key, and also the PCR values to which the private key was sealed).

6. On successful completion of steps 3–5, C sends S the public key certificate $Cert_{BK_{pk}}$ (obtained in the previous step), and the AIK credential $Cred_{AIK_{pk}}$.

7. On receiving the above message, S verifies the two certificates. On successful verification, S is convinced that (i) the public AIK is authentic, and (ii) BK is a non-migratable binding key originating from a genuine TPM, and that it is sealed to the PCR values that the server specified in step 2. S is now happy to send the secret to C.

8. S extracts the encryption key BK_{pk} from $Cert_{BK_{pk}}$, and uses it to encrypt the secret data d_s. S then sends the encrypted secret $E_{BK_{pk}}(d_s)$ to C.

9. On receiving the above message, C decrypts the encrypted secret by invoking the `TPM_UnBind` command. This command takes as input (i) the encrypted secret $E_{BK_{pk}}(d_s)$, and (ii) the handle h_{BK} pointing to the decryption key BK_{sk}. Before performing the decryption, the TPM checks whether the current operational state of the client platform matches the state specified in *PCRInfo*. If the outcome is positive, the encrypted secret is decrypted (unsealed) and returned to C.

1. C $\rightarrow$ S	*SeReq*
2. C $\leftarrow$ S	*PCRInfo*, N
3. TPM $\leftarrow$ C	invokes `TPM_CreateWrapKey`(h_{MK}, *PCRInfo*)
TPM:	generates non-migratable binding key (BK_{pk}, BK_{sk})
TPM $\rightarrow$ C	ouputs ($BK_{pk}, \widetilde{BK}_{sk}$)
4. TPM $\leftarrow$ C	invokes `TPM_LoadKey2`($h_{\widetilde{BK}_{sk}}, h_{BK_{pk}}, h_{MK}$)
TPM $\rightarrow$ C	outputs $h_{BK_{sk}}$
5. TPM $\leftarrow$ C	invokes `TPM_CertifyKey`($h_{BK_{sk}}, h_{AIK}, N$)
TPM $\rightarrow$ C	outputs $Cert_{BK_{pk}}$
6. C $\rightarrow$ S	$Cert_{BK_{pk}}, Cred_{AIK_{pk}}$
7. S:	verifies AIK_{pk}
	asserts BK is binding
	asserts BK is sealed to *PCRInfo*
8. C $\leftarrow$ S	$E_{BK_{pk}}(d_s)$
9. TPM $\leftarrow$ C	`TPM_UnBind`($E_{BK_{pk}}(d_s), h_{BK}$)
TPM:	asserts C is in same state as specified in *PCRInfo*
TPM $\rightarrow$ C	outputs d_s

Figure 6.1: TPM-based secret distribution and storage scheme

6.4 Analysis of the Scheme

We now identify three significant weaknesses in the SSB scheme.

6.4.1 Absence of Server-to-Client Authentication

The protocol does not provide any authentication of the server to the client; in particular, the client has no means of verifying the origin of messages 2 and 8. This means that the client will be issuing commands to the TPM on the basis of an unauthenticated request (message 2), and receiving 'secret data' (in message 8) of unknown provenance and without any integrity guarantees. Whilst server to client authentication is not a specified design goal for the scheme, its absence could have serious implications for practical uses of the protocol.

6.4.2 Preventing the Client from Receiving the Secret

The main design goal of the scheme is to provide assurance to the server that the provided secret remains confidential and can only be accessed by a client platform operating in a specified trustworthy software state.

Since no security assurance is provided to the client by the server (i.e. in the form of server to client authentication), a malicious entity can eavesdrop on message 6, obtain the key BK_{pk}, use it to encrypt an arbitrary message d'_s and then send it to C in step 8. C will decrypt the message, believing it to originate from S. An adversary could also mount a denial of service and cause C and its TPM to perform a large number of decryptions. An adversary could also replay an old encrypted secret, which C will accept as fresh.

The scheme's designers [138] argue that the SSB scheme achieves information theoretic security because, of the four messages (steps 1, 2, 6 and 8) exchanged over a public communications channel (which is assumed to be insecure), only message 8 is a function of the secret. Hence there is no way for an adversary to obtain any information regarding the secret from the first three messages. However, the fact that the message sent in step 8 cannot be linked to the other three messages or to any value representing the current protocol run helps to cause the identified problems.

6.4.3 Exploiting the TPM as an (Signing) Oracle

TPMs are designed to enhance the security of computing platforms and systems, and in the setting of the SSB scheme only the client C has direct access to it. This greatly restricts the ability of an adversary to exploit the TPM via its applications program interface (API). Indeed, access to a tamper-resistant device's API may give an adversary unintended access to secret information, as shown by the work of Bond [25] in attacking the IBM 4758 cryptoprocessor API to recover ATM PINs.

However, the SSB scheme allows the TPM to be exploited as an oracle to generate signatures on values (namely the *PCRInfo*) of an adversary's choosing. To see this, observe that the message in step 2 is sent unencrypted, without integrity or origin authentication protection; thus an adversary could replace it with arbitrary value. In step 6, C responds with a message containing the signature of the TPM on the value sent in message 2, i.e. a value that could have been chosen by an adversary. The exact consequences of this signing oracle are unclear, but are potentially very damaging.

6.5 Discussion

We now consider how the SSB scheme could be modified to address the identified issues.

1. As discussed in the previous section, the TPM can be exploited by an adversary as a signing oracle because the message in step 2 is unauthenticated. As a countermeasure, the message sent in step 2 could be signed by S. We additionally suggest the inclusion of a timestamp t_s so that the client can also verify that the message is fresh. Message 2 would then have the form:

$$S \rightarrow C : \mathit{PCRInfo}, N, t_s, \mathit{Sig}_{S_{sk}}(\mathit{PCRInfo}, N, t_s), \mathit{Cert}_S$$

   where S_{sk} denotes the private key of S, and Cert_S is the public key certificate of S. Note that C will need a copy of the public key of the CA which signed Cert_S.

2. Authentication of the encrypted secret sent in step 8 can be provided by adding a server signature, so that the message becomes:

$$S \rightarrow C : E_{BK_{pk}}(d_s), N, t_s, Sig_{S_{sk}}(E_{BK_{pk}}(d_s), N, t_s).$$

3. In the SSB scheme, the server forces the client (via *PCRInfo* in step 2) to be in a specific trusted software state before it can decrypt the secret. However, in practice it is very difficult for a device to arrange to be in a specific software state; the slightest variations in the sequence of measured software will cause a non-matching *PCRInfo*. A more practical alternative would be for the client to seal a newly produced key to its current state, and the server could then ascertain if this particular state is to be trusted. This could be done by checking against a list of known trustworthy states, for example as produced by the device manufacturer.

   This would also make the protocol more efficient, as the message in step 2 would no longer need to be sent.

The above modifications only make the SSB scheme resistant to the attacks we have identified; a more comprehensive analysis using a formally defined security model would clearly be highly desirable.

6.6 Summary

In this chapter we considered the SSB Scheme, designed to secure the distribution of secrets from a server to a client. We have highlighted a number of security issues in this scheme which may prevent it from achieving its objectives. We have also suggested several minor modifications to the scheme which may prevent the security attacks from being realised.

Part III

Applications of Trusted Computing to Secure Ubiquitous Services

Chapter 7

Secure and Private Service Discovery

Contents

Most authentication schemes are based on authenticating the identity of a principal in one way or another. This method of authentication is commonly known as entity authentication. In emerging computing paradigms, which are highly dynamic and mobile in nature, entity authentication alone may not be sufficient or even appropriate, especially if a principal's privacy is to be protected. In order to preserve the privacy of a principal, other attributes (such as location or trustworthiness) of a principal may need to be authenticated to a verifier. In this chapter we propose Ninja, an authentication scheme for use in a mobile ubiquitous environment. Ninja is a *non-identity based authentication scheme, in which the trustworthiness of a user's device is authenticated anonymously to a remote Service Provider (verifier) during the service discovery process. The scheme relies on the use of Trusted Computing functionality.*

7.1 Introduction

As discussed in Chapter 3, in the Mobile VCE[1] Core 4 research programme on Ubiquitous Services, it is envisaged that, in a mobile ubiquitous environment (as shown in Figure 3.1), users (using their mobile devices and and available network access technologies) will be able to seamlessly discover, select, and access a rich offering of services and content from a range of service providers. To realise this vision, security and privacy issues must be addressed from the outset, alongside other technological innovations. Only if users are confident that their security and privacy will not be compromised will ubiquitous services be widely adopted.

As shown in Figure 3.1, one of the primary aims for a user is to access the services that are offered. But, before any services can be accessed and consumed, they must first be found via a process known as service discovery. Many service discovery schemes [34, 51, 148, 164] have been proposed for ubiquitous environments; unfortunately, very few address security and privacy issues from the outset (one exception being the scheme of Zhu et al. [166, 167]), despite the fundamental importance of such issues. It is imperative that the process of service discovery is conducted in a secure and private way, in order to protect the security and privacy of both users and service providers. One fundamental security requirement is mutual authentication between a user and service provider.

Authentication is important for several reasons. Firstly, it is a basic security service upon which a range of other security services (e.g. authorisation) can be built. Secondly, it gives users and service providers assurance that they are indeed interacting with the intended parties, and not with a malicious entity. Unfortunately, conventional entity authentication [65] may not be adequate for a ubiquitous environment [43], because an identity may be meaningless in such a setting. Instead, other

[1]http://www.mobilevce.com/

user attributes [43] may need to be authenticated to a service provider. Furthermore, consumers are becoming increasingly concerned about their privacy [21, 24] and the potential risks (such as identity theft) of leaving any form of digital trail when making electronic transactions. Given a choice, users may prefer to interact with service providers anonymously (or pseudonymously). Under these circumstances, it may, in fact, be undesirable to authenticate the identity of a user. We also note that preserving user privacy can be particularly challenging in a ubiquitous environment [33, 158].

If, on the other hand, privacy is preserved (through user anonymity), how can a service provider be convinced that an anonymous user is trustworthy? This is the challenge addressed in this chapter.

We propose Ninja, a non-identity based, privacy preserving, mutual authentication scheme designed to address the service discovery security and privacy challenges arising in a mobile ubiquitous environment. During service discovery, a service user and service provider are mutually authenticated in a way that preserves user privacy. Instead of authenticating the user identity to a service provider, the user's trustworthiness is authenticated. The scheme employs two key functionalities of Trusted Computing (TC) technology [11, 100], namely, the Integrity Measurement, Storage and Reporting Mechanism (IMSR), and the Direct Anonymous Attestation (DAA) Protocol (discussed in Sections 4.3.1 and 4.4.2, respectively). We therefore implicitly assume that a user device is equipped with TC functionality; current trends suggest that this is a reasonable assumption for the near future. Ninja is an application layer solution, and possesses many desirable security and privacy properties such as user anonymity, service information confidentiality, unlinkability, and rogue blacklisting.

The remainder of the chapter is organised as follows. In Section 7.2, we discuss service discovery security and privacy issues. Section 7.3 reviews related work, and Section 7.4 introduces the building blocks of the Ninja Scheme. In Section 7.5, we present the Ninja authentication scheme, and in Section 7.6 we analyse its security. Finally, conclusions are drawn in Section 7.7.

7.2 Service Discovery Security & Privacy Issues

In this section we focus on the security and privacy issues arising from the service discovery process in a ubiquitous computing environment.

7.2.1 Adversary Model

Service discovery typically involves interactions between a user, the user's device, a service provider, and, at times, a trusted third party. Unfortunately, these entities may be malicious, and pose a variety of threats to the service discovery process and to the participating entities. Against this backdrop, we identify eight adversarial settings, covering both active and passive adversaries. They are as follows:

1. **Innocent User with Malicious Device (IUMD).** Unbeknownst to the user, his/her device has been compromised (e.g. with malware, keystroke-logger, etc).
2. **Malicious User with Trustworthy Device (MUTD).** An MUTD is a malicious user who has taken physical control of (e.g. stolen) another entity's device.
3. **Malicious User with Malicious Device (MUMD).** The combination of IUMD and MUTD.
4. **Malicious Service Provider(s) (MSP).** An MSP's main motive is to masquerade to a user as a legitimate service provider.
5. **Curious Service Provider(s) (CSP).** A CSP is not malicious, but seeks only to learn more about the behaviour of its users.
6. **Malicious Man-in-the-Middle (MitM).** A MitM's actions are intended to disrupt the proper operation of the service discovery process.
7. **Curious Trusted Third Party (CTTP).** A CTTP performs its role correctly, but also seeks to learn about the activities and habits of a user.
8. **Passive Eavesdropper (PE).** A PE does not disrupt the communication, but monitors it to learn the content and the entities involved.

7.2.2 Security and Privacy Threat Model

We now consider possible service discovery threats. We also consider what threats are posed by each of the above adversarial settings, and present them in a Threats versus Adversary Matrix (in Table 7.1). The service discovery threats are as follows:

1. **Spoofing.** A malicious entity may masquerade as a legitimate service provider or service user by sending false service advertisements/requests, through replay, or by man-in-the-middle attacks.

2. **Information Disclosure.**

   (a) **User's Personally Identifiable Information (PII):** During the process of service discovery, a user's PII, such as his/her identity (e.g. in the form of a long lived key) or physical location, might be revealed (either willingly or unwillingly) to a service provider or passive eavesdropper.

   (b) **Service Information (SI):** By observing the service information exchanged by a user and service provider (e.g. the service request types), a passive adversary may build up a profile of the user. This information could later be used to predict future patterns and habits of the user. The privacy of the user is potentially compromised as a result.

3. **Profile Linking.** Colluding service providers may buy, sell or exchange information about their users or customers. This could not only provide service providers with monetary benefits, but also enhance their business intelligence and gain competitive advantage, e.g. if they are able to build more comprehensive user profiles (with or without user permission). Finally, the consequences for user privacy could be even more serious if a trusted third party colludes with service providers.

4. **Encouragement of Rogue Behaviour.** With the knowledge that privacy enhancing technologies are employed to protect their identities, users might be tempted to "misbehave" or act maliciously, since it may be difficult or even impossible for service providers to determine who is misbehaving.

Table 7.1: Threats and Adversary Matrix

Threats vs Adversary	IUMD	MUTD	MUMD	MSP	CSP	MitM	CTTP	PE
Spoofing	✓	✓	✓	✓		✓		
User Identity Disclosure	✓	✓	✓	✓	✓	✓	✓	✓
SI Disclosure	✓		✓		✓	✓		✓
User Profile Linking					✓	✓	✓	
Rogue Behaviour Denial	✓		✓					

7.2.3 Specific Security and Privacy Requirements

From the above threat analysis, we can derive the corresponding security and privacy requirements:

1. **Mutual Authentication.** This is one of the most important requirements, as it can prevent spoofing (by malicious users or service providers). The mutual authentication scheme should also be designed to prevent replay and man-in-the-middle attacks. To protect privacy, a user may want to remain anonymous to a service provider. So, instead of authenticating his identity to a service provider, the user may want to somehow prove or authenticate his "trustworthiness" to the service provider.

2. **User Anonymity.** Unique user identifying information (e.g. an identifier or a long lived key) should not be divulged to a service provider during service discovery. For example, a user may wish to interact with service providers using a pseudonym.

3. **Service Information Confidentiality.** To preserve the privacy of the user, it may be desirable to protect the confidentiality of the service information, e.g. by encrypting part or all of certain messages.

4. **Unlinkability.** Colluding service providers should not be able to link the activities of a user. Similarly, if a trusted third party colludes with a service provider, they should not be able to correlate the actions of a particular user. In other words, it should be impossible for colluding service providers to tell if two sets of prior service transactions (made with different providers) involved the same or different user(s).

5. **Transaction Linkability/History.** For billing or other purposes, it may be necessary for a service provider to maintain the transaction histories of its users. A service provider may thus need to be able to determine whether a particular user is a repeat user (and, if so, which one) or a first time user, whilst still being unable to determine the unique identity of the user. This is not always a requirement, and providing it may require user consent.

6. **Rogue Blacklisting.** Service providers should be able to identify and/or blacklist malicious and untrustworthy hosts.

7.2.4 Challenges

We need to devise a mutual authentication scheme that meets all the above requirements. This is particularly challenging for several reasons. Conventional mutual authentication schemes normally require the user identity to be authenticated to a verifier. But here user privacy is a priority, and so user anonymity is required during authentication. How then can we convince a service provider that an anonymous user is trustworthy? Also, if user anonymity is provided, how can we detect malicious or illegitimate users? We are, in fact, trying to achieve security and privacy concurrently, whilst protecting the interests of both users and service providers. This is the challenge addressed here.

Ninja allows a user to authenticate the service provider, whilst simultaneously allowing a service provider to anonymously authenticate a user's trustworthiness. The scheme is so called because the process is to some extent analogous to the process of a ninja assassinating a person in Japanese folklore[2].

7.3 Related Work

Apart from being unsuitable for ubiquitous computing environments [164], existing service discovery approaches (such as Java Jini [144], UPnP [153], IETF's Service

[2]A ninja is asked to assassinate someone (Bob) whom he has never met; he is only given Bob's photograph. When they meet, the ninja authenticates Bob physically. Bob, on seeing a ninja with a sword, knows (trusts) that the ninja wishes to kill him, but does not need to know the ninja's real identity, whose anonymity is preserved.

Location Protocol (SLP) [68, 69], DEAPspace [108] and Salutation [134]) do not address the privacy issues raised here. Zhu et al. [165, 166] describe a privacy preserving service discovery protocol, in which users and service providers progressively reveal Personally Identifiable Information (PII) to each other. A user's PII is eventually divulged to a service provider, and so service providers could still collude to link user activities. Abadi and Fournet [3] proposed a private authentication scheme which protects two communicating principals (i.e. their identity and location) from third parties. This only protects the privacy of a user's PII against eavesdropping third parties, and not from the service providers. Ren et al.'s privacy preserving authentication scheme [126] uses blind signatures and hash chains to protect the privacy of service users. This scheme requires a mobile user and service provider to authenticate each other via out of band mechanisms, prior to a privacy-preserving service interaction. This may not be a realistic approach for a mobile ubiquitous environment.

In the k-Times Anonymous Authentication scheme [147], a user can anonymously access a service a predetermined number of times (as decided by the service provider). This approach is extremely inflexible for a ubiquitous environment. For example, a service provider cannot prevent a user found to be malicious from having future service interactions. In the Anonymous Authentication scheme due to Chowdhury et al. [39], users interact with different service providers using different surrogates (one-time values) on every occasion, to preserve user anonymity. However, the trusted 'Issuing Authority', can still link user activities. Similarly, in v1.1 of the TCG specifications [11, 152], a user's activities are unlinkable by different service providers, but if the trusted 'Privacy CA' colludes with the service providers then the activities of a user can be linked and his/her privacy compromised (as discussed in Section 4.4.1). In the Ninja authentication scheme (described in Section 7.5), the trusted third party, i.e. the DAA Issuer, is unable to link the activities of a user, even if it colludes with service providers.

7.4 Building Blocks

In this section, we introduce techniques that are used to help build the Ninja Authentication Scheme.

7.4.1 Diffie-Hellman Key Agreement

The scheme we describe makes use of the Diffie-Hellman key agreement protocol, described in Section 2.4.6.

7.4.2 Trusted Computing

To achieve the security objectives set out in Section 7.2.3, Ninja uses two Trusted Computing functions, namely the Integrity, Storage, and Reporting Mechanism (IMSR), and the Direct Anonymous Attestation (DAA) Protocol. Together, they allow a platform to anonymously authenticate its operational state to a remote service provider. A detailed description of these mechanisms can be found in Chapter 4.

7.5 The Ninja Authentication Scheme

In this section, we present the Ninja authentication scheme, designed to mutually authenticate a user (via his platform) and a service provider, whilst preserving the privacy of the user. The Ninja scheme is intended to be used during the service discovery process, immediately prior to service provisioning. It is designed to meet the security and privacy requirements set out in Section 7.2.3.

We first introduce the entities participating in the protocol. We then state the assumptions upon which the scheme is based. Finally, we describe the operation of the scheme.

7.5.1 The Entities

The entities involved in the protocol are as follows:

- the **Service User**, often a human, is the end consumer of a service;

- the trusted platform, or **Platform**, is the device which a service user uses to interact with other entities;
- the **DAA Issuer** issues DAA Certificates to legitimate platforms;
- the **Service Provider** is an entity that provides service(s) or content (e.g. music, videos, podcasts) to service users via the platforms. A service provider also acts as the verifier of a platform's DAA Signatures.

7.5.2 Working Assumptions

The correct working of the scheme relies on a number of assumptions.

- The service user is already authenticated to the platform.
- The platform running the Ninja protocol is equipped with TC functionality conforming to v1.2 of the TCG specifications (see Chapter 4).
- Service users and service providers have agreed on the cryptographic functions to be used as part of the protocol. These are as follows.
  - A hash-function H.
  - A symmetric encryption algorithm E, where $E_K(X)$ is used to denote the encryption of data X using the secret key K.
  - A MAC algorithm, where $\text{MAC}_K(X)$ denotes a MAC computed on data X using the key K.
  - A digital signature algorithm, where $Sig_K(X)$ is used to denote a signature computed on data X using the private signature key K.
- Service users only obtain DAA Certificates from trustworthy DAA Issuers (i.e. those that use the same public key for a very large set of users in order to avoid the Rudolph attack discussed in Chapter 5).
- Each service provider possesses one or more X.509v3 public key certificates (described in Section 2.4.7), issued by Certification Authorities (CAs) which it trusts. The platform is equipped with the public keys of these CAs, and is capable of periodically receiving Certificate Revocation List (CRL) updates issued by these CAs.

- Service users and service providers have loosely synchronised clocks (e.g. within an hour of each other). This enables service users and service providers to check that a service advertisement or service reply message is sufficiently fresh.
- All parties must agree on Diffie-Hellman parameters p and g, subject to the constraints listed in Section 2.4.6.

Finally note that the scheme is independent of the underlying transport and network layer protocols, as it is purely an application layer solution.

7.5.3 The Scheme

The notation used in the protocol descriptions below is summarised in Table 7.2.

Table 7.2: Notation

Notation	Description
P	The Platform
SP	The Service Provider
I	The DAA Issuer
f	A non-migratable DAA secret value generated by the TPM
v', v'', e	DAA parameters (described in Section 3.2)
σ	A DAA Signature
p, g	System parameters for DH–key agreement
$SrvAdv$	Service Advertisement Message
$SrvRep$	Service Reply Message
$SrvInfo$	Service Information
$AdvID$	An Advertisement ID number
(A_{pk}, A_{sk})	The public/private key pair of a principal A
(AIK_{pk}, AIK_{sk})	A pair of public/private Attestation Identity Keys
(EK_{pk}, EK_{sk})	The pair of public/private Endorsement Keys

As shown in Figure 7.1, the Ninja authentication scheme involves three distinct phases, namely, the *Join Phase*, the *Mutual Authentication Phase* and the *Verification Phase*. We now describe the workings of each phase.

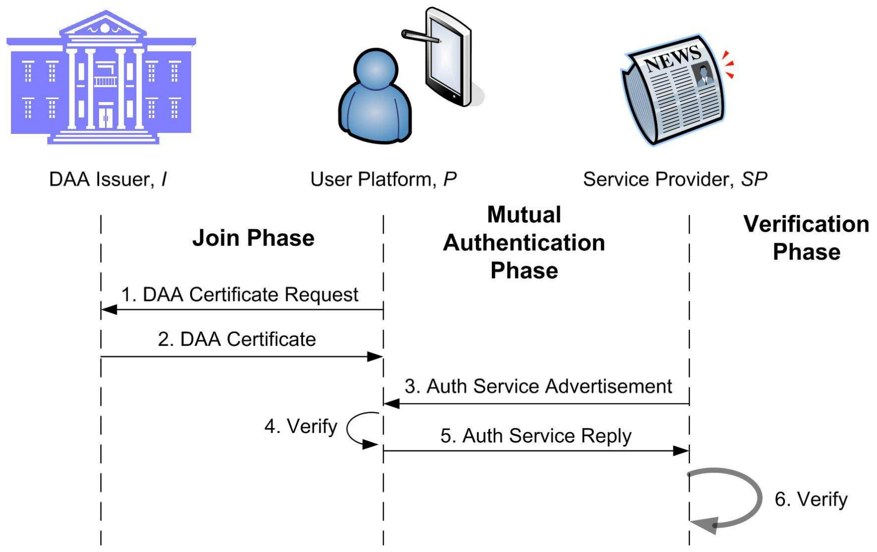


Figure 7.1: The Ninja Authentication Scheme

7.5.3.1 Join Phase

The *Join Phase* enables a platform to obtain a *DAA Certificate* from a *DAA Issuer*. The platform uses this DAA Certificate in the mutual authentication phase to anonymously authenticate itself to a service provider. The entities involved are the *Platform*, P, and the *DAA Issuer*, I. Note that the Join Phase is identical to the DAA Join Protocol specified in [27]; it may have taken place before a device is shipped to the user. The sequence of events is as follows (see also Figure 7.2).

1. The TPM in the user platform generates its DAA Secret value f and a random value v'. It computes U and N_I (as described in Section 4.4.2), and then sends U, N_I, and its Endorsement Public Key, EK_{pk} to the DAA Issuer.

2. To verify that U originates from the TPM in the platform that owns EK_{pk}, the DAA Issuer engages in a simple challenge-response protocol with the platform. It generates a random message m, and encrypts m using EK_{pk}. It sends the challenge, $Chl = \bar{E}_{EK_{pk}}(m)$, to the platform.

3. If the platform owns EK_{pk}, it should have the corresponding EK_{sk}, and hence is able to decrypt $\bar{E}_{EK_{pk}}(m)$, to retrieve m. The TPM in the platform then computes and sends the response $r = H(U||m)$ to the DAA Issuer.

4. The DAA Issuer computes $H(U||m)$ using the value of m sent in step 2, and compares the result with the value of r received from the platform. If the two values agree then the DAA Issuer is convinced that U originated from the TPM that owns EK_{pk}.

5. Finally, the DAA Issuer generates v'' and e and computes A (as described in Section 4.4.2). The DAA Issuer then sends (A, e, v'') to the platform.

6. The DAA Certificate is (A, e, v), where $v = v' + v''$. The DAA Issuer does not have full knowledge of the DAA Certificate, since the certificate was jointly created by the platform and the DAA Issuer. This property helps preserve the anonymity of the user/platform.

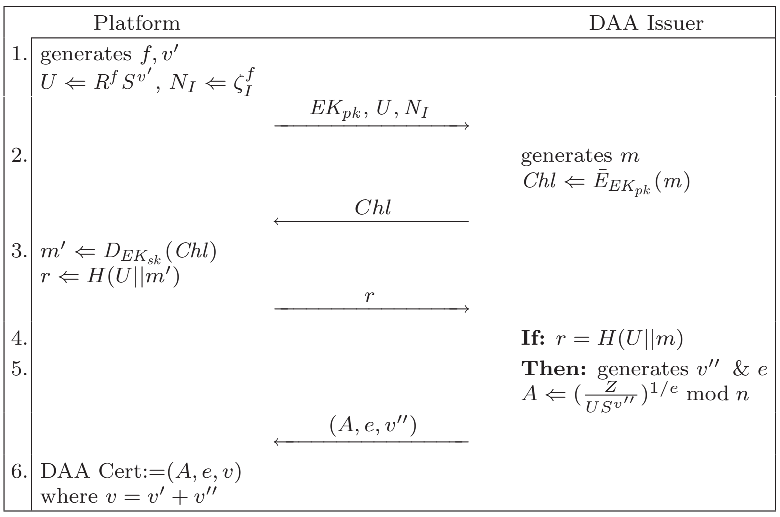

Platform
DAA Issuer
1. generates f, v'
$U \Leftarrow R^f S^{v'}$, $N_I \Leftarrow \zeta_I^f$
EK_{pk}, U, N_I
2. generates m
$Chl \Leftarrow \bar{E}_{EK_{pk}}(m)$
Chl
3. $m' \Leftarrow D_{EK_{sk}}(Chl)$
$r \Leftarrow H(U||m')$
r
4. **If:** $r = H(U||m)$
5. **Then:** generates v'' & e
$A \Leftarrow (\frac{Z}{US^{v''}})^{1/e} \bmod n$
(A, e, v'')
6. DAA Cert:=(A, e, v)
where $v = v' + v''$

Figure 7.2: Join Phase

7.5.3.2 Mutual Authentication Phase

Service discovery typically involves the exchange of service advertisement and service reply messages between a user and a service provider. To avoid increasing the communication overheads, we incorporate the authentication mechanisms into these messages. In other words, service discovery and mutual authentication take place concurrently. We now examine how the messages are constructed to achieve mutual authentication.

The service provider, *SP*, initiates the service discovery and mutual authentication processes by constructing and sending an authenticated service advertisement message, as follows (see also Figure 7.3).

S1. SP generates a random number b and computes $g^b \bmod p$. These values are used later as part of a Diffie-Hellman key agreement to establish a shared key with the user.

S2. SP constructs the service advertisement message as follows:

$$SrvAdv = (ID_{SP}, SrvInfo, AdvID, N, t_{sp}, g, p, g^b \bmod p),$$

where $SrvInfo$ contains information about the offered service and $AdvID$ is an advertisement identifier chosen so that it is unique within the lifetime of SP_{pk}. N is a nonce chosen so that the probability of an SP ever using the same nonce twice during the lifetime of a signature key pair (i.e. SP_{pk} and SP_{sk}) should be negligible. That is, if the number of possible nonces is s, (and assuming that nonces are chosen uniformly at random from the set of all possible values) then the number of times that a key pair should be used should be significantly less than $s^{0.5}$.

S3. Service provider SP signs $SrvAdv$ using its private key, SP_{sk}, to obtain the signature $Sig_{SP_{sk}}(SrvAdv)$. SP then broadcasts $SrvAdv$, $Sig_{SP_{sk}}(SrvAdv)$, and $Cert_{SP}$ to the platforms of prospective service users:

$$SP \rightarrow \text{platforms} : SrvAdv, Sig_{SP_{sk}}(SrvAdv), Cert_{SP}.$$

Suppose that a prospective user receives the above service advertisement (via his/her platform), and is interested in the advertised service. The user's platform then authenticates the service provider by retrieving SP_{pk} from $Cert_{SP}$, using it to verify $Sig_{SP_{sk}}(SrvAdv)$, and checking to see if the timestamp is valid. If the verification outcome is satisfactory, then, the service provider is deemed authenticated to the user.

The platform now anonymously authenticates itself (i.e. its trustworthiness) to the service provider, as follows (see also Figure 7.3).

P1. The platform generates an AIK key pair (AIK_{pk}, AIK_{sk}).

P2. The platform assembles *PCR*, the set of PCR values to be supplied to the service provider; the platform also extracts SML, the part of its Stored Measurement Log necessary to interpret the PCR values in *PCR*. To prove that the PCR values originate from the TPM, the TPM signs the PCR values along with the nonce N (obtained from *SrvAdv*), using AIK_{sk} (as generated on the previous step), to create:

$$Sig_{AIK_{sk}}(PCR||N).$$

The Nonce N is included to prevent replay attacks.

P3. The platform computes $\zeta = H(ID_{SP})$. It then creates a pseudonym, $N_v = \zeta^f$ (where f is the DAA Secret generated during the join phase) for use when interacting with the service provider.

P4. To prove that the AIK (from steps P1 and P2) originates from a genuine TPM, the platform DAA-Signs AIK_{pk} using f, *DAA Certificate*, and the other public parameters of the system. The output is the DAA Signature σ (which also includes ζ and N_v).

P5. To complete the Diffie-Hellman key agreement, the platform generates a random number a, and computes:

$$g^a \bmod p \quad \text{and} \quad K = (g^b)^a \bmod p.$$

P6. The platform constructs the Service Reply message as:

$$SrvRep = (AdvId, SrvInfo, AIK_{pk}, \mathrm{SML}, Sig_{AIK_{sk}}(PCR||N), \sigma, t_p).$$

P7. The platform encrypts *SrvRep* using the key K_1 and computes a MAC on the encrypted *SrvRep* using the key K_2, where $K = K_1||K_2$, to give:

$$E_{K_1}(SrvRep) \quad \text{and} \quad \mathrm{MAC}_{K_2}(E_{K_1}(SrvRep)).$$

P8. The platform sends $E_{K_1}(SrvRep)$, $\mathrm{MAC}_{K_2}(E_{K_1}(SrvRep))$, and $g^a \bmod p$ to the service provider.

$$P \rightarrow SP : E_{K_1}(SrvRep), \mathrm{MAC}_{K_2}(E_{K_1}(SrvRep)), g^a \bmod p.$$

	Platform		Service Provider
S1.			generates b, $x \Leftarrow g^b \bmod p$
S2.			constructs $SrvAdv{:=}(ID_{SP}, SrvInfo, AdvID, N, t_{sp}, g, p, x)$
S3.			$S_x \Leftarrow Sig_{SP_{sk}}(SrvAdv)$
		$\xleftarrow{SrvAdv, S_x, Cert_{SP}}$	
P1.	generates AIK_{pk}, AIK_{sk}		
P2.	retrieves SML & PCR, $S_y \Leftarrow Sig_{AIK_{sk}}(PCR\|\|N)$		
P3.	$\zeta \Leftarrow \mathrm{H}(ID_{SP})$, $N_v \Leftarrow \zeta^f$		
P4.	$\sigma \Leftarrow$ DAA-Signs (AIK_{pk})		
P5.	generates a, $y \Leftarrow g^a \bmod p$ $K \Leftarrow (g^b)^a \bmod p$		
P6.	constructs $SrvRep{:=}(AdvId, SrvInfo, S_y,$ SML, $AIK_{pk}, \sigma, t_p)$		
P7.	$E_x \Leftarrow E_{K_1}(SrvRep)$, $\mathrm{MAC}_{K_2}(E_x)$		
P8.		$\xrightarrow{E_x, \mathrm{MAC}_{K_2}(E_x), y}$	

Figure 7.3: Mutual Authentication Phase

7.5.3.3 Verification Phase

On receiving a service reply message from a platform, the service provider *SP* performs the following steps to verify its trustworthiness.

1. *SP* computes $K = (g^a)^b \bmod p$ and hence obtains K_1 and K_2 (where $K = K_1||K_2$). *SP* then checks the integrity of the received value of $E_{K_1}(SrvRep)$ by recomputing the MAC using K_2 and comparing it with the received value.

2. *SP* extracts *SrvRep* by decrypting $E_{K_1}(SrvRep)$ using K_1. *SP* also checks the validity of the timestamp t_p extracted from *SrvRep*.

3. *SP* verifies the DAA Signature σ, and is thus convinced that the platform is in possession of a legitimate DAA Certificate from a specific DAA Issuer, which implies that a genuine TPM is contained in the platform.

4. *SP* is also convinced that AIK_{pk} was signed using the platform's DAA Secret f. Even though the value of f is never revealed to *SP*, *SP* knows that the value is related to a genuine DAA Certificate.

5. *SP* checks that the nonce N is the same as that sent in *SrvAdv*.

6. *SP* verifies the trustworthiness of the platform by examining the platform integrity measurements. This involves recursively hashing the values contained in SML, and comparing them with the corresponding PCR values.

7. If *SP* is satisfied with the integrity measurements, then the platform (and hence the user) is authenticated to *SP*.

To authenticate to another service provider, the user platform should generate a new AIK key pair, but only needs to repeat the mutual authentication phase, i.e. it does not need to perform the join phase again. The user platform should also use a different N_v value.

7.6 Security Analysis

We now assess the Ninja scheme against the security and privacy requirements outlined in Section 7.2.3.

Mutual Authentication

Mutual authentication is achieved in the following way. A service provider is first authenticated to a prospective service user through a service advertisement message, protected using conventional cryptographic mechanisms (e.g. as supported by a PKI). If a prospective user is interested in the service, then the trustworthiness of the user platform is anonymously authenticated to the service provider via a service reply message using DAA.

The scheme is resistant to the following attacks.

- **Replay:** The use of the timestamps t_{sp} and t_p in the *SrvAdv* and *SrvRep* messages allows the recipients to check that they are sufficiently fresh. An adversary which knows an old session key K might be able to decrypt an old *SrvRep* message, and could try to use the corresponding old signature $Sig_{AIK_{sk}}(PCR||N)$ to reply to a new *SrvAdv* message. This will fail because the signature is computed as a function of the nonce N from *SrvAdv*, and a replayed signature will have been computed using a different value of N.

- **Man-in-the-Middle (MitM):** Since *SrvAdv* is authenticated, a MitM cannot masquerade as a service provider to a user. A MitM can make a response on its own behalf (as can anyone receiving *SrvAdv*). However, a MitM cannot masquerade as a legitimate user by manipulating the *SrvRep* message. If it tries to generate a *SrvRep* with a different Diffie-Hellman parameter y, then it can only generate a completely new response, since it cannot decrypt a *SrvRep* generated by another user. If it leaves y unchanged, then any modifications to *SrvRep* will be detected by the service provider, since it is integrity protected using a MAC computed as a function of the Diffie-Hellman key.

User Anonymity

The public part of the Endorsement Key, EK_{pk}, is never disclosed to a service provider, since it would function as a permanent identifier for the platform. Users instead interact with service providers using AIKs, which act as pseudonyms. Since it is computationally infeasible for service providers, or even the DAA Issuer, to link two AIK public keys from the same platform (see Section 4.4.2), users will remain anonymous to service providers (i.e. the case of curious service providers), as well as curious DAA Issuers (i.e. the case of curious trusted third parties) and passive eavesdroppers.

Service Information (SI) Confidentiality

A *SrvRep* message contains service information which, if disclosed, could reveal a user's service preferences and habits, thereby compromising user privacy. To prevent such a disclosure (e.g. to eavesdroppers or a malicious man-in-the middle), *SrvRep* is encrypted using a secret key known only to the service user and the service provider. Whilst there is nothing to prevent a malicious service provider from divulging the service information of an anonymous user, the confidentiality of the user's service information is still preserved, as the malicious service provider is unable to determine which service information corresponds to which user.

Unlinkability/Collusion Resistance

User platforms should interact with different service providers using different AIK public keys and N_v values. It is computationally infeasible for colluding service providers to link these keys (see Section 4.4.2), and as a result a user's service activities with different service providers are unlinkable. This remains true even if a DAA Issuer (i.e. a curious trusted third party) colludes with one or more service providers, again as discussed in Section 4.4.2. Our scheme is therefore resistant to two or more colluding service providers (i.e. the curious service providers case), as well as a DAA Issuer colluding with one or more service providers (i.e. the curious trusted third party case).

Transaction History

For business reasons (e.g. to support customer loyalty rewards or discounts), it may be necessary for service providers to link services originating from the same user. This can be achieved without compromising a user's privacy or anonymity if a service user always uses the same value of N_v to interact with a particular service provider. A service user will not need to store N_v, as it will be recovered during re-computation (since ζ and f should remain unchanged).

Blacklisting malicious parties

A detected rogue service provider can be added to the appropriate CRLs, enabling users to avoid known fraudulent service providers. Similarly, a service provider may want to blacklist a misbehaving or malicious user, in order to bar this user from future service interactions. This requires a means for the service provider to recognise a malicious platform, whilst it remains anonymous. This can be achieved by blacklisting platform pseudonyms, i.e. the N_v values of such platforms. Blacklisting the AIK will not work, as a rogue user can simply generate a new AIK, DAA-Sign it, and then interact with the service provider again.

A rogue user could only avoid detection by obtaining a new pseudonym, N_v. This would involve using a new value for f (the DAA secret). Although a TPM could generate a new f value, it is unlikely that it will be able to obtain a DAA Certificate for it. DAA certificate issue is expected to be subject to careful checks, and a platform is not expected to possess more than one DAA Certificate from a DAA Issuer. Also, if a DAA Certificate (i.e. a triple of values A,e,v) and the value f are found in the public domain (e.g. on the Internet), then they should be sent to all potential service providers for blacklisting. The service providers can then add them to privately maintained lists of rogue keys.

7.7 Summary

We identified security and privacy threats that may arise during service discovery in a ubiquitous computing environment; we also derived corresponding security and privacy requirements. We presented the Ninja mutual authentication scheme, that uses Trusted Computing functionality and which preserves user privacy. Apart from being communications-efficient (only two messages are required), the scheme also satisfies all the identified security requirements. To a service user and service provider, security and privacy are both desirable. However, they are potentially conflicting requirements, and it is challenging to achieve them both. However, this is achieved by the scheme presented here, enabling services to be discovered securely and privately.

The Ninja scheme is flexible and could easily be adapted and applied to other contexts and environments (e.g. Mobile Ad Hoc Networks or P2P Networks) to fulfil similar security requirements. Alternatively, if there are requirements for two way (mutual) anonymous authentication between two communicating entities, then the scheme could also be adapted to meet this need.

Chapter 8

A Framework for Secure Ubiquitous Service Delivery

Contents

In a mobile ubiquitous environment, service interactions between a user device and a service provider should be secure, regardless of the type of device used to access a service. In this chapter, we present the Secure Device Management Framework (SDMF), designed to securely deliver services to user devices, whilst also hiding (some of) the complexity of security management from users. Key to this framework is the Device Management Entity (DME), that manages a user device's security credentials, and interacts with service providers on its behalf. This framework also provides users with assurance that a compromised device cannot consume the delivered service, and, at the same time, prevents users from illegally sharing their *credentials with other users. We achieve these objectives using Trusted Computing functionality and certain other security mechanisms.*

8.1 Introduction

In a mobile ubiquitous environment (such as the one shown in Figure 1), users typically own disparate devices (e.g. PDAs, laptops, mobile phones, etc.) with varying form factors and computation capabilities, which are attached to a range of network access technologies and used to consume a variety of content and services offered by service providers. It is extremely important to secure the service interactions between user devices and service providers. However, achieving this presents many interesting challenges. For instance, a user may own many different types of device, and the type of device used to access a service may depend on the user's physical location or context. The user wishes to be able to access a service in a simple and secure manner using an appropriate device without being burdened with (too much) technical complexity. A service provider may be happy to allow user access to services provided that they have a legitimate subscription (i.e. using a properly issued credential) and that the user device is not in a compromised state. The latter task is further complicated by the current landscape, where security threats (e.g. viruses, malware, trojans, spam, etc.) abound. In this chapter, we aim to address these challenges collectively; to the best of our knowledge, there is no prior published work with precisely this focus.

In this chapter we propose the SDMF, a Device Management Framework for the secure delivery of ubiquitous services to end user devices. Apart from providing secure service interactions amongst the players, the framework is also designed to reduce the complexity of device security management tasks for users. Furthermore, the framework protects the interests of service providers by preventing unauthorised credential sharing amongst user devices. One other novel feature of the framework is that compromised devices are self-revoking, hence removing the need for a cumbersome revocation infrastructure. We achieved these objectives by incorporating Trusted Computing functionality into an entity known as the Device Management Entity (DME), and then integrating it with a number of other security mechanisms (such as the MANA protocol [75], and the Ninja service discovery protocol described in Chapter 7).

The remainder of this chapter is organised as follows. In Section 8.2 we identify the major challenges to providing secure service delivery. Section 8.3 gives an overview of the building blocks that we employ to develop the secure device management framework. In Section 8.4 we present our secure device management framework, and Section 8.5 analyses its security. Finally, in Section 8.6, we draw conclusions.

8.2 Service Delivery Security Issues

In this section we first identify the security threats that users and service providers may be exposed to during the delivery of services in a mobile ubiquitous environment. We then specify a corresponding set of security requirements designed to mitigate these threats.

8.2.1 Security Threats

The security threats can be summarised as follows:

1. **Spoofing.** Malicious entities may masquerade as legitimate service providers by broadcasting fraudulent service messages to users in an attempt to initiate a bogus service interaction. Similarly, malicious entities may masquerade as service users by replaying old service messages (captured earlier) to service providers.

2. **Tampering.** Service messages (e.g. service advertisements, registration messages, or reply messages) exchanged between users and service providers over wireless communication channels, may be overheard by a passive adversary, and modified or deleted by an active adversary. In addition, an adversary may tamper with the service or content information requested by a user whilst the data is in transit. Tampering includes sending (injecting) false service messages or malicious content to users.

3. **Information Disclosure.**

   (a) **During Communications:** Service messages may be overheard whilst in transit (the risk is particularly high when using a wireless communications channel). This may reveal sensitive information about the user (e.g. the type of service being requested, the service providers a user interacts with, the exact times or dates when services are being accessed, etc.). A user's privacy may thus be compromised. Similarly, an eavesdropper may attempt to intercept the transmitted service/content for his/her own consumption.

   (b) **Physical Loss of Device:** Sensitive personal information (such as contact information, emails, etc.) may be stored in user devices. The devices may also hold the security credentials that are used to access subscribed services. Thus, if a user device is lost or stolen, an adversary may be able to both obtain sensitive information about a user and gain unauthorised access to services.

4. **Malicious software attack on user devices.** A user device, although still in the physical possession of the user, may be infected with malicious software (e.g. viruses, trojans, etc.). Such software could perform a variety of attacks, including infecting a device, stealing a user's credentials or personal information, or encrypting user data on the device and then demanding payment for the release of the decryption key. A user might be oblivious to such an attack, as his/her device may still appear to be functioning "as normal".

5. **Unauthorised Credential Sharing/Distribution.** Having legitimately obtained the access tokens or security credentials necessary to access a service or content, a user could share the credentials (e.g. with friends or family) in an unauthorised way, possibly for financial gain.

8.2.2 Security Requirements

Building on the threat analysis in the previous section, we now give a corresponding set of security requirements designed to address the identified threats.

1. **Entity Authentication.** In general, if they use an insecure channel, communicating entities need to authenticate each other to prevent potential spoofing

attacks. First, users should authenticate themselves to their devices, e.g. using a PIN or password, to address the threat of information disclosure arising from physical loss of a device. Secondly, devices and service providers should be mutually authenticated to each other.

2. **Integrity Protection.** Any message exchanges (either service management messages or the actual service/content) between communicating entities should be integrity protected in order to address data manipulation attacks. Possible integrity protection mechanisms include MACs and digital signatures (see Sections 2.4.3 and 2.4.5).

3. **Service Confidentiality.** This requirement aims to address potential information disclosure threats. Messages (either service management messages or the service itself) exchanged between entities need to be protected against eavesdroppers and active attackers. Symmetric encryption algorithms using a shared secret key (as discussed in Section 2.4.1) can be used to provide this service.

4. **Device Integrity Assurance.** Preventing software attacks on a device is a particularly challenging problem. A viable alternative to prevention would be to detect whether a device has deviated from specific trustworthy software states to untrustworthy states. It would be even better if access to service could be prevented if the device has been compromised. This could be achieved using the attestation mechanisms provided by Trusted Computing technology.

5. **Credential Sharing Prevention.** Service and content providers could lose significant revenue if users share their access credentials in an unauthorised way, e.g. for monetary gain. Digital Rights Management (DRM) mechanisms can be employed to prevent unauthorised credential sharing.

8.3 Building Blocks

In this section we introduce the building blocks used to meet the security requirements and objectives identified in Section 8.2.2. We first introduce the Device Management Entity, followed by Manual Authentication (MANA) protocols, the Ninja Service Discovery Protocol, and, finally, Trusted Computing technology.

8.3.1 Device Management Entity (DME)

As discussed in Section 3.3.2, the Device Management Entity (DME) is a semi-distributed, logical construct designed to abstract the technical complexities faced by users (especially non-expert users) in managing (i.e. configuring and operating) their devices in a mobile ubiquitous environment. The DME acts as an interface between user devices and external entities. In this chapter we specify additional security functionality for the security register of a DME, in order to offer users a richer and more comprehensive set of security services.

8.3.2 Manual Authentication (MANA) Protocols

Manual entity authentication (MANA) protocols allow two devices to authenticate one another using a combination of an insecure wireless channel and manual data transfer. MANA protocols are particulary useful in situations where it is necessary to establish a security association (e.g. in the form of a shared key) between two devices in close physical proximity, while minimising the amount of user intervention (e.g. pressing buttons or reading data from a display). MANA protocols are resistant to Man-in-the-Middle (MitM) attacks on the wireless channel. The basic requirements for MANA protocols are that the devices must possess user interfaces capable of data input (e.g. buttons) and data output (e.g. a display). ISO has standardised four MANA protocols in ISO/IEC 9798-6 [75], designed for use with devices with differing interface capabilities. The properties of the four MANA protocols can be summarised as follows:

- Mechanisms using a short check value:
  - One device with simple input, and one device with simple output;
  - Both devices with simple input capabilities.
- Mechanisms using a Message Authentication Code (MAC):
  - Both devices with simple output capabilities;
  - One device with simple input, and one device with simple output.

There is a growing literature on such mechanisms [12, 81, 83, 93, 94, 141]; we choose to use the MANA protocols in our framework because they have been adopted by ISO, which implies that this set of protocols has been subjected to close scrutiny by the security community.

8.3.3 Ninja: A Secure Service Discovery Protocol

Service discovery is an important pre-cursor to the process of service delivery. As discussed in Chapter 7, many service discovery protocols have been proposed, although very few of them address security and privacy issues. It is imperative that service discovery is conducted in a secure and private manner in order to protect the security and privacy interests of the users and service providers.

The Ninja service discovery scheme described in Chapter 7 is used to achieve the objective of secure service discovery. The Ninja protocol offers a number of security features specifically designed for a mobile ubiquitous environment. It provides mutual authentication between user and service provider, preserves the privacy of users, is efficient (has a low communications overhead), and establishes a shared key between the user and service provider. It is therefore well suited to our requirements.

8.3.4 Trusted Computing

To achieve the security objectives set out in Section 8.2.2, the device management framework uses the Trusted Computing protected message exchange mechanism; namely the binding and the sealing functionalities. As described in Chapter 4, binding involves encrypting data with a non-migratable public key, i.e. a public key stored by the TPM and for which the private key will never leave the TPM. As a result, bound data is available only to the TPM. Sealing involves binding data to integrity metrics representing a particular platform state. The TPM will only make the data available to the platform if the current values in the PCRs are equal to those bound to the data, i.e. only if the platform is in the pre-specified state. Note that arbitrary data as well as keys can be bound or sealed to a TPM and to particular platform states.

8.4 A Framework for Secure Device Management

We now present the Secure Device Management Framework (SDMF), designed to secure the process of service provisioning to end user devices. At the heart of this framework is the Device Management Entity (DME), that manages a user device's security credentials on its behalf and interacts with service providers. This framework also provides users with assurance that a compromised device is unable to consume the delivered service, and, at the same time, prevents users from sharing their credentials with other users in unauthorised ways. We achieve these security objectives using Trusted Computing functionality.

We first introduce the entities in the SDMF framework. We then state the assumptions upon which this framework is based. Finally, we describe the operation of the framework.

8.4.1 The Entities

The entities participating in the Secure Device Management Framework are as follows:

- the **User**, often a human, who is the end consumer of a service or application;
- the **User Devices** (such as PDAs, laptops, smartphones) which are used by the user to access the available services;
- the **Device Management Entity** (**DME**) performs a number of functions (as discussed in section 3.3.2). In the framework, its main roles are to securely interact with service providers on behalf of user devices and to manage the security credentials of user devices;
- the **Service Provider** provides end users with a range of service offerings (e.g. music, videos, software applications, etc.).

8.4.2 Working Assumptions

The operation of the SDMF relies upon a number of assumptions:

- User devices are equipped with Trusted Computing functionality conforming to version 1.2 of the TCG TPM specifications (see Chapter 4).
- User devices have a (possibly very simple) interface capable of data input and data output.
- User devices and the DME have agreed on the MANA protocol to be used in the Initialisation phase.
- User devices and the DME have agreed on the cryptographic functions to be used in the Enrolment and Secure Service Re-Distribution phases, i.e. a hash function H, a symmetric encryption algorithm E, and a MAC function.
- The DME is trusted by both the user and service provider to perform its tasks correctly.
- The DME and the service providers have agreed on the cryptographic functions to be used during the Secure Service Delivery phase, i.e. a symmetric encryption algorithm E and a MAC function.
- The user has ownership control of the TPM in the device. The user therefore possesses an authorisation secret that can later be used to authenticate to the TPM.
- User devices and the DME have loosely synchronised clocks. This enables the user devices and the DME to check that a received message is sufficiently fresh.
- Prior to the enrolment phase, the TPM within the User Device has generated a pair of Attestation Identity Keys (AIK_{pk}, AIK_{sk}), and has obtained a Certificate $Cert_{AIK_{pk}}$ for the public key AIK_{pk} from a Privacy CA (a trusted computing specific trusted third party, as described in Section 4.4.1).

Finally note that our framework is independent of the underlying transport and network layer protocols, as it is purely an application layer solution.

8.4.3 The SDMF

The notation used in the description of the framework is summarised in Table 8.1.

Table 8.1: Notation

Notation	Description
U	The User
D	The User Device
M	The Device Management Entity (DME)
S	The Service Provider
$EnReq$	Enrolment Request Message
$EnRep$	Enrolment Reply Message
$EnMsg$	Enrolment Message
$EnAck$	Enrolment Acknowledgement Message
$DevID$	A Unique Device ID number
$SessID$	A Unique Session ID number
Srv	A Service
K_{Srv}	The Service Encryption Key
$SKEK$	Service Key Encrypting Key

As shown in figure 8.1, the SDMF involves five phases. We now describe the workings of each phase in greater detail.

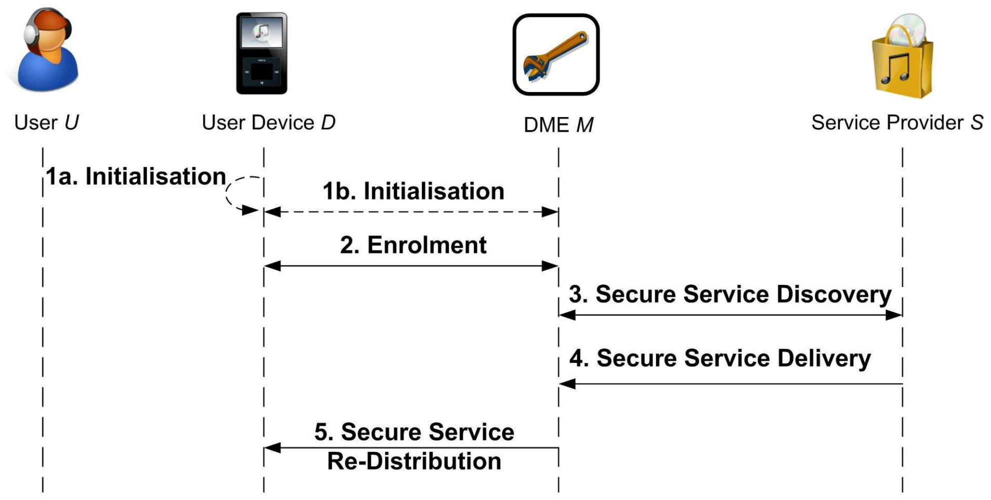


Figure 8.1: SDMF Operational Phases

8.4.3.1 Initialisation Phase

This phase involves the user preparing a device for the subsequent phases. One of the primary aims of this phase is for the user device and the DME to establish a shared key, which is subsequently used to secure communications. The user U, the user device D, and the DME M are required to be in close physical proximity. The user performs the following steps:

1. The user authenticates himself/herself to the TPM in the device D using the authorisation secret (obtained earlier when the user took ownership of the TPM). This enables the user to access the TPM functionality.
2. The user instructs D and the DME M to engage in one of the MANA protocols (as described in Section 8.3.2). The choice of protocol will depend on the types of interface (input and display) that the device possesses. The user is required to take part in the protocol and to perform his/her role in the protocol correctly (e.g. reading data from the display accurately and pressing the correct buttons). At the end of the protocol execution, D and M will have securely established a shared secret key K_{DM}. This is a long lived master key, used primarily to establish subsequent session keys, and should be stored securely when not in use.

The user will need to repeat this initialisation procedure for every device he/she owns (or intends to use for service consumption).

8.4.3.2 Enrolment Phase

In this phase, a user device D is securely enrolled with the DME M, which involves M being given D's security credentials. This phase may take place at any time after the Initialisation phase, and does not require the participating entities to be in close physical proximity. If necessary this phase can be performed at regular intervals, or even for every ‘session’, without repeating the Initialisation phase. It involves the following steps.

1. D constructs an Enrolment Request message, *EnReq*, which includes a Device Identifier *DevID*, a Session Identifier *SessID*, and a timestamp t_D:

   $$\mathit{EnReq} = (\mathit{DevID}, \mathit{SessID}, t_D).$$

   Note that *SessID* should be chosen such that it is unique during the lifetime of key K_{DM}. Using the shared key K_{DM} (established in the Initialisation Phase), D derives two session keys as follows: $K_1 = H(0||K_{DM}||SessID)$ and $K_2 = H(1||K_{DM}||SessID)$, where H is a cryptographic hash function. Note that K_1 and K_2 will be discarded at the end of the enrolment phase, and must not be reused. D then computes a MAC on *EnReq* using K_2, and sends the *EnReq* and the MAC to M:

   $$D \rightarrow M : (\mathit{EnReq}, \mathrm{MAC}_{K_2}(\mathit{EnReq})).$$

2. On receiving the above message, M retrieves *SessID*, and computes K_1 and K_2 in the same way as D. M checks t_D to ensure that *EnReq* is a fresh (or recent) message. M then verifies the integrity of the received message by recomputing the MAC using K_2, and (if the outcome is satisfactory) prepares an Enrolment Reply message *EnRep* for D:

   $$\mathit{EnRep} = (\mathit{ID}_M, \mathit{DevID}, \mathit{SessID}, N, t_M, \mathit{PCRInfo}),$$

   where N is a nonce chosen by M such that the probability of M ever using the same nonce twice during the lifetime of a key K_{DM} is negligible. That is, if the number of possible nonces is s, (and assuming that nonces are chosen uniformly at random from the set of all possible values) then the number of times that a key pair should be used should be significantly less than $s^{0.5}$. *PCRInfo* contains the indices of the PCR registers that M wishes to inspect.

   M computes a MAC on *EnRep* using K_2, and then sends *EnRep* and the MAC to D:

   $$M \rightarrow D : (\mathit{EnRep}, \mathrm{MAC}_{K_2}(\mathit{EnRep})).$$

3. On receiving the above message, D inspects t_M to ensure that the message is fresh. D then checks the integrity of the above message by recomputing the MAC and, if it verifies correctly, instructs its TPM to attest to its current software state. This involves D's TPM signing the concatenation of *PCR*,

made up of the PCR values requested by M, with the nonce N supplied by M:

$$Sig_{AIK_{sk}}(PCR||N).$$

Note that the nonce N is included to prevent possible replay attacks.

4. Using the `TPM_CreateWrapKey` command, the TPM in D generates a non-migratable Service Key Encrypting Key pair $(\mathit{SKEK}_{pk}, \mathit{SKEK}_{sk})$ which is bound to this TPM and sealed to D's current software state (i.e. the PCR values used in the previous step). The TPM then uses AIK_{pk} to sign a certificate $Cert_{\mathit{SKEK}_{pk}}$ for SKEK_{pk} using the `TPM_CertifyKey` command. $Cert_{\mathit{SKEK}_{pk}}$ contains a digest of SKEK_{pk}, and also describes the properties of the key, including the key type (bind, identity, signing, or storage), whether the key is migratable, and the PCRs to which the key is sealed. M will later use SKEK_{pk} to encrypt services destined for this device.

5. D constructs an Enrolment Message EnMsg as follows:

$$\begin{aligned}\mathit{EnMsg} = (&\mathit{DevID}, \mathit{SKEK}_{pk}, Cert_{\mathit{SKEK}_{pk}}, t_D, \mathit{SML},\\ &Sig_{AIK_{sk}}(PCR||N), \mathit{AIK}_{pk}, Cert_{AIK}),\end{aligned}$$

   where SML represents the set of values taken from D's Stored Measurement Log necessary to interpret the PCR values given in PCR.

   D then encrypts EnMsg using K_1, and computes a MAC on the encrypted EnMsg using K_2 to give:

$$E_{K_1}(\mathit{EnMsg}) \quad \text{and} \quad \mathrm{MAC}_{K_2}(E_{K_1}(\mathit{EnMsg})).$$

   D sends the following to M:

$$D \rightarrow M : (\mathit{DevID}, E_{K_1}(\mathit{EnMsg}), \mathrm{MAC}_{K_2}(E_{K_1}(\mathit{EnMsg}))).$$

6. On receiving the above message, M checks the integrity of the received message using K_2, and then decrypts the encrypted EnMsg using K_1. M next verifies the signature on the PCR values using AIK_{pk}, and assesses the "trustworthiness" of D by comparing the reported integrity metrics (i.e. the PCR values and the accompanying SML) against a list of known trustworthy states. M also checks that the nonce N is the same as that sent to D in step 2. If the device is deemed to be in a trustworthy state, M creates a new entry for this

device in its security register. The security register holds information associated with this device, i.e. $DevID$, K_{DM}, AIK_{pk}, and $SKEK_{pk}$. At this point, the user device D is successfully enrolled with the DME M.

7. M creates an Enrolment Acknowledgement message, $EnAck$, to inform D of the enrolment outcome $EnOut$ (i.e. Success or Failure):

   $$EnAck = (ID_M, DevID, EnOut, t_M).$$

   M computes a MAC on $EnAck$ using K_2, and sends the following to D:

   $$D \rightarrow M : (EnAck, \mathrm{MAC}_{K_2}(EnAck)).$$

It should be noted that, at any subsequent time, the user device D must be in the state indicated by the values of its PCRs in order to be able to access services via the DME.

8.4.3.3 Secure Service Discovery Phase

In this phase, the Ninja service discovery protocol (described in Chapter 7) is used to secure the process of service discovery between the DME (acting on behalf of the user devices) and a service provider. Prospective service providers advertise their service offerings via authenticated service advertisement messages. If a user is interested in a particular advertised service, the DME M (on behalf of the user devices) can then respond to the service advertisement with a service reply message. Service interactions between the DME and service provider are secured. At the end of the Ninja protocol run, a shared secret key K_{MS} has been established between the DME and the service provider. This shared key is used to secure the delivery of the sought services in the Secure Service Delivery phase, described immediately below.

8.4.3.4 Secure Service Delivery Phase

In this phase, the service provider securely delivers a requested service, Srv, to the DME.

1. The service provider generates a secret service encryption key, K_{Srv}, and uses it to encrypt the service:

$$E_{K_{Srv}}(Srv).$$

The service provider encrypts K_{Srv} using K_3 to obtain $E_{K_3}(K_{Srv})$, and computes a MAC on the encrypted service and the encrypted copy of K_{Srv} using K_4, where $K_{MS} = K_3||K_4$, to obtain:

$$\mathrm{MAC}_{K_4}(E_{K_{Srv}}(Srv)||E_{K_3}(K_{Srv})||t_S).$$

2. The service provider sends the following to the DME:

$$S \to M : (E_{K_{Srv}}(Srv), E_{K_3}(K_{Srv}), \mathrm{MAC}_{K_4}(E_{K_{Srv}}(Srv)||E_{K_3}(K_{Srv})), t_S).$$

3. On receiving the above message, the DME checks its integrity by recomputing the MAC using K_4, and then comparing it with the received value. If the two values agree, the DME proceeds to the next step.

4. The DME decrypts $E_{K_3}(K_{Srv})$ using K_3 to obtain K_{Srv}. Key K_{Srv} will be securely stored by the DME.

8.4.3.5 Secure Service Re-Distribution Phase

In this phase, the DME M securely redistributes the service (received from the service provider in the Secure Service Delivery phase) to a specific device D.

1. M consults its security register, retrieves the corresponding device public key $SKEK_{pk}$ (which was sealed to a trustworthy platform state), and re-encrypts K_{Srv} using $SKEK_{pk}$:

$$\bar{E}_{SKEK_{pk}}(K_{Srv}).$$

2. M chooses a new $SessID$ which is unique for the lifetime of key K_{DM}. M derives a new session key K_5 by hashing the concatenation of $DevID$, K_{DM} (from the Initialisation phase), and the newly generated $SessID$, i.e. $K_5 =$

$H(\mathit{DevID}||K_{DM}||\mathit{SessID})$. M then computes a MAC on the encrypted service and the re-encrypted service key using K_5 to obtain:

$$\mathrm{MAC}_{K_5}(E_{K_{Srv}}(\mathit{Srv})||\bar{E}_{\mathit{SKEK}_{pk}}(K_{Srv})).$$

3. M sends the secured service (i.e. the encrypted service) and the MAC to D:

$$\begin{aligned} M \to D : (&\mathit{SessID}, E_{K_{Srv}}(\mathit{Srv}), \bar{E}_{\mathit{SKEK}_{pk}}(K_{Srv}), \\ &\mathrm{MAC}_{K_5}(E_{K_{Srv}}(\mathit{Srv})||\bar{E}_{\mathit{SKEK}_{pk}}(K_{Srv}))). \end{aligned}$$

To consume the service, the user device performs the following steps:

1. On receiving the above message, the user device first computes K_5 and then uses it to check the integrity of the received message by recomputing the MAC.
2. If the outcome of the previous step is satisfactory, the user device unseals SKEK_{sk} using the `TPM_Unseal` command. Note that this step is only possible if the TPM in the device is in the same state as it was when the key was created and sealed.
3. If the previous step is successful, the user device decrypts $\bar{E}_{\mathit{SKEK}_{pk}}(K_{Srv})$ using SKEK_{sk} to obtain K_{Srv}.
4. The User Device decrypts $E_{K_{Srv}}(\mathit{Srv})$ using K_{Srv} to obtain the service Srv.

The service can now be consumed on device D.

8.5 Security Analysis

We now evaluate the security of the proposed framework against the security requirements identified in Section 8.2.2.

Entity Authentication

Entity authentication takes place between the following pairs of entities:

- User-to-Device: During the initialisation phase, the user is authenticated by the TPM in the device using an authorisation secret. Only the user (i.e. the authorised custodian of the TPM and the device) knows this secret. An attacker who has stolen the device would therefore be unable to access the TPM functionality required to unlock and access a service.
- Device-to-DME: During the enrolment and secure service re-distribution phases, the use of a shared secret key, known only to the device and DME (i.e. K_{DM} and the keys that are derived from it), to compute a MAC on the service messages, allows the device and the DME to mutually authenticate each other.
- DME-to-Service Provider: The DME and the service provider are mutually authenticated during the service discovery phase.

Timestamps and nonces are included in the service messages to provide freshness guarantees to prevent potential replay attacks.

Integrity Protection

All message exchanged during the various phases are integrity protected.

In the Enrolment and Secure Service Re-Distribution phases, the integrity of the *EnReq*, *EnRep*, *EnMsg*, and *EnAck* messages, the Service *Srv*, and the encrypted service key K_{Srv}, are protected with MACs that are computed using a shared key known only to the device and the DME (i.e. K_2 and K_5).

In the Secure Service Delivery phase, the Service *Srv* and the service encryption key K_{Srv} are integrity protected with MACs that are computed using a shared secret key known only to the DME and the service provider. Data origin authentication is also achieved for the service that is delivered to the DME by the service provider.

Service Confidentiality

During the Enrolment Phase, the Enrolment message *EnMsg* is encrypted before it is sent to the DME. This prevents eavesdroppers from learning its contents. In the Service Delivery and Service Re-distribution phases, the service, *Srv*, as well as the service encryption key, K_{Srv}, are encrypted whilst in transit. Eavesdroppers are thus unable to learn the types of service a user is consuming by observing the messages. An active attacker is also unable to capture the encrypted service and consume it.

Device Integrity Assurance

If a user device has been compromised (e.g. by malicious software), its software (as recorded in the PCR values) will differ from the expected value, and hence the device will be prevented from accessing a service, since it will now be unable to perform the UnSeal operation to retrieve the necessary secret key. If the device is unable to access a service, a user will be able to deduce that the device may have been compromised.

Since a compromised (or untrustworthy) device is "automatically" prevented from accessing a service, the requirement for a revocation infrastructure (for distributing and maintaining up to date CRLs) may no longer be necessary.

Unauthorised Credential Sharing Prevention

Depending on the type of service plans that a user subscribes to, service providers may, for example, allow a user to access a service on a maximum of (say) three of his own personal devices. These three devices are enrolled with the DME. As a service is encrypted with an service encryption key, and the service encryption key is encrypted using a key which is bound to the TPM in the device, a service can therefore be consumed only on the device to which the key is bound. The private part of the Service Key Encrypting Key $SKEK_{sk}$ never leaves the platform, and hence a user is unable to share it. Even if he is able to extract it, it may not be possible to unseal it in another device, as the PCR values will probably differ.

8.6 Summary

We have introduced SDMF, a framework for the secure delivery of ubiquitous services in a mobile environment. Using Trusted Computing and a combination of other security mechanisms, the framework supports the secure delivery of services to user devices, whilst removing the complexities of security management from the users. A user need not be worried if his/her device has been compromised. Service providers are also assured that unauthorised credential sharing is prevented. Our security analysis shows that the framework meets all the identified security objectives.

Chapter 9

Privacy Preserving Content Distribution Protection

Contents

A variety of Content Distribution Protection (CDP) schemes (e.g. Buyer-Seller Watermarking and Asymmetric Fingerprinting) have been proposed to address the problem of unauthorised distribution of copyrighted content. All the existing CDP schemes rely heavily on a Trusted Third Party to achieve the desired security objectives. In this chapter we present a Privacy Preserving CDP Watermarking Scheme *which uses trusted computing functionality and minimises the reliance on a Trusted Third Party. Our scheme, apart from being suited for a mobile environment, also allows a buyer to anonymously purchase digital content, whilst enabling the content provider to blacklist the buyers that are distributing content in unauthorised ways.*

9.1 Introduction

A mobile ubiquitous environment provides content providers with the ideal platform with which to reach a large pool of potential customers. This new distribution medium has the potential to dramatically increase a provider's revenue from content sales. On the other hand, the ease with which digital content can be delivered, duplicated, and distributed amongst end consumers also presents the digital content industry with a major problem, namely the unauthorised distribution of proprietary digital content. Hence, the challenge for content providers is how to prevent or deter unauthorised distribution of proprietary material, whilst embracing the business opportunity presented by digital distribution.

One technical means for deterring unauthorised content distribution is for the content provider to embed a unique watermark into every piece of content. If unauthorised copies of the content are found, the content provider can potentially use the watermark to trace the copy of the content back to the original buyer. This approach suffers from two problems. Firstly, an honest buyer could be wrongly accused (framed) of unauthorised distribution (e.g. if the content provider matches the wrong identity to the unauthorised copies of the content). Secondly, it may also be possible for a malicious buyer plausibly to claim that an unauthorised copy was in fact leaked by the content provider.

To address these problems, two types of content distribution protection (CDP) schemes have been proposed, with the goal of protecting the interests of both buyers and sellers. These are the *Buyer-Seller Watermarking* (BSW) schemes [98] and the *Asymmetric Fingerprinting* (AF) schemes [118]. These schemes require the inclusion of a buyer watermark in content, in addition to a watermark generated by the seller. Further, in order to preserve buyer privacy, a number of anonymous BSW and AF schemes [31, 38, 79, 85, 119] have subsequently been proposed.

In BSW schemes, a *Trusted Third Party* (TTP) typically has responsibility for generating buyer watermarks, while in AF schemes a buyer generates its own watermark, which is then proven to be well-formed to the content provider (using zero-knowledge proofs). In general, both of these approaches prevent an honest buyer from being framed, as well as a malicious buyer from denying that he/she has distributed proprietary content without authorisation. If buyer privacy is required, then a TTP can be employed to provide buyers with certified pseudonyms.

The requirement for an (online) trusted third party in existing BSW and AF schemes, either to generate the buyer watermarks or to provide pseudonyms for buyers, represents a major constraint, especially in a mobile ubiquitous environment where online trusted third parties are not necessarily readily available. Our goal is to remove this constraint, so that the resulting schemes are both inherently scalable and suitable for use in distributed and mobile environments. Trusted Computing technology offers a range of relevant security functionality, in particular a privacy preserving mechanism, which can be used to meet this objective.

Tomsich and Katzenbeisser [149] proposed a watermarking framework that uses tamper-proof hardware to protect proprietary content. Using Trusted Computing functionality, we take their approach a step further by proposing a concrete construction of a privacy preserving (i.e. anonymous) CDP scheme.

That is, in this chapter, we propose a privacy preserving, CDP watermarking scheme using two TC functionalities, namely the Direct Anonymous Attestation (DAA) protocol, and the Integrity Measurement, Storage and Reporting (IMSR) mechanism. By using DAA, our scheme minimises reliance on a TTP for privacy protection, as the buyer can generate verifiable pseudonyms on its own. As a result, we are able to reduce the communication overheads, and hence improve overall efficiency by comparison with existing BSW and AF schemes. A second contribution of our scheme is that, through the use of IMSR, the content provider is able to obtain assurance that a buyer-generated watermark is well-formed. The scheme also provides the following security features: framing resistance, user anonymity, content information confidentiality, unlinkability (even against the TTP), and transaction linkability. To the best of our knowledge, this is the first CDP scheme based on Trusted Computing functionality.

The remainder of this chapter is organised as follows. In Section 9.2, we provide an overview of digital watermarking technology, and discuss related work. Section 9.3 highlights the various CDP watermarking security issues. Section 9.4 describes the building blocks used to construct the novel scheme. In Section 9.5 we present the novel privacy preserving anonymous CDP watermarking scheme, and then analyse its security in Section 9.6. In the penultimate section, we compare our scheme against two other recent CDP watermarking schemes, and, finally, conclusions are drawn in Section 9.8.

9.2 Background and Related Work

In this section we briefly review digital watermarking technology. We then discuss related work.

9.2.1 Digital Watermarking Overview

The relative ease with which digital content can be duplicated and then distributed poses a major problem to the digital content industries. To address this problem, a variety of *Digital Watermarking* methods [41, 89, 92, 154] catering to different content types (e.g. audio, image, video) have been proposed. As defined by Memon and Wong [98]:

"*a watermark is a secret key dependent signal added to digital data (e.g. audio, video, or an image), which can later be extracted or detected to make an assertion about the data [98].*"

According to Cox et al. [41], an effective watermark should possess the following seven properties: *Unobtrusiveness*, *Robustness*, *Common Signal Processing*, *Common Geometric Distortion*, *Collusion and Forgery Attack Resistance*, *Universality*, and *Unambiguity* (the interested reader is referred to [41] for a detailed explanation of these properties). A digital watermarking method provides a means of embedding a watermark into a piece of digital content. In this thesis we are not concerned with the watermarking embedding techniques themselves; instead, we focus on secure watermarking protocols, i.e. the steps whereby buyers and sellers of digital copyrighted content interact with each other whilst trying to protect their individual interests. We simply assume that the underlying watermarking method (often referred to as the embedding operation) is secure.

The earliest digital watermarking protocols appeared in the late 1990s, notably in the work of Memon and Wong [97], and Voyatzis and Pitas [155]. Traditionally, the watermarking process is only carried out by the content seller, i.e. only the seller is responsible for inserting the watermark into the content. Although this protects the interest of the content seller by allowing the content seller to trace unauthorised copies of content found in the public space, it does not protect the interests of the buyer. This is because a buyer has no control over what is done to the watermarked content, and, as was noted by Qiao and Nahrestedt [125], the buyer is susceptible to a framing attack. That is, an honest buyer could be falsely accused by an dishonest content seller of illegally duplicating or distributing content. To overcome this problem, Qiao and Nahrestedt [125], and Memon and Wong [98] proposed watermarking protocols which involve the contribution of a watermark from a buyer, in addition to a seller's watermark. Such protocols are commonly referred to as Buyer-Seller Watermarking protocols.

Further, in order to preserve the privacy of a buyer, a variety of anonymous or privacy preserving buyer-seller watermarking protocols have subsequently been proposed. We observe that, in order for these protocols to achieve the desired security objectives, and to protect the interests of both the buyers and sellers, one or more online trusted third parties are required as active participants. Since one of our aims is to design a protocol for use in mobile environments, such a requirement for an online trusted third party may not be realistic. In the next section we briefly review existing work, and, in particular, we examine the role that a trusted third party plays in each of the protocols considered.

9.2.2 Related Work

Memon and Wong [98] proposed the first buyer-seller watermarking protocol in 2001. This scheme is the basis upon which many other schemes have subsequently been built.

Ju et al. [79], were one of the first, in 2002, to recognise the importance of preserving a buyer's privacy during a content purchase transaction. In their scheme, a buyer is able to purchase content anonymously. A buyer's transactions are also unlinkable. Three trusted third parties are involved. Firstly, it is assumed that all participants have a pair of public and private key pairs certified by a Certification Authority prior to the protocol run. Secondly, a Watermark Certification Authority (WCA) is also involved in generating the buyer's watermark on his/her behalf, and certifying his/her anonymous credential. A judge (i.e. another trusted third party) is also required to arbitrate in the event of a dispute. Shortly after the publication of [79], Choi, Sakurai, and Park [38] discovered that the scheme of Ju et al. remains susceptible to the framing problem if the content seller colludes with the WCA and/or the judge. They also proposed a new protocol which requires only one trusted third party, called a WCA, that has the role of generating a buyer's watermark as well as providing the buyer with the anonymous credential. Not long after the publication of [38], both of the above mentioned schemes were cryptanalysed by Goi et al. [60], revealing serious security issues which could potentially compromise user privacy.

Independently, Lei et al. [85] proposed an anonymous scheme which binds an instance of a content purchase transaction to the watermarking protocol. They argued that this is a more efficient process. This scheme involves three trusted third parties. Firstly, a CA is responsible for issuing anonymous certificates to buyers, and, secondly, a WCA is responsible for the generation of buyer's watermarks. Finally an arbiter is necessary to resolve disputes in the event of an infringement of the rights of content owners.

More recently, Zhang, Kou and Fan [163], proposed a scheme which involves only one trusted third party, i.e. a CA responsible for issuing anonymous credentials to buyers. A buyer's watermark is generated by the buyer without the assistance of a watermark authority. We will show later (in Section 9.7) that this scheme is not secure, as a buyer can easily remove his portion of the watermark.

In summary, all of the above mentioned schemes, apart from possessing security issues, are neither scalable nor suitable for use in environments where a online Trusted third party is not guaranteed to be available.

9.3 CDP Security Issues

In this section we examine the security and privacy issues associated with digital content distribution in a mobile ubiquitous environment.

9.3.1 CDP Threat Model

The potential security and privacy threats that may be posed to content buyers and content providers are as follows.

1. **Unauthorised Content Distribution.** A malicious user could without authorisation distribute proprietary content (which may have earlier been legally purchased from a content provider), resulting in the content being used by others without the appropriate payment being made to the content provider. This translates to a potential loss of revenue for the content provider.
2. **Framing.** To deter unauthorised content distribution, the content provider could employ a digital watermarking scheme to embed a unique seller-generated watermark into every piece of content bought by the buyer. Such a scheme, however, does not prevent an honest buyer from being falsely accused (framed) of unauthorised content distribution. This is a problem if there is no way for the buyer to challenge the decision and prove his/her innocence.
3. **Information Disclosure.**
    - **Buyer's Personally Identifiable Information:** During the process of content purchase, a buyer's PII, such as his/her identity or physical location, may be revealed (either willingly or unwillingly) to a content provider or passive eavesdropper.
    - **Content Information:** By observing the type of content that a buyer purchases, a passive adversary may gradually build up a profile of the buyer. This information may later be used to predict future patterns

and habits of the buyer. The privacy of the buyer could potentially be compromised as a result.

4. **Conspiracy Attacks.** Colluding content providers may buy, sell or exchange information about their customers. Such collusion could not only provide content providers with monetary benefits, but also enhance their business intelligence by building a more comprehensive profile of their clients. With the aid of a TTP, buyers can employ privacy enhancing mechanisms to protect their identity when they interact with content providers. The consequences for buyer privacy could be even more serious if a TTP decides to collude with content providers.

9.3.2 CDP Security Requirements

We now give a set of security requirements designed to address the identified threats.

1. **Traceability.** The most fundamental requirement for any content protection or watermarking scheme is the ability for a content seller to determine (via some means) that content it has sold is being duplicated and/or distributed in an unauthorised way.

2. **Framing Resistance.** It should not be possible for the content provider to falsely accuse an honest buyer of unauthorised content distribution.

3. **Buyer Anonymity.** Unique identifying information for a buyer (such as a long lived key) should not be divulged to a content provider during the content purchasing process. If required, a buyer should be able to interact with content providers using pseudonyms.

4. **Content Information Confidentiality.** Eavesdroppers (listening to the communications between a buyer and content provider) should not be able to determine the type of content that is being purchased by the buyer.

5. **Conspiracy Attack/Collusion Resistance.** Colluding content providers should not be able to link the activities of the same buyer. Similarly, when a TTP colludes with a content provider, they should not be able to correlate

the actions of a particular buyer. In other words, it should be impossible for colluding content providers to tell if two sets of prior content purchase transactions (made with different providers) had originated from the same or different buyers.

6. **Transaction History.** For billing or other purposes (e.g. loyalty rewards), it may be necessary for a content provider to maintain the transaction histories of its customers. That is, a content provider may need to be able to identify whether a particular client is a repeat customer (and, if so, which one) or a first time buyer, whilst still being unable to determine his/her unique identity.
7. **Blacklisting of Rogue Buyers.** In the event that unauthorised copies of proprietary content are found (e.g. on the Internet), content providers should be able to blacklist the buyers of these copies of the content.

9.4 Building Blocks

In this section we briefly introduce and describe the building blocks used to develop the privacy preserving CDP watermarking scheme described later in the chapter.

9.4.1 Homomorphic Encryption Functions

We first provide a brief description of a homomorphic encryption function. Let $\mathcal{E}_{\mathcal{K}} : \mathcal{M} \rightarrow \mathcal{C}$ be an encryption function, where $\mathcal{M}$ is the set of plaintexts, $\mathcal{C}$ is the set of ciphertexts, and $\mathcal{K}$ is the set of keys. Then $\mathcal{E}$ is said to be homomorphic if, for every $m_1, m_2 \in \mathcal{M}$, and $k \in \mathcal{K}$:

$$\mathcal{E}_k(m_1) \odot_{\mathcal{C}} \mathcal{E}_k(m_2) = \mathcal{E}_k(m_1 \odot_{\mathcal{M}} m_2),$$

where $\odot_C$ is an operator on $\mathcal{C}$, and $\odot_M$ is an operator on $\mathcal{M}$. We usually consider operators to be either additive or multiplicative. We can therefore regard an encryption scheme as additively homomorphic if the operator is additive, and a encryption scheme as multiplicatively homomorphic if the operator is multiplicative.

Examples of homomorphic encryption schemes include the schemes of Rivest, Shamir and Adelman [127], ElGamal [49], Goldwasser and Micali [62], Benaloh [23], and Pallier [111]. The interested reader is referred to Fontaine and Galand's survey on homomorphic encryption [50].

9.4.2 Permutation Function

According to Memon and Wong [98], a Watermark W can be considered as a sequence of individual elements, i.e.:

$$W = (w_1, w_2, w_3, ..., w_n).$$

If the watermark W is encrypted using a key K, it simply means that the individual elements are encrypted, i.e.:

$$E_K(W) = (E_K(w_1), E_K(w_2), E_K(w_3), ..., E_K(w_n)).$$

A random permutation function ρ can be used to randomise an encrypted watermark $E_K(W)$. This has the effect of randomising the order of the individual encrypted elements of the watermark, i.e.:

$$\rho(E_K(W)) = E_K(\rho(W))$$

Thus the permutation and encryption processes commute.

9.4.3 Trusted Computing

To achieve the security objectives set out in Section 9.3.2, the CDP watermarking scheme utilises two Trusted Computing functionalities, namely the Integrity, Storage, and Reporting (IMSR) mechanism, and the Direct Anonymous Attestation (DAA) protocol. IMSR is employed to provide a content seller with assurance that a reliable watermarking method was used to generate the buyer watermark. DAA is used to provide privacy and anonymity properties to a content buyer. A detailed description of these two functionalities can be found in Chapter 4.

9.5 A Novel CDP Scheme

In this section, we present a novel privacy-preserving content distribution protection watermarking scheme. The primary objective of the scheme is to allow the buyer to anonymously purchase digital content, whilst allowing a seller to blacklist any buyer platforms that are distributing content in unauthorised ways. Using the aforementioned Trusted Computing functionalities, the scheme also allows a buyer to generate verifiable pseudonyms and to convince a content provider that the buyer generated watermark is well-formed, without the involvement of an (online) TTP. The scheme is also designed to meet all the security requirements set out in section 9.3.2.

We first introduce the entities participating in the protocol. Next, we state the assumptions upon which the scheme is based. Finally, we describe the operation of the scheme.

9.5.1 The Entities

The entities participating in the scheme are as follows:

- the **Content Buyer** of digital content (e.g. music, video, podcasts, etc.);
- the **Platform** (or device), which consists of the TPM and its host; the platform is also the device which a buyer will use to interact with other entities;
- the **Content Seller** (also referred to as the content provider) of digital content;
- the **DAA Issuer**, which is also the authority that issues DAA Certificates to legitimate platforms.

9.5.2 Working Assumptions

The proper operation of the scheme relies upon the following assumptions:

- The buyer is already authenticated to the platform (via some out-of-band mechanism such as the one given in [59]) that is used for the CDP scheme. We refer to the combination of the buyer and the platform as the *Buyer Platform (BP).*

- The buyer and seller are already authenticated to each other prior to content distribution, for example as part of an authenticated service discovery process (such as that described in Chapter 7).
- The device/platform running the CDP scheme is equipped with TCG functionality conforming to v1.2 of the TCG specifications (as described in Chapter 4).
- Content buyers and content sellers have loosely synchronised clocks (e.g. within an hour of each other). This enables content sellers to check that a content request message is sufficiently fresh.
- Buyers only obtain DAA Certificates from trustworthy DAA Issuers (i.e. those that use the same public key for a very large set of users in order to avoid the Rudolph attack discussed in Chapter 5).
- The parties involved have agreed on the cryptographic functions to be used as part of the protocol. These are as follows.
  - A hash-function H.
  - A homomorphic encryption algorithm $\mathcal{E}$, where $\mathcal{E}_K(X)$ is used to denote the homomorphic encryption of data X using the secret key K.
  - A digital signature algorithm, where $Sig_K(X)$ is used to denote a signature computed on data X using the private signature key K.
- The watermark embedding operation $\otimes$ is public knowledge, and the security of this embedding relies only on the secrecy of the key used to embed the watermark W. (In our case this key is a secret random permutation ρ). In addition, the watermark W embedded using $\otimes$ is *collusion resistant*, which means that it is computationally infeasible for attackers to remove W or cause W to be undetectable even if they have access to multiple copies of the content containing different watermarks.

9.5.3 The Scheme

Table 9.1 gives the notation used in describing the scheme.

Table 9.1: Notation

Notation	Description
BP	The Buyer Platform
S	The Seller or Content Provider
I	The DAA Issuer
f	A DAA secret value generated by the TPM
v', v'', e	DAA parameters (described in Section 3.2)
σ	A DAA Signature
ρ	A random permutation function
X'	Watermarked Content
$X \otimes W$	Embedding of W into X using the embedding operation, $\otimes$
(AIK_{pk}, AIK_{sk})	A pair of public/private Attestation Identity Keys
(EK_{pk}, EK_{sk})	The pair of public/private Endorsement Keys

As shown in figure 9.1, the proposed CDP scheme involves three distinct phases, namely the *Buyer Registration Phase*, the *Watermarking Phase*, and the *Content Acquisition Phase*. We now describe the workings of each phase in greater detail.

9.5.3.1 Buyer Registration Phase

The objective of the *Buyer Registration Phase* is to enable a buyer platform to obtain a *DAA Certificate* from a *DAA Issuer*. Note that the Buyer Registration Phase of this scheme is identical to the DAA Join Protocol (as described in Chapter 4), and may have taken place before the device is shipped to the content buyer. This is a typically a one-off process. We now describe the sequence of events (also depicted in Figure 9.2)

1. The TPM in the buyer platform generates its DAA Secret value f and a random value v'. It computes U and N_I (as described in Section 4.4.2), and then sends U, N_I, and its Endorsement Public Key, EK_{pk} to the DAA Issuer.

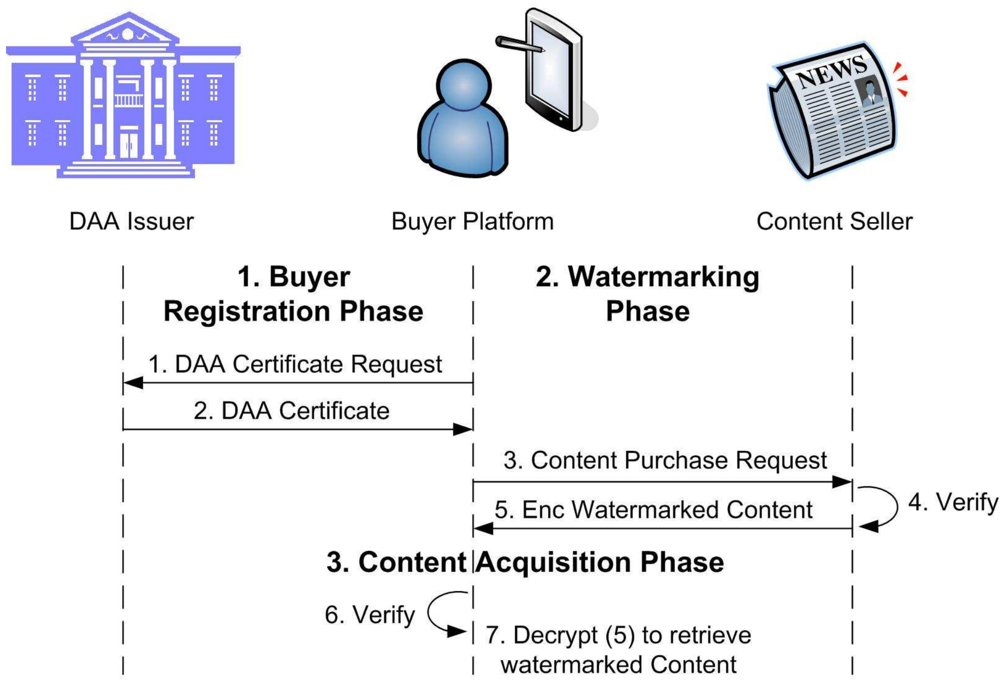

DAA Issuer
Buyer Platform
Content Seller
NEWS
1. Buyer Registration Phase
2. Watermarking Phase
1. DAA Certificate Request
2. DAA Certificate
3. Content Purchase Request
4. Verify
5. Enc Watermarked Content
3. Content Acquisition Phase
6. Verify
7. Decrypt (5) to retrieve watermarked Content

Figure 9.1: The CDP Watermarking Scheme Phases of Operation

2. To verify that U originates from the TPM in the platform that owns EK_{pk}, the DAA Issuer engages in a simple *Challenge-Response* protocol with the platform. It generates a random message m, and encrypts m using EK_{pk}. It sends the challenge, $Chl = \bar{E}_{EK_{pk}}(m)$ to the platform.

3. If the platform owns EK_{pk}, it should have the corresponding EK_{sk} and hence is able to decrypt $\bar{E}_{EK_{pk}}(m)$ to retrieve m. The platform (TPM) then computes and sends the response $r = H(U||m)$ to the DAA Issuer.

4. The DAA Issuer computes $H(U||m)$ using the value of m it sent in step 2, and compares the result with the value of r received from the platform. If the two values agree, then the DAA Issuer is convinced that U originated from the TPM that owns EK_{pk}.

5. Finally, the DAA Issuer generates v'' and e, and then computes A (as described in Section 4.4.2). The DAA Issuer then sends (A, e, v'') to the platform.

6. The DAA Certificate is (A, e, v), where $v = v' + v''$. The DAA Issuer does not have full knowledge of the DAA Certificate, since the certificate was jointly created by the platform and the DAA Issuer. This property helps preserve the anonymity of the user/platform.

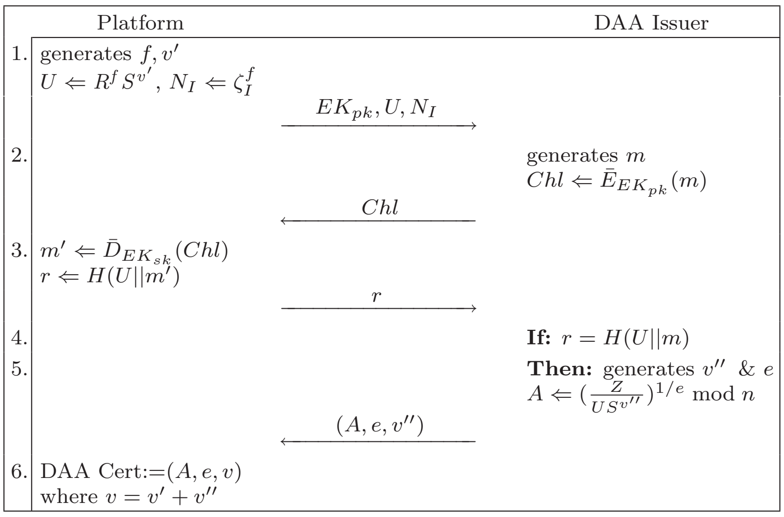

Platform
DAA Issuer
1. generates f, v'
$U \Leftarrow R^f S^{v'}$, $N_I \Leftarrow \zeta_I^f$
EK_{pk}, U, N_I
2. generates m
$Chl \Leftarrow \bar{E}_{EK_{pk}}(m)$
Chl
3. $m' \Leftarrow \bar{D}_{EK_{sk}}(Chl)$
$r \Leftarrow H(U||m')$
r
4. **If:** $r = H(U||m)$
5. **Then:** generates v'' & e
$A \Leftarrow (\frac{Z}{US^{v''}})^{1/e} \bmod n$
(A, e, v'')
6. DAA Cert:=(A, e, v)
where $v = v' + v''$

Figure 9.2: Buyer Registration Phase

9.5.3.2 Watermarking Phase

The purpose of this phase is to enable a buyer to contribute a watermark (fingerprint), and for the seller to embed this buyer watermark into the proprietary content before it is distributed to the buyer. The embedding is performed in such a way that the watermark embedded into the content is 'invisible' to both the buyer and seller, i.e. neither the buyer nor the seller is able to determine what the watermark is. The entities involved in this phase are the *Buyer Platform B* and the *Seller S*. The sequence of events is as follows (also depicted in Fig. 9.3):

B1. B generates a watermark W using the generation function of a reliable watermarking algorithm (e.g. the spread spectrum watermarking algorithm given in [41]).

B2. B generates an encryption key pair (BEK_{pk}, BEK_{sk}) for the chosen homomorphic encryption scheme, and then encrypts the watermark W using BEK_{pk}, to create:

$$\mathcal{E}_{BEK_{pk}}(W).$$

B3. The TPM in B generates a non-migratable signing key pair (BSK_{pk}, BSK_{sk}), and then uses the private key to sign the encrypted watermark $\mathcal{E}_{BEK_{pk}}(W)$

(from step B2), and BEK_{pk}, to obtain:

$$Sig_{BSK_{sk}}(\mathcal{E}_{BEK_{pk}}(W), BEK_{pk}).$$

B4. The TPM in B generates an AIK key pair (AIK_{pk},AIK_{sk}).

B5. B retrieves the Stored Measurement Log (SML), and the corresponding Platform Configuration Register (PCR) values from the TPM (denoted by PCR). The TPM in B then signs the PCR values and a newly generated timestamp t_b using AIK_{sk} (from step B4):

$$Sig_{AIK_{sk}}(PCR||t_B).$$

The SML and PCR values provide evidence that a particular watermarking algorithm was used (by the buyer B) to generate the watermark (in step 1).

B6. The TPM in B computes $\zeta = \mathrm{H}(ID_S)$. It then creates a pseudonym $N_v = \zeta^f$ (where f is the DAA Secret generated during the Buyer Registration Phase) for use when interacting with the Seller.

B7. To prove to the Seller that the AIK (from step B4) originates from a genuine TPM, the TPM in B now DAA-Signs AIK_{pk} using f, the *DAA Certificate*, and the other public parameters of the system. The output of the DAA-sign operation is the DAA Signature, σ (which also includes ζ and N_v).

B8. To prove that BSK originates from the TPM, the TPM signs BSK_{pk} and the timestamp t_B using AIK_{sk}, i.e. it computes:

$$Sig_{AIK_{sk}}(BSK_{pk}||t_B).$$

B9. B sends the following to the Seller S:

$$\begin{aligned} B \to S : & \mathcal{E}_{BEK_{pk}}(W), AIK_{pk}, BSK_{pk}, BEK_{pk}, \sigma, t_B, SML, Sig_{AIK_{sk}}(PCR||t_B), \\ & Sig_{BSK_{sk}}(\mathcal{E}_{BEK_{pk}}(W), BEK_{pk}), Sig_{AIK_{sk}}(BSK_{pk}||t_B). \end{aligned}$$

On receiving this message from the buyer, and to incorporate the buyer's watermark into a piece of content, the seller S performs the following steps:

S1. Checks to see if the timestamp t_B is valid (i.e. successfully close to its current clock value), and verifies the DAA Signature σ; on successful verification, S is convinced that:

(i) B is in possession of a legitimate DAA Certificate from a specific DAA Issuer, which implies that a genuine TPM is held by B.

(ii) AIK_{pk} was signed using the DAA Secret f contained in B's TPM. Even though the value of f is never revealed to the seller, S knows that the value is related to the one in the DAA Certificate.

S2. Examines the integrity metrics of the buyer platform. This is achieved by recursively hashing the values indicated in the SML, and then comparing them with the corresponding PCR values. If the outcome is satisfactory, and if the software state indicated by the SML is a state that S trusts, S is convinced that a reliable watermarking algorithm was used by the buyer platform to generate its buyer watermark W, and that the platform is in a trustworthy state.

S3. Verifies $Sig_{AIK_{sk}}(BSK_{pk}||t_B)$ and $Sig_{BSK_{sk}}(\mathcal{E}_{BEK_{pk}}(W), BEK_{pk})$.

S4. Generates a seller watermark V, and then embeds it into the Content X, to obtain:

$$X' = X \otimes V.$$

S5. Encrypts X' (from step 4) using BEK_{pk} to obtain:

$$\mathcal{E}_{BEK_{pk}}(X').$$

S6. Permutes $\mathcal{E}_{BEK_{pk}}(W)$ (received from buyer in step B9) with a secret random permutation ρ to get $\mathcal{E}_{BEK_{pk}}(\rho W)$ by computing:

$$\rho(\mathcal{E}_{BEK_{pk}}(W))= \mathcal{E}_{BEK_{pk}}(\rho W)).$$

S7. The encrypted permuted watermark (from step S6) is then embedded (while still in the encrypted domain) into the encrypted watermarked content X' (from step S5) by computing:

$$\mathcal{E}_{BEK_{pk}}(X' \otimes \rho W) = \mathcal{E}_{BEK_{pk}}(X') \otimes \mathcal{E}_{BEK_{pk}}(\rho W).$$

This follows from the homomorphic property of the encryption algorithm.

S8. The encrypted, dual watermarked content (from step S7) is then distributed to the buyer platform B.

$$S \to B : \mathcal{E}_{BEK_{pk}}(X' \otimes \rho W).$$

	Buyer Platform B		Content Seller S
B1.	generates W.		
B2.	generates BEK_{pk}, BEK_{sk}, $\mathcal{E}_{b1} \Leftarrow \mathcal{E}_{BEK_{pk}}(W)$.		
B3.	generates BSK_{pk}, BSK_{sk}, $S_{b1} \Leftarrow Sig_{BSK_{sk}}(\mathcal{E}_{b1}, BEK_{pk})$.		
B4.	generates AIK_{pk}, AIK_{sk}.		
B5.	retrieves SML & PCR, $S_{b2} \Leftarrow Sig_{AIK_{sk}}(PCR \| t_B)$.		
B6.	$\zeta \Leftarrow \text{H}(ID_S)$, $N_v \Leftarrow \zeta^f$.		
B7.	$\sigma \Leftarrow$ DAA-Signs (AIK_{pk}).		
B8.	$S_{b3} \Leftarrow Sig_{AIK_{sk}}(BSK_{pk} \| t_B)$.		
B9.	constructs $M_b := (\mathcal{E}_{b1}, AIK_{pk}, BSK_{pk}, \sigma$ $BEK_{pk}, SML, t_B, S_{b1}, S_{b2}, S_{b3})$.		
		$\xrightarrow{M_b}$	
S1.			verifies σ.
S2.			checks PCR and SML values.
S3.			verifies S_{b1} & S_{b3}.
S4.			generates V, then $X' \Leftarrow X \otimes V$.
S5.			$\mathcal{E}_{s1} \Leftarrow \mathcal{E}_{BEK_{pk}}(X')$.
S6.			permutes $\mathcal{E}_{b1}$ to get $E_{BEK_{pk}}(\rho W)$.
S7.			computes $E_{BEK_{pk}}(X' \otimes \rho W)$.
S8.		$\xleftarrow{E_{BEK_{pk}}(X' \otimes \rho W)}$	
1.	decrypts $E_{BEK_{pk}}(X' \otimes \rho W)$, to retrieve $(X' \otimes \rho W)$.		

Figure 9.3: Watermarking and Content Acquisition Phases

9.5.3.3 Content Acquisition Phase

When the buyer platform B receives the encrypted, watermarked content $\mathcal{E}_{BEK_{pk}}(X' \otimes \rho W)$, it decrypts it using BEK_{sk} to retrieve the watermarked content:

$$X'' = D_{BEK_{sk}}(\mathcal{E}_{BEK_{pk}}(X' \otimes \rho(W))) = X' \otimes \rho W.$$

The (final) watermarked content X'' is now ready for buyer consumption (e.g. viewing or listening).

9.5.3.4 Summary

In summary, and with reference to Figure 9.3:

- steps B1–B3, and S4–S8, collectively allow both the seller's watermark V and a secret function $\rho(W)$ of the buyer's watermark W to be embedded into the content X;
- steps B4–B8 allow the TPM and its host B to use the IMSR functionality to attest to the watermark generation process on the platform, and, using DAA, they also allow B to create a certifiable pseudonym (i.e. the AIK) for use when interacting with the Seller;
- Steps S1–S3 allow the seller to verify both the integrity of the buyer's platform and the authenticity of the buyer's pubic encryption key pair BEK_{pk}.

9.6 Security Analysis

We now discuss how the proposed scheme meets the security requirements outlined in Section 9.3.2.

Framing Resistance

It is not possible for a malicious content provider to falsely accuse an honest buyer of unauthorised content distribution, since neither the buyer nor seller know the final effective watermark $\rho(W)$ that is embedded into the distributed content X''. This is because a seller knows ρ and not W, while the buyer knows W and not ρ.

Buyer Anonymity

The scheme preserves buyer privacy by allowing the buyer to interact anonymously with content providers through the use of certified pseudonyms. Anonymity is inherently provided by the DAA functionality of the buyer's TPM. The Endorsement Key (EK), which is also the long-lived and unique identity of the TPM in the buyer platform, is never disclosed to the seller during content purchase or distribution. Buyers interact with sellers using AIKs and N_v values, which act as pseudonyms. Since it is computationally infeasible for sellers to link a specific EK and an AIK

from the same platform, buyers will remain anonymous to sellers. Similarly, since the DAA Secret f is never revealed outside the TPM, a seller with only N_v is not able to determine f.

Collusion Resistance

Unlinkability is another feature provided by the DAA functionality. To prevent collusion attacks amongst malicious content sellers and curious Trusted Third Parties, buyers should interact with different sellers using different AIKs and N_v values. Since, it is computationally infeasible for colluding sellers to link these keys and values to each other, a buyer's content purchasing activities with different sellers remain unlinkable.

As a DAA Issuer knows which TPMs possess valid DAA Certificates, it could collude with a seller in an attempt to link EKs with AIKs. To make such a link, the DAA Issuer would require knowledge of the TPM's DAA Secret value, f. Again this is computationally infeasible because of the way in which a DAA Certificate is created, and since f never leaves the TPM.

The scheme is therefore resistant to colluding sellers as well as sellers colluding with a DAA Issuer (the trusted third party).

Rogue Blacklisting

Seller security is provided by the ability to perform rogue blacklisting. This feature is offered by the DAA functionality of the TPM. A seller may blacklist malicious content buyers (i.e. those found to be distributing content illegally), to prevent them from purchasing content in future. In other words, if a malicious buyer revisits the seller, it should be possible for the seller to recognise that this buyer platform is malicious, whilst remaining anonymous. This can be achieved by blacklisting the pseudonyms, i.e. the N_v values of all known platforms of rogue buyers. The only way in which a rogue buyer could avoid detection would be to obtain a new pseudonym N_v. This would require the buyer to have a new value for f. Although it is possible for a TPM to generate a new value for f, it is unlikely that the buyer platform will be able to obtain a new DAA Certificate for this value from a DAA Issuer.

Furthermore, if a DAA Certificate and the value f are found in the public domain (e.g. on the Internet), then they should be distributed to all potential sellers, who should add them to their lists of rogue keys. This rogue platform identification method has the advantage of eliminating the need for a centralised revocation authority.

Transaction History

This is an additional usability feature that is not explicitly treated in previous CDP schemes, and is made possible by the variable anonymity feature available in DAA. That is, it may be necessary for sellers to link a repeat content buyer (e.g. for customer loyalty rewards or discounts). This can be achieved without any compromise of a buyer's privacy or anonymity if a content buyer uses the same N_v value to interact with a particular seller. Note that it is not necessary for a content buyer to store the value N_v, as the same value will be recovered during re-computation (since the values of ζ and f should remain unchanged).

Content Information Confidentiality

Content is encrypted with the buyer's public encryption key, BEK_{pk}. Content confidentiality is thus protected from eavesdroppers whilst in transit (e.g. whilst being distributed to the buyer over public networks). This also protects the privacy of a buyer, as malicious or curious entities are unable to determine the type of content that is being consumed.

9.7 Comparison with Related Work

In this section we compare the novel scheme with two other recently proposed CDP schemes, i.e. the schemes of Lei et al. [85], and Zhang, Kou and Fan [163]. To the best of our knowledge, there are no known attacks on these two schemes in the open literature. Note that the schemes of Choi, Sakurai and Park [38], and Ju et al [79], are not included in the comparison, as they have been broken by Goi et al. [60].

Our scheme adopts a similar approach to that adopted by Lei et al. [85], and Zhang, Kou and Fan [163], in the way that a watermark is embedded into content (i.e. through the use of homomorphic encryption scheme). Our scheme, however, is different from these two schemes in the following ways: (i) how and by whom (or where) the buyer's watermark is generated, (ii) the way in which buyer privacy is preserved. These differences provide our scheme with certain security and efficiency advantages over the other schemes. We next discuss these issues in greater detail.

9.7.1 Security

The key security features of our scheme are the provision of true anonymity (to the buyer), and the infeasibility of mounting collusion attacks between the content provider and one of more TTPs, e.g. a Watermark Authority. The scheme also prevents a malicious content buyer from repudiating the fact that he has distributed content without authorisation.

We first review how the three schemes generate the buyer's watermark, as this is often the source of possible collusion attacks, and highlight any shortcomings. Following which, we discuss how buyer privacy is preserved in each of the approaches.

9.7.1.1 Watermark Generation

The three schemes have the following approaches to watermark generation.

- In the scheme of Lei et al. [85], a Watermark Authority (a TTP) is employed to generate the content buyer's watermark. A collusion attack is possible, since the content provider can collude with the Watermark Authority to obtain the content buyer's watermark. The security of this scheme rests heavily on the trustworthiness of the Watermark Authority.

- Although not vulnerable to collusion, the scheme of Zhang, Kou and Fan [163] suffers from another problem. In this scheme, the buyer's watermark is jointly generated by the content buyer and the content provider. Specifically, the content buyer's watermark is computed by adding two partial watermarks, one generated by the content buyer, the other generated by the content provider. Since the buyer generates part of the watermark, and no special authority is required to generate watermarks, a collusion attack is not possible. However, the watermarking process simply involves adding two partial watermarks together and then embedding the result into the piece of content; it is therefore possible for the content buyer to remove his/her (partial) watermark from the content by subtracting it from the marked content. A content buyer can thus repudiate a claim that he/she has distributed content without authorisation.

- The scheme presented in this chapter does not suffer from the two problems mentioned above. Firstly, since the buyer's watermark is generated on the content buyer's trusted platform, and no TTP is involved in watermark generation, there is no TTP for the content seller to collude with. Hence our scheme is not vulnerable to a collusion attack. Also, using the IMSR functionality, a content seller is able to gain assurance that the watermark is generated correctly and had not been tampered with. Furthermore, a buyer is unable to remove his/her watermark W from the content, since the watermark is permuted with a random function ρ known only to the seller.

Table 9.2: Security Properties

Properties	Lei et al.	Zhang et al.	Our Scheme
Traceability	Yes	Yes	Yes
Framing Resistance	Yes	Yes	Yes
Security against Collusion	No	Yes	Yes
Privacy Preservation without TTP	No	No	Yes

Table 9.2 summarises the main security features of the three schemes. All the schemes provide traceability for the seller of copyrighted content, as well as framing resistance for the buyer. The other two properties, i.e. collusion resistance and privacy preservation, have been discussed above.

9.7.1.2 Privacy Preservation

We now discuss how a buyer's privacy is preserved by each of the schemes during content distribution.

- The schemes of Lei et al. [85] and Zhang, Kou and Fan [163] use the same method to preserve a buyer's privacy. A buyer generates an arbitrary number of key pairs (acting as pseudonyms), for use when interacting with different content sellers. These key pairs are then certified by a CA. The certificate obtained from the CA does not contain any identifying information about the buyer. A buyer can hence anonymously purchase content from content sellers using these certificates. Only the CA knows the association between keys and users; hence, if a CA is not trustworthy, the scheme may be vulnerable to collusion attack.
- In the scheme presented in this chapter, the Trusted Third Party is the DAA Issuer. The main role of the DAA Issuer is to issue the buyer with a DAA Certificate. From then on, a buyer (via his/her trusted platform) is able to generate an arbitrary number of self-certifiable pseudonyms using the DAA protocol (as discussed in Chapter 4), without the assistance of the DAA Issuer. DAA provides two levels of anonymity to a buyer. Firstly, two colluding content sellers are unable to identify whether two or more content purchase transactions have originated from the same buyer. Secondly, even when the DAA Issuer colludes with a seller, they are unable to identify a buyer based on his prior transactions. This property is the main distinguishing feature of our scheme, i.e. security against a malicious or curious Trusted Third Party. Hence, our scheme provides a buyer with true anonymity.

9.7.2 Efficiency

The watermarking scheme presented here is more efficient than the other two schemes in the following respects (see also Table 9.3):

- Firstly, there is no need for the buyer to interact with a TTP (CA) to obtain a pseudonym every time the buyer wishes to buy content, unlike the two other schemes. Once the buyer platform has obtained a DAA Certificate from the DAA Issuer, it is able to generate an arbitrary number of verifiable pseudonyms (AIKs) using the DAA Sign Protocol (as discussed in Chapter 4).

- Secondly, there is also no need for the content provider to interact with a TTP (e.g. a Watermark Authority) to obtain an encrypted watermark, unlike the Lei et al. scheme in which the Watermark Authority must always be online. All the content provider needs is the public key of the DAA Issuer.
- Finally, during the content distribution phase, there are only two message passes between the content buyer and the content provider, making it one of the most communications efficient schemes available. This feature makes the scheme well suited for deployment in mobile environments.

Table 9.3: Efficiency

Properties	Lei et al.	Zhang et al.	Our Scheme
Online TTP Requirement for:			
Watermark Generation	Yes	No	No
Privacy	Yes	Yes	No
Message Passes (between)			
Buyer - Seller	3	2	2
Seller - TTP	2	2	0

9.8 Summary

In this chapter we have provided the motivation for a novel privacy preserving CDP watermarking scheme. We identified the security threats that may arise during the process of content purchase and distribution, and derived a corresponding set of security requirements for such a setting. We then presented a privacy preserving CDP watermarking scheme which makes use of trusted computing functionality. To the best of our knowledge, this is the first trusted computing based CDP scheme. Our subsequent security analysis has shown that this scheme is able to satisfy all the identified security requirements, and a comparison with two other recently proposed schemes also shows that the novel scheme has significant advantages. The attractive security and efficiency features of our scheme also make it suitable for use in mobile environments.

Chapter 10

Conclusions

Contents

This chapter concludes the thesis by summarising the issues examined and the results achieved. We also provide suggestions for future work.

10.1 Summary of Contributions

We now summarise the contributions of this thesis. The main focus of the thesis is on ways in which trusted computing can be used to enhance the security of a mobile ubiquitous environment, in particular in securing service interactions.

In Chapter 3 we introduced the concepts of a mobile ubiquitous environment and a ubiquitous service. We motivated the need for security in a mobile ubiquitous environment, and also the need for secure service interactions. We then identified a set of ubiquitous services security requirements. The concepts of trusted computing and trusted platforms were introduced in Chapter 4, and its key functionalities were described. We also discussed related research work in trusted computing. Building on the assessment of the security issues that need to be addressed, we proposed three novel protocols for securing mobile ubiquitous services. All three of these protocols use trusted computing as the main building block. The main features of these protocols, described in detail in Chapters 7, 8 and 9, are summarised below.

In Chapter 7 we proposed Ninja, a privacy-preserving mutual authentication scheme for securing a service discovery process. The scheme is designed to address the security and privacy challenges arising in a mobile ubiquitous environment. Instead of authenticating the user identity to a service provider, the user's trustworthiness is anonymously authenticated, thereby preserving the privacy of the service user. A service discovery threat analysis was also provided and a corresponding set of security requirements was identified. The Ninja scheme exploits trusted computing functionality and achieves desirable security and privacy properties including: user anonymity, service information confidentiality, unlinkability and rogue blacklisting. The scheme is also communications-efficient, as only two message passes are required.

In Chapter 8 we proposed SDMF, a secure device management framework designed to deliver ubiquitous services to end users, whilst also hiding security management complexity from users. We conducted a service delivery threat analysis and identified a set of corresponding security requirements. Apart from providing secure service interactions, the framework helps minimise the complexity of device security management tasks for users. The framework also protects the interests of service providers by preventing unauthorised credential sharing amongst user devices. A further novel feature of the framework is that compromised devices are self-revoking, hence removing the need for cumbersome revocation infrastructure. These security objectives are achieved by assuming the presence of trusted computing functionality in the Device Management Entity (part of the ubiquitous system architecture in the context of which the novel schemes are presented), and using it in conjunction with certain other security mechanisms (e.g. the MANA protocol). Apart from the one TPM command which uses asymmetric cryptography, the SDMF employs less computationally intensive symmetric cryptographic algorithms to meet its security objectives. This makes SDMF suitable for deployment in a mobile ubiquitous environment where most of the devices are expected to be resource-constrained.

In Chapter 9 we proposed a privacy-preserving content distribution protection (CDP) watermarking scheme. The scheme allows a buyer to anonymously purchase digital content, whilst enabling the content provider to blacklist buyers that distribute content in an unauthorised manner. A CDP security threat and requirements analysis was provided. Unlike existing CDP schemes, the novel scheme minimises the reliance on a TTP for privacy protection, as a buyer can generate verifiable pseudonyms on its own. Another important feature of the scheme is that the content provider can obtain assurance that a buyer-generated watermark is well-formed. The scheme also provides the following security features: framing resistance, user anonymity, content information confidentiality, unlinkability (even against a TTP), and transaction linkability. A comparison of the scheme with two other recently proposed schemes shows that it is both more efficient (in terms of the number of message passes) and relies less on TTPs to provide the necessary security properties.

Although the Mobile VCE architecture provides the context for much of the work described in this thesis, the results are of much wider relevance. The security architectures and protocols are intended to be applicable to any analogous mobile ubiquitous system architecture.

Apart from the three novel protocols, we also analysed two trusted computing related research contributions, namely Rudolph's observation of a privacy flaw in the Direct Anonymous Attestation Protocol (DAA), and a secret distribution protocol of Sevinç, Strasser and Basin. Chapter 5 considers the privacy flaw in the TCG implementation of the Direct Anonymous Attestation (DAA) protocol, described by Rudolph. This analysis showed that, in typical usage scenarios, the weakness is not likely to lead to a feasible attack; specifically we argued that the attack is only feasible if honest DAA signers and verifiers never check the behaviour of DAA issuers. We also proposed possible ways of avoiding the attack. We pointed out that it is not an attack on the DAA protocol itself, but rather a weakness introduced in the particular use of DAA. Establishing the robustness of DAA is particularly important to this thesis, because DAA is employed in two of the proposed protocols.

In Chapter 6 we analysed a protocol due to Sevinc, Strasser and Basin. The protocol uses trusted computing functionality to secure the distribution and storage of secrets from a server to a client. We identified two inherent security weaknesses in the protocol, namely the absence of server-to-client authentication and the unauthenticated encryption of secrets sent from the server to the client. We showed how, as a result of these weaknesses, the TPM could be exploited as a signing oracle, undermining the overall security of the scheme. We proposed possible ways of making the protocol more secure.

10.2 Directions for Future Work

The protocols described in this thesis would appear to be the first proposed application of trusted computing to the secure delivery of ubiquitous services. However, this thesis has by no means addressed all the security issues arising in this domain. We now briefly review some examples of other possible security issues for ubiquitous service delivery which might benefit from the use of trusted computing technology.

- **Secure Service Selection/Recommendation.** As briefly discussed in Section 3.6, when more than one service provider is offering a service, a selection or recommendation (e.g. reputation or ranking) mechanism could be employed by a user to assist in the decision-making process. Whilst many reputation schemes have been proposed, they are typically prone to manipulation by malicious entities, e.g. competing service providers. Trusted computing could potentially be applied in this context to provide a degree of trustworthiness to reputation/ranking values. A user or selection agent would thereby gain assurance that the reputation system is trustworthy.

- **Secure Zero Configuration.** Device or network settings (e.g. an IP address, DNS settings) can be configured automatically on behalf of a user through a process known as Zero Configuration [159]. One known problem with zero configuration is that malicious devices can frequently change their IP addresses. This makes it difficult to identify and track malicious devices. The process of obtaining an IP address thus needs to be made secure, and trusted computing could potentially be used to enforce stable identities [17].

- **Anonymous Payment.** A secure service discovery scheme is given in Chapter 7. After a user has discovered a suitable service, it may be necessary to arrange for payment to the service provider. The majority of payment schemes require a user to reveal his/her identity to the service provider or to a third party. It would be desirable to integrate an anonymous payment mechanism into the Ninja service discovery process.

- **Formal Analysis.** A formal security analysis of the proposed schemes would provide a useful validation of their security properties. Well-established approaches include the complexity-theoretic approach, also known as provable security [63], and a variety of formal methods approaches, such as pi-calculus [4].

Bibliography

[1] 3rd Generation Partnership Project, Technical Specication Group Services and System Aspects, 3G Security, Valbonne, France. 3GPP TS 33.102 V7.1.0 — Security Architecture (Release 7), December 2006.

[2] 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, 3G Security, Valbonne, France. 3GPP TS 35.202 V7.0.0 — Specfication of the 3GPP Confidentiality and Integrity Algorithms, Document 2: KASUMI Specification (Release 7), June 2007.

[3] M. Abadi and C. Fournet. Private authentication. *Theoretical Computer Science*, 322(3):427–476, 2004.

[4] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: the spi calculus. In *Proceedings of the 4th ACM Conference on Computer and Communications Security (CCS '97), Zurich, Switzerland, April 1–4, 1997*, pages 36–47, New York, NY, USA, 1997. ACM Press.

[5] I. Abbadi. Authorised domain management using location based services. In *Proceedings of the 4th International Conference on Mobile Technology, Applications & Systems (Mobility'07), Singapore, September 10–12, 2007*, pages 288–295. ACM Press, 2007.

[6] I. Abbadi. Digital rights management using a master control device. In I. Cervesato, editor, *12th Annual Asian Computing Science Conference Focusing on Computer and Network Security (ASIAN'07), Doha, Qatar, December 9–11, 2007. Proceedings*, volume 4846 of *Lecture Notes in Computer Science*, pages 126–141. Springer-Verlag, Berlin, 2007.

[7] I. Abbadi and C. J. Mitchell. Digital rights management using a mobile phone. In *Proceedings of the Ninth International Conference on Electronic Commerce (ICEC 2007), Minneapolis, MN, USA, August 19–22, 2007*, pages 185–194. ACM Press, 2007.

[8] C. Adams and S. Lloyd. *Understanding PKI: Concepts, Standards, and Deployment Considerations*. Addison Wesley, second edition, 2002.

[9] A. Alsaid and C. J. Mitchell. Preventing phishing attacks using trusted computing technology. In *Proceedings of the Sixth International Network Conference (INC 2006), Plymouth, UK, July 11–14, 2006*, pages 221–228, 2006.

[10] R. C. Atkinson, J. Irvine, J. Dunlop, and S. Vadagama. The personal distributed environment. *IEEE Wireless Communications*, 14(2):62–69, April 2007.

[11] B. Balacheff, L. Chen, S. Pearson, D. Plaquin, and G. Proudler. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall PTR, Upper Saddle River, New Jersey, 2003.

[12] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad hoc wireless networks. In *Proceedings of Network and Distributed Systems Security Symposium 2002 (NDSS'02), San Diego, California, Feburary 6–8, 2002*. The Internet Society, Reston, Virgina, 2002.

[13] S. Balfe and E. Gallery. Mobile agents and the deus ex machina. In *Proceedings of the 21st International Conference on Advanced Information Networking and Applications (AINA 2007): 2007 IEEE Symposium on Ubisafe Computing (UBISAFE 2007), Niagara Falls, Canada, May 21-23, 2007*, volume 2, pages 486–492. IEEE Computer Society, 2007.

[14] S. Balfe, E. Gallery, C. J. Mitchell, and K. G. Paterson. Crimeware and trusted computing. In M. Jakobsson and Z. Ramzan, editors, *Crimeware: Understanding New Attacks and Defenses*, chapter 15, pages 457–472. Addison-Wesley, 2008.

[15] S. Balfe, E. Gallery, K. G. Paterson, and C. J. Mitchell. Challenges for trusted computing. Technical Report RHUL-MA-2008-14, Department of Mathematics, Royal Holloway, University of London, Feburary 2008.

[16] S. Balfe, A. D. Lakhani, and K. G. Paterson. Securing peer-to-peer networks using trusted computing. In C. J. Mitchell, editor, *Trusted Computing*, chapter 10, pages 271–298. IEE Press, London, 2005.

[17] S. Balfe, A. D. Lakhani, and K. G. Paterson. Trusted computing: Providing security for peer-to-peer networks. In *Proceedings of the Fifth International Conference on Peer-to-Peer Computing (P2P'05), Konstanz, Germany, August 31–September 2, 2005*, pages 117–124. IEEE Computer Society, Aug-Sep 2005.

[18] S. Balfe and A. Mohammed. Final fantasy: Securing on-line gaming with trusted computing. In B. Xiao, L. T. Yang, J. Ma, C. Muller-Schloer, and Y. Hua, editors, *4th International Conference on Autonomic and Trusted Computing (ATC 2007), Hong Kong, China, July 11–13, 2007. Proceedings*, volume 4610 of *Lecture Notes in Computer Science*, pages 123–134. Springer-Verlag, Berlin, 2007.

[19] S. Balfe and K. G. Paterson. Augmenting internet-based card not present transactions with trusted computing. Technical Report RHUL-MA-2006-9v2, Department of Mathematics, Royal Holloway, University of London, 2006.

[20] S. Balfe and K. G. Paterson. e-EMV: Emulating EMV for internet payments using trusted computing technology. Technical Report RHUL-MA-2006-10 v2, Department of Mathematics, Royal Holloway, University of London, 2006.

[21] F. Bao and R. H. Deng. Privacy protection for transactions of digital goods. In S. Qing, T. Okamoto, and J. Zhou, editors, *Third International Conference on Information and Communications Security (ICICS2001), Xian, China, November 13–16, 2001. Proceedings*, volume 2229 of *Lecture Notes in Computer Science*, pages 202–213. Springer-Verlag, Berlin, 2001.

[22] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In R. Rueppel, editor, *Advances in Cryptology — EUROCRYPT 94, 13th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Perugia, Italy, May 9–12, 1994. Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, Berlin, 1994.

[23] J. Benaloh. *Verifiable Secret-Ballot Elections.* PhD thesis, Yale University, Department of Computer Science, New Haven, Conn, USA, 1988.

[24] B. Berendt, O. Günther, and S. Spiekermann. Privacy in e-commerce: Stated preferences vs. actual behavior. *Communications of the ACM*, 48(4):101–106, 2005.

[25] M. Bond. Attacks on cryptoprocessor transaction sets. In C. K. Koç, D. Naccache, and C. Paar, editors, *Third International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001), Paris, France, May 14–16, 2001. Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 220–234. Springer-Verlag, Berlin, 2001.

[26] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment.* Springer-Verlag, 2003.

[27] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security, Washington DC, USA, October 25–29, 2004*, pages 132–145. ACM Press, 2004.

[28] E. Brickell, L. Chen, and J. Li. A new direct anonymous attestation scheme from bilinear maps. In P. Lipp, A. R. Sadeghi, and K. M. Koch, editors, *Trust 2008, Villach, Austria, March 11–12, 2008. Proceedings*, volume 4968 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2008.

[29] E. Brickell, L. Chen, and J. Li. Simplified security notions of direct anonymous attestation and a concrete scheme from pairings. Cryptology ePrint Archive, Report 2008/104, International Association for Cryptologic Research, 2008. http://eprint.iacr.org/2008/104.pdf.

[30] E. Brickell and J. Li. Enhanced privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. In *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society (WPES '07), Alexandria, Virginia, USA, October 29, 2007*, pages 21–30. ACM Press, 2007.

[31] J. Camenisch. Efficient anonymous fingerprinting with group signatures. In T. Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3–7, 2000, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*, pages 415–428. Springer-Verlag, 2000.

[32] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In S. Cimato, C. Galdi, and G. Persiano, editors, *Third Conference on Security in Communication Networks (SCN 2002), Amalfi, Italy, September 12–13, 2002. Proceedings*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer-Verlag, Berlin, 2003.

[33] R. Campbell, J. Al-Muhtadi, P. Naldurg, and G. S. Mickunas. Towards security and privacy for pervasive computing. In M. Okada, B. Pierce, A. Scedrov, H. Tokuda, and A. Yonezawa, editors, *International Symposium on Software Security (ISSS 2002), Tokyo, Japan, November 8–10, 2002. Proceedings*, volume 2609 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, Berlin, 2002.

[34] D. Chakraborty, A. Joshi, Y. Yesha, and T. Finin. Toward distributed service discovery in pervasive computing environments. *IEEE Transactions on Mobile Computing*, 5(2):97–112, 2006.

[35] D. Challener, K. Yoder, R. Catherman, D. Safford, and L. V. Doorn. *A Practical Guide to Trusted Computing.* IBM Press, Pearson plc, Upper Saddle River, NJ, USA, 2008.

[36] H. Chen, J. Chen, W. Mao, and F. Yan. Daonity — Grid security from two levels of virtualization. *Information Security Technical Report*, 12(3):123–138, 2007.

[37] L. Chen, S. Pearson, and A. Vamvakas. On enhancing biometric authentication with data protection. In R. J. Howlett and L. C. Jain, editors, *Fourth International Conference on Knowledge-Based Intelligent Information Engineering Systems & Allied Technologies (KES 2000), Brighton, UK, August 30–September 1, 2000, Proceedings*, volume 1 of 2, pages 249–252. IEEE, 2000.

[38] J.-G. Choi, K. Sakurai, and J.-H. Park. Does it need trusted third party? Design of buyer-seller watermarking protocolwithout trusted third party. In J. Zhou, M. Yung, and Y. Han, editors, *First International Conference on Applied Cryptography and Network Security (ACNS 2003), Kunming, China, October 16–19, 2003. Proceedings*, volume 2846 of *Lecture Notes in Computer Science*, pages 265–279. Springer-Verlag, Berlin, 2003.

[39] P. D. Chowdhury, B. Christianson, and J. Malcolm. Anonymous authentication. In B. Christianson, B. Crispo, J. A. Malcolm, and M. Roe, editors, *The 12th International Security Protocols Workshop, Cambridge, UK, April 26–28, 2004. Proceedings*, volume 3957 of *Lecture Notes in Computer Science*, pages 299–305. Springer-Verlag, Berlin, 2006.

[40] A. Cooper and A. Martin. Towards a secure, tamper-proof grid platform. In *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2006), Singapore, May 16–19, 2006*, pages 373–380. IEEE Computer Society Press, 2006.

[41] I. J. Cox, J. Killian, T. Leighton, and T. Shamoon. Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing*, 6(12):1673–1687, 1997.

[42] S. Crane. Privacy preserving trust agents. Technical Report HPL-2004-197, Hewlett-Packard Laboratories, Bristol, UK, November 2004.

[43] S. Creese, M. Goldsmith, B. Roscoe, and I. Zakiuddin. Authentication for pervasive computing. In D. Hutter, G. Muller, W. Stephan, and M. Ullmann, editors, *First International Conference on Security in Pervasive Computing, Boppard, Germany, March 12–14, 2003. Proceedings*, volume 2802, pages 116–129. Springer-Verlag, Berlin, 2004.

[44] J. Daemen and V. Rijmen. *The Design of Rijndael: AES — The Advanced Encryption Standard.* Springer-Verlag, Berlin, 2002.

[45] A. W. Dent and C. J. Mitchell. *User's Guide to Cryptography and Standards.* Artech House, 2004.

[46] K. Dietrich, M. Pirker, T. Vejda, R. Toegl, T. Winkler, and P. Lipp. A practical approach for establishing trust relationships between remote platforms using trusted computing. In G. Barthe and C. Fournet, editors, *Trustworthy Global Computing (TGC 2007), Sophia-Antipolis, France, November 5–6, 2007. Proceedings*, volume 4912 of *Lecture Notes in Computer Science*, pages 156–168, 2007.

[47] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[48] J. R. Douceur. The sybil attack. In P. Druschel, F. Kaashoek, and A. Rowstron, editors, *The 1st International Workshop on Peer-to-Peer Systems (IPTPS 2002), Cambridge, MA, USA, March 7–8, 2002. Proceedings*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer-Verlag, Berlin, 2002.

[49] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology — CRYPTO'84, The 4th Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, New York, USA, 1985.

[50] C. Fontaine and F. Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security*, 2007:1–10, 2007.

[51] A. Friday, N. Davies, N. Wallbank, E. Catterall, and S. Pink. Supporting service discovery, querying and interaction in ubiquitous computing environments. *Wireless Networks*, 10(6):631–641, 2004.

[52] S. Gajek, A.-R. Sadeghi, C. Stüble, and M. Winandy. Compartmented security for browsers - or how to thwart a phisher with trusted computing. In *Proceedings of the Second International Conference on Availability, Reliability and Security (ARES 2007), Vienna, Austria, April 10–13, 2007*, pages 120–127, 2007.

[53] A. S. Gajparia and C. J. Mitchell. Enhancing user privacy using trusted computing. In C. J. Mitchell, editor, *Trusted Computing*, chapter 8, pages 239–250. IEE Press, London, 2005.

[54] E. Gallery. *Authorisation Issues for Mobile Code in Mobile Systems.* PhD thesis, RHUL-MA-2007-3, Department of Mathematics, Royal Holloway, University of London, 2007.

[55] E. M. Gallery and C. J. Mitchell. Trusted computing: Security and applications. *Cryptologia*, 2009. to appear.

[56] E. M. Gallery and A. Tomlinson. Secure delivery of conditional access applications to mobile receivers. In C. J. Mitchell, editor, *Trusted Computing*, chapter 7, pages 195–237. IEE Press, London, 2005.

[57] S. L. Garfinkel, A. Juels, and R. Pappu. RFID privacy: An overview of problems and proposed solutions. *IEEE Security and Privacy*, 3(3):34–43, 2005.

[58] H. Ge and S. R. Tate. A direct anonymous attestation scheme for embedded devices. In T. Okamoto and X. Wang, editors, *10th International Conference on Practice and Theory in Public-Key Cryptography (PKC 2007), Beijing, China, April 16–20, 2007. Proceedings*, volume 4450 of *Lecture Notes in Computer Science*, pages 16–30. Springer Verlag, Berlin, 2007.

[59] C. Gehrmann, C. J. Mitchell, and K. Nyberg. Manual authentication for wireless devices. *Cryptobytes*, 7(1):29–37, 2004.

[60] B.-M. Goi, R. C.-W. Phan, Y. Yang, F. Bao, R. H. Deng, and M. U. Siddiqi. Cryptanalysis of two anonymous buyer-seller watermarking protocols and an improvement for true anonymity. In M. Jakobsson, M. Yung, and J. Zhou, editors, *Second International Conference on Applied Cryptography and Network Security (ACNS 2004), Yellow Mountain, China, June 8–11, 2004, Proceedings*, volume 3089 of *Lecture Notes in Computer Science*, pages 369–382. Springer-Verlag, Berlin, 2004.

[61] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):690–728, 1991.

[62] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[63] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:279–299, 1984.

[64] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[65] D. Gollmann. What do we mean by entity authentication? In *Proceedings of the IEEE Symposium on Security and Privacy, Oakland, California, May 6–8, 1996*, pages 46–54. IEEE Computer Society, 1996.

[66] D. Grawrock. *The Intel Safer Computing Initiative: Building Blocks for Trusted Computing.* Intel Press, 2006.

[67] U. Grossmann, E. Berkhan, L. C. Jatoba, J. Ottenbacher, W. Stork, and K. D. Mueller-Glaser. Security for mobile low power nodes in a personal area network by means of trusted platform modules. In F. Stajano, C. Meadows, S. Capkun, and T. Moore, editors, *4th European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS 2007), Cambridge, UK, July 2–3, 2007. Proceedings*, volume 4572 of *Lecture Notes in Computer Science*, pages 172–186. Springer-Verlag, Berlin, 2007.

[68] E. Guttman. Service Location Protocol: Automatic discovery of IP network services. *IEEE Internet Computing*, 4(3):71–80, 1999.

[69] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2. Request for Comments 2608, The Internet Engineering Task Force (IETF), June 1999.

[70] V. Haldar, D. Chandra, and M. Franz. Semantic remote attestation — A virtual machine directed approach to Trusted Computing. In *Proceedings of the 3rd USENIX Virtual Machine Research & Technology Symposium (VM '04), San Jose, CA, USA, May 6–7, 2004*, pages 29–41. USENIX, May 2004.

[71] International Organization for Standardization, Geneva, Switzerland. ISO 7498–2, Information processing systems — Open systems Interconnection — Basic reference model —Part 2: Security Architecture, 1989.

[72] International Organization for Standardization, Geneva, Switzerland. ISO/IEC 9797–1, Information technology — Security techniques — Message Authentication Codes (MACs) — Part 1: Mechanisms using block cipher, 1999.

[73] International Organization for Standardization, Geneva, Switzerland. ISO/IEC 9797–2, Information technology — Security techniques — Message Authentication Codes (MACs) — Part 2: Mechanisms using a dedicated hash-function, 2002.

[74] International Organization for Standardization, Geneva, Switzerland. ISO/IEC 10118–3, Information technology — Security techniques — Hash functions — Part 3: Dedicated hash functions, 2004.

[75] International Organization for Standardization, Geneva, Switzerland. ISO/IEC 9798–6, Information technology — Security techniques — Entity authentication — Part 6: Mechanisms using manual data transfer, 2005.

[76] International Organization for Standardization, Geneva, Switzerland. ISO/IEC 18033–2, Information technology — Security techniques — Encryption algorithms — Part 2: Asymmetric ciphers, 2006.

[77] International Telecommunications Union (ITU-T). Recommendation X.509 Information technology — Open Systems Interconnection — The Directory: Public-key and attribute certificate frameworks, November 2008.

[78] M. Jarrett and P. Ward. Trusted computing for protecting ad hoc routing. In *Proceedings of the 4th Annual Communication Networks and Services Research Conference (CNSR 2006), Moncton, New Brunswick, Canada, May 24–25, 2006*, pages 61–68. IEEE Computer Society, 2006.

[79] H. S. Ju, H. J. Kim, D. H. Lee, and J. I. Lim. An anonymous buyer-seller watermarking protocol with anonymity control. In P. J. Lee and C. H. Lim, editors, *5th International Conference on Information Security and Cryptology (ICISC 2002), Seoul, Korea, November 28–29, 2002. Proceedings*, volume 2587 of *Lecture Notes in Computer Science*, pages 421–432. Springer-Verlag, Berlin, 2002.

[80] M. Kinateder and S. Pearson. A privacy-enhanced peer-to-peer reputation system. In K. Bauknecht, A. M. Tjoa, and G. Quirchmayr, editors, *4th International Conference on Electronic Commerce and Web Technologies (EC-Web 2003), Prague, Czech Republic, September 2–5, 2003. Proceedings*, volume 2738 of *Lecture Notes in Computer Science*, pages 206–215. Springer-Verlag, Berlin, 2003.

[81] T. Kindberg and K. Zhang. Secure spontaneous device association. In A. Dey, A. Schmidt, and J. F. McCarthy, editors, *5th International Conference on Ubiquitous Computing (Ubicomp'03), Seattle, Washington, USA, October 12–15, 2003. Proceedings*, volume 2864 of *Lecture Notes in Computer Science*, pages 124–131. Springer-Verlag, Berlin, 2003.

[82] C. Krauß, F. Stumpf, and C. Eckert. Detecting node compromise in hybrid wireless sensor networks using attestation techniques. In F. Stajano, C. Mead-

ows, S. Capkun, and T. Moore, editors, *4th European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS 2007), Cambridge, UK, July 2–3, 2007. Proceedings*, volume 4572 of *Lecture Notes in Computer Science*, pages 203–217. Springer-Verlag, Berlin, 2007.

[83] S. Laur and K. Nyberg. Efficient mutual data authentication using manually authenticated strings. In D. Pointcheval, Y. Mu, and K. Chen, editors, *5th International Conference on Cryptology and Network Security (CANS 2006), Suzhou, China, December, 8–10, 2006. Proceedings*, volume 4301 of *Lecture Notes in Computer Science*, pages 90–107. Springer-Verlag, Berlin, 2006.

[84] G. Lawton. Is it finally time to worry about mobile malware. *IEEE Computer*, 41(5):12–14, 2008.

[85] C.-L. Lei, P.-L. Yu, P.-L. Tsai, and M.-H. Chan. An efficient and anonymous buyer-seller watermarking protocol. *IEEE Transactions on Image Processing*, 13(12):1618–1626, 2004.

[86] S. Li, S. Balfe, J. Zhou, and K. Chen. Enforcing trust in pervasive computing with trusted computing technology. In J. Lopez, editor, *First International Workshop on Critical Information Infrastructure Security (CRITIS 2006), Samos, Greece, August 31–September 1, 2006. Proceedings*, volume 4347 of *Lecture Notes in Computer Science*, pages 195–209. Springer-Verlag, Berlin, 2006.

[87] S. Li, S. Balfe, J. Zhou, and K. Chen. Enforcing trust in pervasive computing. *International Journal of System of Systems Engineering*, 1(1-2):96–110, 2008.

[88] H. Löhr, H. V. Ramasamy, A.-R. Sadeghi, S. Schulz, M. Schunter, and C. Stüble. Enhancing grid security using trusted virtualization. In B. Xiao, L. T. Yang, J. Ma, C. Muller-Schloer, and Y. Hua, editors, *4th International Conference on Autonomic and Trusted Computing (ATC 2007), Hong Kong, China, July 11–13, 2007. Proceedings*, volume 4610 of *Lecture Notes in Computer Science*, pages 372–384. Springer-Verlag, Berlin, 2007.

[89] B. M. Macq and J. J. Quisquater. Cryptology for digital TV broadcasting. *Proceedings of the IEEE*, 83(6):944–957, 1995.

[90] W. Mao, F. Yan, and C. Chen. Daonity — Grid security with behaviour conformity from trusted computing. In *Proceedings of the First ACM Workshop*

on Scalable Trusted Computing (STC'06), Fairfax, Virginia, US, November 3, 2006, pages 43–46. ACM Press, 2006.

[91] A. Martin and P.-W. Yau. Grid security: Next steps. *Information Security Technical Report*, 12(3):113–122, 2007.

[92] K. Matsui and K. Tanaka. Video-steganography: How to secretly embed a signature in a picture. *IMA Intellectual Property Project Proceedings*, 1(1):187–205, 1994.

[93] R. Mayrhofer and H. Gellersen. Shake well before use: Authentication based on accelerometer data. In A. LaMarca, M. Langheinrich, and K. N. Truong, editors, *5th International Conference on Pervasive Computing (Pervasive 2007), Toronto, Ontario, Canada, May 13–16, 2007. Proceedings*, volume 4480 of *Lecture Notes in Computer Science*, pages 144–161. Springer-Verlag, Berlin, 2007.

[94] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *Proceedings of the 2005 IEEE Symposium of Security and Privacy (SP'05), Oakland, California, USA, May 8–11, 2005*, pages 110–124. IEEE Computer Society, 2005.

[95] J. M. McCune, A. Perrig, A. Seshadri, and L. van Doorn. Turtles all the way down: Research challenges in user-based attestation. In *Proceedings of the 2nd USENIX Workshop on Hot Topics in Security (HotSec '07), Boston, MA, USA, August 7, 2007*, 2007.

[96] D. H. McKnight and N. L. Chervany. The meanings of trust. Technical report, University of Minnesota, Minneaplois, MN, 1996. http://misrc.umn.edu/wpaper/WorkingPapers/9604.pdf.

[97] N. Memon and P. W. Wong. Protecting digital media content. *Communications of the ACM*, 4(7):11–24, 1998.

[98] N. Memon and P. W. Wong. A buyer-seller watermarking protocol. *IEEE Transactions on Image Processing*, 10(4):643–649, 2001.

[99] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, FL, USA, 1997.

[100] C. J. Mitchell, editor. *Trusted Computing.* IEE Press, London, 2005.

[101] W. Mohr. The wireless world research forum — WWRF. *Computer Communications*, 26(1):2–10, Jan 2003.

[102] D. Molnar, A. Soppera, and D. Wagner. Privacy for RFID through trusted computing. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society (WPES '05), Alexandria, VA, USA, November 7, 2005*, pages 31–34. ACM Press, 2005.

[103] G. E. Moore. Cramming more components onto integrated circuits. *Electronics Magazine*, 38(8):114–117, 1965.

[104] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet public key infrastructure online certificate status protocol (OCSP). Request for Comments 2560, Internet Engineering Task Force (IETF), June 1999.

[105] National Institute of Standards and Technology (NIST). *Federal Information Processing Standards: Data Encryption Standard (DES).* FIPS Publication 46-3, National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, MD, USA, October 1999.

[106] National Institute of Standards and Technology (NIST). *Federal Information Processing Standards: Advance Encryption Standard (AES).* FIPS Publication 197, National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, MD, USA, November 2001.

[107] National Institute of Standards and Technology (NIST). *Federal Information Processing Standards: Secure Hash Standard.* FIPS Publication 180-2, National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, MD, USA, 2002.

[108] M. Nidd. Service discovery in DEAPspace. *IEEE Personal Communications*, 8(4):39–45, 2001.

[109] N. Niebert, A. Schieder, H. Abramowicz, G. Malmgren, J. S. C. Prehofer, and H. Karl. Ambient networks: An architecture for communication beyond 3G. *IEEE Wireless Communications*, 11(2):14–22, April 2004.

[110] I. G. Niemegeers and S. M. H. de Groot. Research issues in ad-hoc distributed personal networking. *Wireless Personal Communications*, 26(2–3):149–167, 2003.

[111] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT 1999, 18th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Prague, Czech Republic, May 2–6, 1999. Proceedings*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, Berlin, 1999.

[112] A. Pashalidis and C. J. Mitchell. Single Sign-On using TCG-conformant platforms. In C. J. Mitchell, editor, *Trusted Computing*, chapter 6, pages 175–193. IEE Press, London, 2005.

[113] S. Pearson. Trusted agents that enhance user privacy by self-profiling. Technical Report HPL-2002-196, Hewlett-Packard Laboratories, Bristol, UK, July 2002.

[114] S. Pearson. How trusted computers can enhance for privacy preserving mobile applications. In *Proceedings of the 2005 International Conference on a World of Wireless, Mobile and Multimedia Networks (WOWMOM 2005): First International IEEE WoWMoM Workshop on Trust, Security and Privacy for Ubiquitous Computing, Taormina, Italy, June 13–16, 2005*, pages 609–613. IEEE Computer Society, 2005.

[115] M. Peinado, P. England, and Y. Chen. An overview of NGSCB. In C. J. Mitchell, editor, *Trusted Computing*, chapter 4, pages 115–141. IEE Press, London, 2005.

[116] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. Request for Comments 3561, Internet Engineering Task Force (IETF), July 2003.

[117] A. Pfitzmann and M. Hansen. Anonymity, unlinkability, unobservability, pseudonymity, and identity management: A consolidated proposal for terminology. Version v0.31, Privacy and Data Security, Faculty of Computer Science, Institute of Systems Architecture, Technische Universität Dresden, Germany, Feburary 2008. Available at: http://dud.inf.tu-dresden.de/Anon_Terminology.shtml.

[118] B. Pfitzmann and M. Schunter. Asymmetric fingerprinting. In U. M. Maurer, editor, *Advances in Cryptology — EUROCRYPT 1996, 15th Annual Inter-*

national Conference on the Theory and Applications of Cryptographic Techniques, Zaragoza, Spain, May 12–16, 1996. Proceedings, volume 1070 of *Lecture Notes in Computer Science*, pages 84–95. Springer-Verlag, Berlin, 1996.

[119] B. Pfitzmann and M. Waidner. Anonymous fingerprinting. In W. Fumy, editor, *Advances in Cryptology — EUROCRYPT 1997, 16th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Konstanz, Germany, May 11–15, 1997. Proceedings*, volume 1233 of *Lecture Notes in Computer Science*, pages 88–102. Springer-Verlag, Berlin, 1997.

[120] F. Piper and S. Murphy. *Cryptography: A Very Short Introduction.* Oxford University Press, 2002.

[121] Platform for Privacy Preferences (P3P) Working Group. Platform for Privacy Preferences (P3P) Specifications. Version 1.1, World Wide Web Consortium (W3C), MIT, Cambridge, MA, USA, ERCIM, Sophia-Antipolis, France, Keio, Kanagawa, Japan, November 13 2006. http://www.w3.org/TR/P3P11.

[122] B. Preneel, A. Bosselaers, and H. Dobbertin. The cryptographic hash function RIPEMD-160. *Cryptobytes*, 3(2):9–14, 1997.

[123] A. Pridgen and C. Julien. A secure modular mobile agent system. In *Proceedings of the 2006 International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS '06), Shanghai, China, May 22–23, 2006*, pages 67–74. ACM Press, New York, NY, USA, 2006.

[124] G. J. Proudler. Concepts of trusted computing. In C. J. Mitchell, editor, *Trusted Computing*, chapter 2, pages 11–27. IEE Press, London, 2005.

[125] L. Qiao and K. Nahrstedt. Watermarking schemes and protocols for protecting rightful ownership and customer's rights. *Journal of Visual Communication and Image Representation*, 9(3):194–210, 1998.

[126] K. Ren, W. Luo, K. Kim, and R. Deng. A novel privacy preserving authentication and access control scheme for pervasive computing environments. *IEEE Transactions on Vehicular Technology*, 55(4):1373–1384, 2006.

[127] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[128] R. L. Rivest. The md5 message digest algorithm. Request for Comments 1321, Internet Engineering Task Force (IETF), April 1992.

[129] R. L. Rivest. The RC4 encryption algorithm. Technical report, RSA Data Security Inc, Redwood City, CA, USA, 1992.

[130] J. Rosenburg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session initiation protocol. RFC 3261, Internet Engineering Task Force, Jun 2002.

[131] RSA Laboratories. PKCS #1: RSA Cryptography Standard. Version 2.1, RSA Security, Bedford, MA, USA, 2002.

[132] C. Rudolph. Covert identity information in direct anonymous attestation (DAA). In H. Venter, M. Eloff, L. Labuschagne, J. Eloff, and R. von Solms, editors, *22nd IFIP TC-11 International Information Security Conference (SEC2007) on "New Approaches for Security, Privacy and Trust in Complex Environments", Sandton, South Africa, May 14–16, 2007. Proceedings*, volume 232 of *IFIP International Federation for Information Processing*, pages 443–448. Springer, Boston, 2007.

[133] A.-R. Sadeghi and C. Stüble. Property-based attestation for computing platforms: Caring about properties, not mechanisms. In *Proceedings of the 2004 Workshop on New Security Paradigms (NSPW '04), Nova Scotia, Canada, September 20–23, 2004*, pages 67–77. ACM Press, 2004.

[134] Salutation Consortium. Salutation Architecture Specification, June 1999. http://www.salutation.org/.

[135] R. Sandhu and X. Zhang. Peer-to-peer access control architecture using trusted computing technology. In *Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies (SACMAT '05), Stockholm, Sweden, June 01–03, 2005*, pages 147–158. ACM Press, New York, NY, USA, 2005.

[136] R. R. Schell and M. F. Thompson. Platform security: What is lacking? *Information Security Technical Report*, 5(1):26–41, 2000.

[137] S. Schwiderski-Grosche, A. Tomlinson, and D. B. Pearce. Towards the secure initialisation of a personal distributed environment. Technical Report RHUL–MA–2005–9, Mathematics Department, Royal Holloway, University of London, July 2005.

[138] P. E. Sevinç, M. Strasser, and D. A. Basin. Securing the distribution and storage of secrets with trusted platform modules. In D. Sauveron, K. Markantonakis, A. Bilas, and J.-J. Quisquater, editors, *First International Workshop in Information Security Theory and Practices: Smart Cards, Mobile and Ubiquitous Computing Systems (WISTP 2007), Heraklion, Crete, Greece, May 9–11, 2007. Proceedings*, volume 4462 of *Lecture Notes in Computer Science*, pages 53–66. Springer-Verlag, Berlin, 2007.

[139] E. Shi, A. Perrig, and L. V. Doorn. BIND: A fine-grained attestation service for secure distributed systems. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 8–11, 2005*, pages 154–168. IEEE Press, 2005.

[140] B. Smyth, M. Ryan, and L. Chen. Direct anonymous attestation (DAA): Ensuring privacy with corrupt administrators. In F. Stajano, C. Meadows, S. Capkun, and T. Moore, editors, *4th European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS 2007) Cambridge, UK, July 2–3, 2007. Proceedings*, volume 4572 of *Lecture Notes in Computer Science*, pages 218–231. Springer-Verlag, Berlin, 2007.

[141] C. Soriente, G. Tsudik, and E. Uzun. BEDA: Button-enabled device association. In *Proceedings of UbiComp 2007 Workshops: First International Workshop on Security for Spontaneous Interaction (IWSSI 2007), Innsbruck, Austria, September 16, 2007*, pages 443–449, 2007.

[142] W. Stallings. *Network Security Essentials: Applications and Standards.* Pearson Prentice Hall, Upper Saddle River, New Jersey, USA, third edition, 2007.

[143] F. Stumpf, C. Eckert, and S. Balfe. Towards secure e-commerce based on virtualization and attestation techniques. In *Proceedings of the The Third International Conference on Availability, Reliability and Security (ARES 2008), Barcelona, Spain, March 4–7, 2008*, pages 376–382. IEEE Computer Society, 2008.

[144] Sun Microsystems. Jini Architecture Specification. Version 1.2, Sun Microsystems, Palo Alto, CA, USA, December 2001. http://www.sun.com/software/jini/specs/.

[145] F. Swiderski and W. Snyder. *Threat Modeling.* Microsoft Press, Redmond, Washington, 2004.

[146] Symantec Enterprise Security. Symantec Global Internet Security Threat Report: Trends for July–December 07. Volume XIII, Symantec Corporation, Cupertino, CA, USA, April 2008.

[147] I. Teranishi, J. Furukawa, and K. Sako. k-times anonymous authentication. In P. J. Lee, editor, *Advances in Cryptology — ASIACRYPT 2004, 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5–9, 2004. Proceedings*, volume 3329 of *Lecture Notes in Computer Science*, pages 308–322. Springer-Verlag, Berlin, 2004.

[148] M. S. Thompson and S. F. Midkiff. Service description for pervasive service discovery. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops: First International Workshop on Services and Infrastructure for the Ubiquitous and Mobile Internet (SIUMI) (ICDCSW'05), Columbus, Ohio, USA, June 6–10, 2005*, pages 273–279. IEEE Computer Society, 2005.

[149] P. Tomsich and S. Katzenbeisser. Towards a robust and de-centralized digital watermarking infrastructure for the protection of intellectual property. In K. Bauknecht, S. K. Madria, and G. Pernul, editors, *First International Conference on Electronic Commerce and Web Technologies (EC-Web 2000), London, UK, September 4–6, 2000. Proceedings*, volume 1875 of *Lecture Notes in Computer Science*, pages 38–47. Springer-Verlag, Berlin, 2000.

[150] Trusted Computing Group (TCG). TPM v1.2 Specification Changes. A summary of changes, Trusted Computing Group, Portland, Oregon, USA, October 2003.

[151] Trusted Computing Group (TCG). TCG Specification Architecture Overview. Version 1.2, The Trusted Computing Group, Portland, Oregon, USA, April 2004.

[152] Trusted Computing Platform Alliance (TCPA). TCPA Main Specification. Version 1.1b, Trusted Computing Group, Portland, Oregon, USA, February 2002.

[153] Universal Plug and Play (UPnP) Forum. UPnP Device Architecture. version 1.0, December 2003. http://www.upnp.org/.

[154] R. G. van Schyndel, A. Z. Tirkel, and C. F. Osbourne. A digital watermark. In *Proceedings of the IEEE Conference on Image Processing (ICIP'94), Austin, Texas, USA, November 13–16, 1994*, pages 86–90. IEEE Press, 1994.

[155] G. Voyatzis and I. Pitas. The use of watermarks in the protection of digital multimedia products. *IEEE Proceedings*, 87:1197–1207, 1999.

[156] M. Walker and T. Wright. Security. In F. Hillebrand, editor, *GSM and UMTS: The Creation of Global Mobile Communication*, chapter 15, pages 385–406. John Wiley & Sons, New York, 2002.

[157] M. Weiser. The computer for the twenty-first century. *Scientific American*, 265(3):94–104, 1991.

[158] M. Wu and A. Friday. Integrating privacy enhancing services in ubiquitous computing environments. In *Proceedings of the 4th International UbiComp Workshop (UBICOMP 2002): Security in Ubiquitous Computing, Goteborg, Sweden, September 29 – October 1, 2002*, pages 1–5, 2002.

[159] E. D. Yan. Zero configuration networking. *The Internet Protocol Journal*, 5(4):20–26, 2002.

[160] Z. Yan and P. Cofta. A mechanism for trust sustainability among trusted computing platforms. In S. Katsikas, J. Lopez, and G. Pernul, editors, *First International Conference on Trust and Privacy in digital Business (TrustBus 2004), Zaragoza, Spain, August 30 – September 1, 2004. Proceedings*, volume 3184 of *Lecture Notes in Computer Science*, pages 11–19. Springer-Verlag, Berlin, 2004.

[161] P.-W. Yau and A. Tomlinson. Using trusted computing in commercial grids. In U. Priss, S. Polovina, and R. Hill, editors, *15th International Conference*

on Conceptual Structures (ICCS 2007), Sheffield, UK, July 22–27, 2007, Proceedings, volume 4604 of *Lecture Notes in Computer Science*, pages 31–36. Springer-Verlag, Berlin, 2007.

[162] P.-W. Yau, A. Tomlinson, S. Balfe, and E. M. Gallery. Securing grid workflows with trusted computing. In M. Bubak, G. dick van Albada, P. M. A. Sloot, and J. J. Dongarra, editors, *8th International Conference on Computer Science (ICCS '08), Kraków, Poland, June 23–25, 2008, Proceedings, Part III*, volume 5103 of *Lecture Notes in Computer Science*, pages 510–519. Springer-Verlag, Berlin, 2008.

[163] J. Zhang, W. Kou, and K. Fan. Secure Buyer-Seller Watermarking Protocol. *IEE Proceedings on Information Security*, 153(1):15–18, 2006.

[164] F. Zhu, M. Mutka, and L. Li. Service discovery in pervasive computing environments. *IEEE Pervasive Computing*, 4(4):81–90, 2005.

[165] F. Zhu, M. Mutka, and L. Ni. Prudent Exposure: A private and user-centric service discovery protocol. In *Proceedings of the Second IEEE Conference on Pervasive Computing and Communications (PerCom'04), Orlando, Florida, March 14–17, 2004*, pages 329–328. IEEE Computer Society, 2004.

[166] F. Zhu, M. Mutka, and L. Ni. A private, secure and user-centric information exposure model for service discovery protocols. *IEEE Transactions on Mobile Computing*, 5(4):418–429, 2006.

[167] F. Zhu, W. Zhu, M. W. Mutka, and L. Ni. Expose or not? A progressive exposure approach for service discovery in pervasive computing environments. In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications (PerCom 2005), Kauai Island, Hawaii, March 8–12, 2005*, pages 225–234. IEEE Computer Society, 2005.