Secure Payment Architectures and Other Applications of Trusted Computing

Shane Balfe

Technical Report

RHUL–MA–2009–10

20 February 2009

Department of Mathematics

Royal Holloway, University of London

Egham, Surrey TW20 0EX, England

http://www.rhul.ac.uk/mathematics/techreports

Secure Payment Architectures and Other Applications of Trusted Computing

Shane Balfe

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

Declaration

These doctoral studies were conducted under the supervision of Professor Kenneth G. Paterson. The work presented in this thesis is the result of original research carried out by myself, in collaboration with others, whilst enrolled in the Department of Mathematics as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Shane Balfe

October, 2008

Acknowledgements

My most heartfelt thanks goes to my supervisor, Professor Kenny Paterson, who consistently provided timely and insightful feedback on my work. I am also extremely grateful to all the people in the Information Security Group at Royal Holloway. You have helped me grow as both a researcher and a person. In particular, I would like to thank Amit Lakhani, Eimear Gallery, Hoon Wei Lim, Stephane Lo Presti, Po-Wah Yau and Anish Mohammed. Additionally, I would like to thank Chris Dale for helping to proof-read many of the chapters of this thesis. I would also like to thank my family, for their unwavering support. Finally, I would like to thank the University of London's distance learning programme; without their financial support and friendly enthusiasm I would not have been able to complete this thesis.

I dedicate this thesis to my wife, Ling-Yu Chiu.

Abstract

This thesis is divided into two distinct parts. The first part of the thesis explores the role Trusted Computing can play in securing Internet-based Card Not Present (CNP) transactions. We highlight how Trusted Platform Module (TPM) enabled Platforms, as are currently available in the marketplace, can be used as adjuncts to CNP enabling protocols, such as SSL and 3-D Secure. As an extension to this, we demonstrate how newer Trusted Computing technologies, such as processor, chipset and operating system extensions, can provide a measured virtualisation layer on top of which emulated EMV (chip and pin) cards can run.

The second part of this thesis looks at how Trusted Computing can be used to add security functionality to a number of computing paradigms. Firstly, we examine how Trusted Computing can be used to provide stable pseudonymous identities on top of which reputation systems can be built for Peer-to-Peer systems. Secondly, we examine the role Trusted Computing can play in protecting mobile agent systems. In this regard, we examine how mechanisms for protecting both agent hosts and mobile agents can be achieved by augmenting agent systems with Trusted Computing functionality.

Contents

List of Figures

List of Algorithms

Abbreviations

3-D Secure:	3-Domain Secure
AAC:	Application Authentication Cryptogram
AC:	Application Cryptogram
ACS:	Access Control Server
AFL:	Application File Locator
AIK:	Attestation Identity Key
AIP:	Application Interchange Profile
APWG:	Anti-Phishing Working Group
AR:	Access Requester
ARPC:	Authorisation Response Cryptogram
ARQC:	Authorisation Request Cryptogram
ATC:	Application Transaction Counter
AVS:	Address Verification Service
BIOS:	Basic Input/Output System
CA:	Certification Authority
CCP:	Cryptographic Coprocessor
CDA:	Combined DDA and application cryptogram generation
CL:	Camenisch-Lysyanskaya
CMK:	Certified Migratable Key
CNP:	Card Not Present
CPU:	Central Processing Unit
CRTM:	Core Root of Trust for Measurement

CSC:	Card Security Code
CVM:	Cardholder Verification Method
DRTM:	Dynamic Root of Trust for Measurement
DAA:	Direct Anonymous Attestation
DDA:	Dynamic Data Authentication
DHT:	Distributed Hash Table
DPPBM:	Deferred Physical Presence Bit Map
DRM:	Digital Rights Management
DSS:	Data Security Standard
e-EMV:	Emulated EMV
EEPROM:	Electrically Erasable Programmable Read-Only Memory
EK:	Endorsement Key
EKG:	Environmental Key Generation
EMSCB:	European Multilateral Secure Computing Base
EMV:	Europay Mastercard Visa
ICC:	Integrated Circuit Card
IKE:	Internet Key Exchange
IMC:	Integrity Measurement Collector
IMK:	Issuer Master Key
IMV:	Integrity Measurement Verifier
ISAKMP:	Internet Security Association and Key Management Protocol
IWG:	Infrastructure Work Group
JVM:	Java Virtual Machine
LPC:	Low Pin Count
MA:	Migration Authority
MAC:	Message Authentication Code
MK:	Migratable Key
MOG:	Merchant Operating Guidelines

MOTO:	Mail Order Telephone Order
MSA:	Migration Selection Authority
MVMM:	Measured Virtual Machine Monitor
NGSCB:	Next-Generation Secure Computing Base
OIAP:	Object-Independent Authorisation Protocol
OS:	Operating System
OSI:	Open Systems Interconnection
P2P:	Peer-to-Peer
PABP:	Payment Application Best Practices
PAN:	Personal Account Number
PAReq:	Payer Authentication Request
PARes:	Payer Authentication Response
PC:	Personal Computer
PCI:	Payment Card Industry
PCR:	Platform Configuration Register
PDP:	Policy Decision Point
PED:	PIN Entry Device
PEP:	Policy Enforcement Point
PGP:	Pretty Good Privacy
PIN:	Personal Identification Number
PKI:	Public Key Infrastructure
POS:	Point of Sale
PS:	Process Status
RAM:	Random Access Memory
ROM:	Read Only Memory
RTM:	Root of Trust for Measurement
RTR:	Root of Trust for Reporting
RTS:	Root of Trust for Storage

SCADS:	Stamp Collectors Against Dodgy Sellers
SDA:	Static Data Authentication
SET:	Secure Electronic Transaction
SKAE:	Subject Key Attestation Evidence
SML:	Stored Measurement Log
SRK:	Storage Root Key
SSL:	Secure Socket Layer
TBB:	Trusted Building Block
TC:	Transaction Certificate
TCG:	Trusted Computing Group
TCP:	Transmission Control Protocol
TCS:	TCG Core Services
TG:	Transaction Generator
TLS:	Transport Layer Security
TLV:	Tag Length Value
TMAP:	Trusted Mobile Agent Platform
TNC:	Trusted Network Connect
TP:	Trusted Platform
TPE:	Trusted Processing Environment
TPM:	Trusted Platform Module
TTP:	Trusted Third Party
VbV:	Verified by Visa
VM:	Virtual Machine
VMM:	Virtual Machine Monitor

Chapter 1

Introduction

Contents

In this chapter, we provide an overview of the thesis as a whole. We discuss the motivation for our research and describe the contributions of the thesis.

1.1 Motivation

The Trusted Computing Group (TCG) is an industry consortium formed to develop a set of open standards for hardware-enabled Trusted Computing. The TCG's aim is to establish a more secure computing environment on commodity platforms that can protect data from software attacks [108]. Trusted Computing is based on the inclusion of a hardware “root of trust” within a platform, that aims to allow users (and third parties) to assess the “trustworthiness” of the devices with which they interact. To achieve this, a Trusted Platform (TP) should be able to reliably gather and provide evidence of its current operating state, or any sub-component thereof. Any divergence from an intended operating state can be reported to interested parties, allowing them to make informed decisions as to whether to continue to interact with the platform in question. Trusted Computing, as discussed in this thesis, relates directly to the types of systems proposed by the TCG. Namely, a trusted system is one which will behave in a particular manner for a specific purpose[1].

At present, Trusted Platforms, as they are currently deployed in the marketplace, are incapable of providing the rich set of security services described above. The relative absence of additional hardware and software components in the form of new processor designs, new chipsets, new Basic Input/Output Systems (BIOSs) and Operating System (OS) support makes any statement (attestation) made by a current Trusted Platform unreliable. Instead, the current functionality offered by a Trusted Platform is akin to that of a high-end smartcard. Through the use of a Trusted Platform Module (TPM), a small microcontroller attached to a platform’s motherboard, a Trusted Platform can produce random numbers and generate, store and use asymmetric key pairs. As things currently stand, processor, chipset, BIOS and OS support will not be commonplace in the market for several years to come. However, the component-based architecture adopted by the TCG allows the current functionality to be extended over time. Thus, Trusted Computing can currently provide us with a limited, but useful, set of cryptographic functionality which can be extended to provide a more secure operating environment.

Once one accepts that Trusted Computing offers an interesting and powerful set of (near and long-term) security features, the natural question arises: for what

[1]https://www.trustedcomputinggroup.org/groups/glossary/

purposes can this technology be exploited? This thesis aims to examine this question by taking an in-depth look at a number of application domains whose security could be improved by the inclusion of Trusted Computing technologies. In this respect, this thesis is split into two parts. In the first part we look at card-based electronic payments and examine the problem of Card Not Present (CNP) fraud. We examine how the staged roll out of Trusted Computing technology, beginning with ubiquitous client-side TPMs, can be used to enhance the security of current CNP transactions. In the second part, we examine how the security of Peer-to-Peer (P2P) systems and mobile agent systems can be enhanced by the use of Trusted Computing technologies. We note that this thesis could also have been split into near-term and long-term applications of Trusted Computing. Here aspects of our CNP work and our Peer-to-Peer work fall into the former, whilst the rest of our work falls into the latter.

1.2 Organisation of Thesis and Summary of Contributions

The remainder of this thesis is organised as follows. In Chapter 2, we provide an overview of Trusted Computing. This chapter will act as a primer for the rest of the thesis. In this chapter we will examine the Trusted Platform Module (TPM) in considerable detail and see how it fits into the Trusted Computing paradigm. We will also look at Operating System and chipset support for Trusted Computing in the form of Microsoft's Next-Generation Secure Computing Base (NGSCB) and La-Grande technology respectively. Here we will examine how processor extensions can be used to provide isolated execution environments in which applications can run, free from external observation. The remaining chapters of this thesis are organised into two parts.

Part I: In Chapter 3, we provide an overview of Internet-based Card Not Present (CNP) transactions. We define a threat model for CNP transactions, as well as providing an overview of the generic four corner model used in card payment systems, before moving on to discuss some of the more significant protocols used for securing CNP transactions. This chapter concludes with an in-depth examination of Europay Mastercard and Visa (EMV) Integrated Circuit Cards (ICCs) for Point of Sale (PoS) transactions. This chapter will act as background for Chapters 4 and 5. In Chapter 4, we demonstrate how the staged roll out of Trusted Computing technology, beginning with client-side TPMs, can be used to enhance the security of current Internet-based CNP transactions. In Chapter 5, we demonstrate how Trusted Computing technology can be used to emulate EMV for use in Internet-based CNP transactions.

Part II: In Chapter 6, we demonstrate how features of the TCG specifications can be employed to enhance the security of P2P environments. In particular, we show how the TCG protocols for Direct Anonymous Attestation (DAA) can be used to enforce the use of stable, platform-dependent pseudonyms and thus reduce pseudospoofing in P2P networks. Taking this a step further, we show how runs of the DAA protocol can be used to build entity authentication at the level of pseudonyms and can be securely linked to the establishment of secure channels with known endpoints. Chapter 7 is concerned with the issue of mobile agent protection and the benefits that the introduction of Trusted Computing can bring to a mobile agent framework. In particular, we discuss how Trusted Computing technology can be used to support secure agent migration between platforms. We describe a number of options for securely migrating mobile agents between Trusted Computing platforms and compare and contrast the relative merits (and deficiencies) of each approach. We conclude with Chapter 8 in which we provide a summary of the preceding chapters and examine possible extensions to our work.

The main contributions of this thesis can be summarised as follows:

- We propose the use of near-term Trusted Computing technologies as a means of improving the security of Internet-based CNP transactions.
- We propose the use of longer-term Trusted Computing technologies as a means of providing a secure and extensible architecture for Internet-based CNP transactions.
- We provide an identity binding service for P2P networks that is resilient to pseudospoofing attacks. We also show how Trusted Computing can be used to bootstrap secure sub-groups within P2P networks.
- We demonstrate how Trusted Computing can provide an underlying security infrastructure for mobile agent systems.

1.3 Publications

This thesis contains material that was previously published with A.D. Lakhani and K.G. Paterson [16, 17], material that was published with K.G. Paterson [18, 20, 21, 22] as well as material that was published with E.M. Gallery [15]. The content of [18], which appeared in abridged form as [21], forms the basis for Chapter 4, the content of [20] forms the basis for Chapter 5. The content of [18] and [20] have evolved significantly since their first publication, updated versions of which can be found in [19] and [22] respectively. The content of [16, 17] form the basis for Chapter 6 and the content of [15] forms the basis for Chapter 7.

1.4 Cryptographic Primitives

In this section we introduce the cryptographic primitives that will be used throughout the remainder of this thesis.

1.4.1 Hash Functions

A hash function is a cryptographic function $h()$ that takes as input a binary string of arbitrary (but finite) bit-length, x, and outputs a string of fixed bit-length, $h(x)$ [86]. A hash function may have the following properties [86]:

- **Preimage resistance:** For all pre-specified outputs, $h(x)$, for which a corresponding input is not known, it is computationally infeasible to find an input, x', such that $h(x') = h(x)$.
- 2^{nd}**-Preimage Resistance:** Given an input, x, it is computationally infeasible to find a second input, x', where $x \neq x'$, such that $h(x) = h(x')$.
- **Collision Resistance:** It is computationally infeasible to find any two distinct inputs x and x' such that $h(x) = h(x')$.

For the remainder of this thesis we assume that any hash function used will satisfy the three security properties described above.

1.4.2 Message Authentication Codes

Message Authentication Codes (MACs) are a family of functions, $h_k()$, parameterised by a secret key k, that provide (symmetric) data-origin authentication and message integrity. MACs have the following properties [86]:

- **Ease of Computation:** Given a known function, $h_k()$, a key, k, and an input, x, $h_k(x)$ should be easy to compute.
- **Compression:** The function $h_k()$ maps an input of arbitrary (but finite) bit-length, x, to an output string of fixed bit-length, $h_k(x)$.

- **Computation-resistance:** Presented with zero or more text-MAC pairs, $(x_i,\ h_k(x_i))$, it is computationally infeasible to generate any text-MAC pair, $(x,\ h_k(x))$, for any new input $x \neq x_i$.

For the remainder of this thesis we assume that any MAC algorithm used will satisfy the three properties described above.

1.4.3 Symmetric Encryption

An encryption scheme is said to be symmetric if, given an encryption key and decryption key pair (e,d), it is computationally easy to determine e knowing only d (and vice-versa). Indeed, in many symmetric encryption schemes $e = d$. Standards for symmetric ciphers are discussed in [39].

1.4.4 Asymmetric Encryption

Asymmetric cryptography (also known as public-key cryptography) involves the generation/assignment of two keys, one public, k_p, and one private, k_s, by/to an entity. The private key should be kept secret whilst the public key can be freely distributed. In an asymmetric encryption scheme the public key is used to encrypt a message for a given recipient. This encrypted message can only be decrypted by the entity possessing the corresponding private key. Standards for asymmetric ciphers are discussed in [39].

1.4.5 Digital Signature

A digital signature mechanism is an asymmetric cryptographic scheme consisting of a private signing procedure (using a private key) and a public verifying procedure (using the corresponding public key). Digital signatures can be used to provide origin authentication, non-repudiation and/or data integrity services. Standards for digital signatures are discussed in [39].

1.4.6 Public Key Certificates

As part of a Public Key Infrastructure (PKI), Certification Authorities (CAs) issue digitally signed credentials which provide a binding between a public key and an identity (amongst other things). X.509 is a widely adopted standard for specifying the format of these public key certificates [70]. The structure of an X.509 v3 public key certificate can be found in [67].

Chapter 2

An Overview of Trusted Computing

Contents

In this chapter, we provide an overview of fundamental Trusted Computing concepts and discuss a number of Trusted Computing specifications relevant to our research. This chapter acts as a primer for the remainder of this thesis.

2.1 Introduction

Trusted Computing, refers to a collection of interrelated and interoperating technologies, which, when combined, help to establish a more secure operating environment on commodity platforms. A fully-realised trusted computing platform will allow users to reason about the behaviour of a platform, as well as providing standardised mechanisms to protect user private data against software attack [108]. The rationale behind the introduction of Trusted Computing is to provide means for end-users (and third-parties) to derive increased confidence in the platforms with which they interact. However, as trust is perhaps one of the most over-loaded terms in security research, a precise definition of trust is essential to our understanding of Trusted Computing. The definition of trust provided by the TCG centres on the notion of behavioural reputation, that trust is "the expectation that a device will behave in a particular manner for a specific purpose."[1] What this particular manner or specific purpose might be, however, has resulted in a fair amount of criticism; see for example [5, 6, 10, 126, 127, 118, 128, 137, 168]. In particular, privacy concerns relating to Trusted Platforms were raised in [111]. The extent to which Trusted Computing could be used to enable and enforce Digital Rights Management (DRM), and, more generally, the possible expropriation of platform owner control, where raised in [162, 118]. Finally, Anderson [5] expresses the view that Trusted Computing could be used to support censorship, stifle competition between software vendors, facilitate software lock-in and hinder the deployment and use of open source software, thereby potentially enabling market monopolisation by select vendors.

Our aim in this thesis is not to become embroiled in a debate over the possible (mis)appropriation of a technology, but to examine Trusted Computing and its beneficial application to a number of computing paradigms. With that in mind, we will now sketch some of the components of a Trusted Platform and the functionality

[1]https://www.trustedcomputinggroup.org/groups/glossary/

they provide.

2.2 Architecture for a Trusted Platform

Trusted Computing as discussed in this chapter, refers to the types of platforms proposed by the Trusted Computing Group (TCG). The current documentation from the TCG encompasses a vast set of specifications ranging from Personal Computer (PC) [148] and server systems [146] to specifications for trusted networking [152] and trusted mobile platforms [155]. However, it is the TCG's specifications for microcontroller design, in the form of the Trusted Platform Module (TPM), that have perhaps become most synonymous with Trusted Computing [150, 153, 154]. In addition to the TCG specification set, work outside of the TCG looks to develop new processor designs [3, 68] as well as OS support [102, 103] for Trusted Computing.

A Trusted Platform, as defined by Pearson [99], is "a computing platform that has a trusted component, probably in the form of built-in hardware, which it uses to create a foundation of trust for software processes." Figure 2.1 shows an abstract representation of such a platform. It is perhaps most illustrative to step through an "authenticated boot" process of a Trusted Platform to show how this "trusted component" (and its subcomponents) interact to provide a "trustworthy" system. Where indicated we delay discussion of some of the terminology until later in this chapter.

2.3 Authenticated Boot

Authenticated boot is the process through which a platform's configuration (state) is reliably captured and stored. Trusted Computing establishes trust in a software

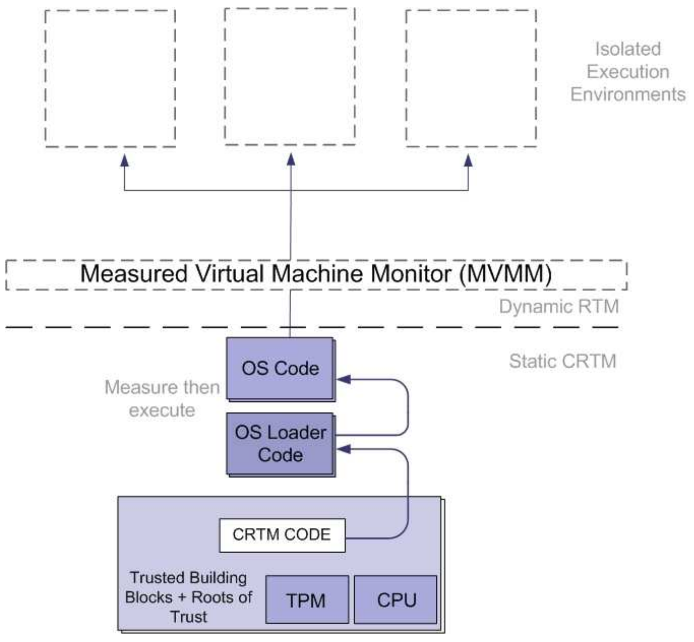
Isolated Execution Environments
Measured Virtual Machine Monitor (MVMM)
Dynamic RTM
Static CRTM
OS Code
Measure then execute
OS Loader Code
CRTM CODE
Trusted Building Blocks + Roots of Trust
TPM
CPU

Figure 2.1: Simplified Trusted Platform Architecture

process by measuring and storing representations of a pre-defined set of platform components (typically the POST BIOS, option ROMs, the OS loader and OS) in a platform's TPM. In this way, an immutable static root of trust (the BIOS) can extend its trust boundary to iteratively incorporate code that does not natively reside within this boundary. In each extension of the trust boundary, the target code is measured and recorded before execution control is transferred. Through this series of controlled handoffs a system can bootstrap larger and more complex software components in an authenticated manner.

In a PC architecture, this process occurs as follows. When powering on a Trusted Platform, a platform's Root of Trust for Measurement (RTM), in combination with a TPM (see Section 2.4) and their connections to a platform's motherboard, forms a platform's Trusted Building Block (TBB). It is the TBB which is responsible for loading a platform's initial (static) Operating System. Here, the RTM forms an immutable portion of the host platform's initialisation code that executes upon a host platform reset. This code exists either as the BIOS or the BIOS boot block, and comprises the executable component of the Root of Trust for Measurement, typically referred to as the Core Root of Trust for Measurement (CRTM) (see Section 2.4.3). This code measures the rest of the BIOS (if present) and stores a representation of the remaining BIOS code in the platform's TPM. The BIOS also measures and stores a representation of any optional Read-only Memory (ROM) code and the platform's OS loader code in the TPM. Once measured, the CRTM passes off control to the OS loader, which finally measures and stores a representation of the platform's base (static) OS. Through this sequence of measuring and executing, a transitive trust chain from the CRTM to the OS is formed. The measurements (as stored in the TPM) that make up this chain can be subsequently compared against known good measurements by either the user of the platform or some interested third party. If a mismatch is detected then the platform has failed to boot into an expected state and remedial action may be taken.

Alternatively, a Trusted Platform can launch a Dynamic-RTM (DRTM), which in turn can launch dynamic operating systems whose configurations are also measured and stored in the platform's TPM. The concept of a DRTM, as defined by Intel in their LaGrande system architecture [68], refers to a protected isolation domain running on-top of a Measured Virtual Machine Monitor (MVMM). As we will see in Section 2.7, the DRTM, much like the CRTM, is an immutable portion of the platform, but unlike the CRTM does not require a platform reset. The DRTM is responsible for loading dynamic Operating Systems in isolated partitions. These partitions run in parallel to standard OS partitions, without requiring a system reboot, and can be used to run arbitrary code free from observation from other partitions within a single Trusted Platform.

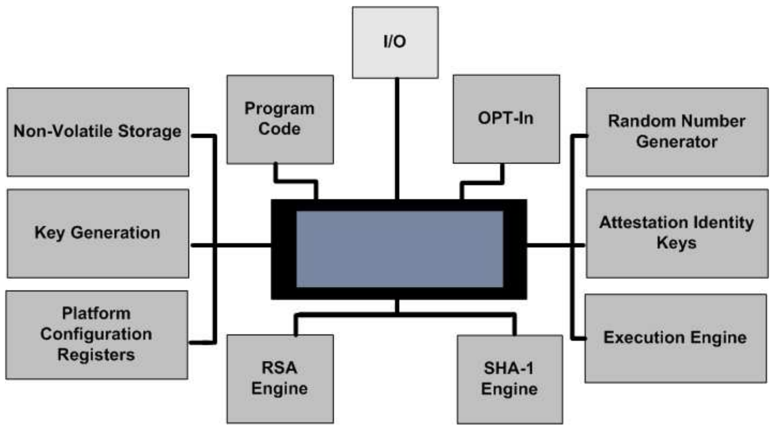

Figure 2.2: Components of Trusted Platform Module

2.4 TPM Specifications

The TPM forms the core of all efforts in Trusted Computing. A TPM (see Figure 2.2) is a microcontroller with Cryptographic Coprocessor (CCP) capabilities that is uniquely bound to a platform's motherboard. A TPM provides a platform with the following facilities: a number of special purpose registers for recording platform state (current configuration); a means of reporting this state to remote entities; secure volatile and non-volatile memory; random number generation; a SHA-1 [92] hashing engine; and asymmetric key generation, encryption and digital signature capabilities. The TPM specifications have evolved over many years, beginning with version 1.1b and most recently with version 1.2 revision 103 [150, 153, 154].

Before a TPM can be used to provide any of the facilities described above to other processes, a TPM must first be "enabled", "activated" and "owned".

2.4.1 Taking Ownership of the TPM

The default delivery state for a new TPM is disabled, inactive and unowned. Before a TPM can become fully operational it must move though a number of states, culminating in TPM ownership. In taking ownership of a TPM, certain commands require an assertion of physical presence at a platform before the command will operate. Such operations include: enabling a TPM; activating a TPM; taking ownership of a TPM; clearing the existing owner of a TPM; temporarily deactivating a TPM and temporarily disabling a TPM. To operate these commands the TPM must obtain "unambiguous assurance" that the operation is authorised by an entity that is physically present at a platform [153]. Examples of such assurance would include "depressing a switch, setting a jumper, depressing a key on the keyboard or some other such action" [153]. When a platform is turned on for the first time, the platform informs the TPM that it is performing a boot process by sending a physical (bus) signal to the TPM. As part of this process the TPM performs a number of self-diagnostic routines to ensure that it is functioning correctly. After this, the TPM performs a start-up sequence which prepares the TPM for operation.

Enabling and Activating: A disabled TPM is unable to execute any commands that use TPM resources. At this point a disabled TPM requires a demonstration of physical presence in order to transition it from a disabled to an enabled state. Much like a disabled TPM, a deactivated TPM also cannot execute commands that use TPM resources. However, an enabled but deactivated TPM can begin the process of taking ownership of a TPM.

Taking Ownership: Taking ownership of a TPM is the process of inserting a secret authorisation value (usually a password) into a TPM shielded location (see Section 2.4.2). This secret value is later used to demonstrate proof of ownership when using certain TPM operations that require "TPM_Owner" authorisation. Any entity that can demonstrate knowledge of this shared secret is considered the TPM_Owner. As part of the ownership process, the TPM generates a new Storage Root Key (SRK, see Section 2.4.4). Once an owner is established, the addition of a new owner requires the removal of the old owner. This invalidates all information associated with the old owner, including the SRK and all keys stored under it. Consequently, all data that is protected with TPM keys generated by the old owner will be unrecoverable.

2.4.2 Protected Capabilities and Shielded Locations

The TPM provides a number of internally shielded locations (memory and registers where it is safe to operate on sensitive data) that are only accessible through the use of a set of commands with exclusive permission to access these locations (protected capabilities). Perhaps the most important shielded locations in terms of any discussion of Trusted Computing functionality are the Platform Configuration Registers (PCRs), which we will now consider in detail.

2.4.3 Integrity Measurement and Storage

Trusted Computing defines several roots of trust within a TPM. Integrity measurement and storage fall under the purview of the Root of Trust for Measurement (RTM) and the Root of Trust for Storage (RTS) respectively. The RTM is a computing engine responsible for making intrinsically reliable integrity measurements. By the same token, the RTS is responsible for maintaining an accurate and sufficiently detailed record of the events measured by the RTM. The word integrity in this context refers to ensuring that state transitions are accurately reflected in the current platform state.

An integrity measurement as defined in [103] is the cryptographic digest or hash of a platform component that is capable of altering a platform's integrity (typically a piece of software executing on the platform). For example, an integrity measurement of a program can be calculated by computing a cryptographic digest of a program's instruction sequence, its initial state (i.e. the executable file) and its input. An integrity metric is then a digest of one or more integrity measurements [100]. Integrity metrics are stored in a TPM's PCRs.

Measurement: Examining the TCG integrity measurement and storage mechanisms at a high level, we note two structures of particular importance, namely the aforementioned PCRs and the Stored Measurement Log (SML), alternatively referred to as the event log. A PCR is a 20-byte storage area that acts as a repository for recording integrity altering events within a platform. The SML is responsible for maintaining an ordered database of integrity changing events. Each PCR has responsibility for the maintenance of a cumulative digest of one or more of these events (integrity metrics). The data held in a PCR, in conjunction with the relevant portion of the SML, is used as evidence to attest to a current platform state (see Section 2.4.5).

PCR updates occur by appending the new event to be measured to the existing value contained in the PCR, and taking a SHA-1 hash of this value. This can be expressed as follows:

$$PCR_x := \text{HASH}\ (PCR_x \parallel \text{Integrity Measurement})$$

Here x is the number of the PCR being updated. This operation is commonly referred to as *extending* the digest. The use of this approach guarantees that related integrity measurements will not be discarded, as the previous register value is incorporated into the new register value. It also ensures that updates to a PCR are not commutative: the order in which events occur and the measured values they contain both affect the value of the stored digest.

PCRs and the SML are intimately related. Without the representative digests contained in PCRs, the record of events maintained by the SML are unreliable (as there is nothing to stop someone from generating false events in the log). Without the relevant portion of the SML, a PCR digest is devoid of any meaning (as the context from which the digest was generated would be lost). Consistency between the two structures is preserved by the fact that PCRs are maintained internally to the TPM and that PCR extension operations are only permitted via TPM protected capabilities. These PCRs provide evidence of attempted tampering of the log, since the sequence of events tied to a specific PCR can be extracted from the SML, rehashed and compared to the actual value contained in a PCR. Thus, the ability to securely update PCR registers is integral to the trustworthiness of a platform.

The two most important properties abstracted from the brief discussion outlined above are firstly, that atomicity of extensions is ensured, and secondly, that updates are noncommutative.

Storage: Storage, as defined by the TCG, refers to two distinct concepts. The first, which we just covered, refers to the intermediate step between measurement and reporting. The second, which we will discuss in the next section, refers to the protection of keys and data within a TPM. The responsibility for the protection of said keys and other sensitive data within a TPM falls under the oversight of the RTS. In this regard, the RTS maintains a small area of volatile memory used for performing cryptographic operations and storing data.

2.4.4 TPM Keys and Key Usage

A TPM can potentially generate an unlimited number of asymmetric key-pairs. All keys generated by the TPM (with one exception) are arranged in a hierarchy with the SRK at its root (see Figure 2.3). Every node (TPM_Key) in the tree has an assigned attribute designation as well as a defined key type. Attribute designations, in the context of a TPM_Key, help to define key mobility. A key can be designated

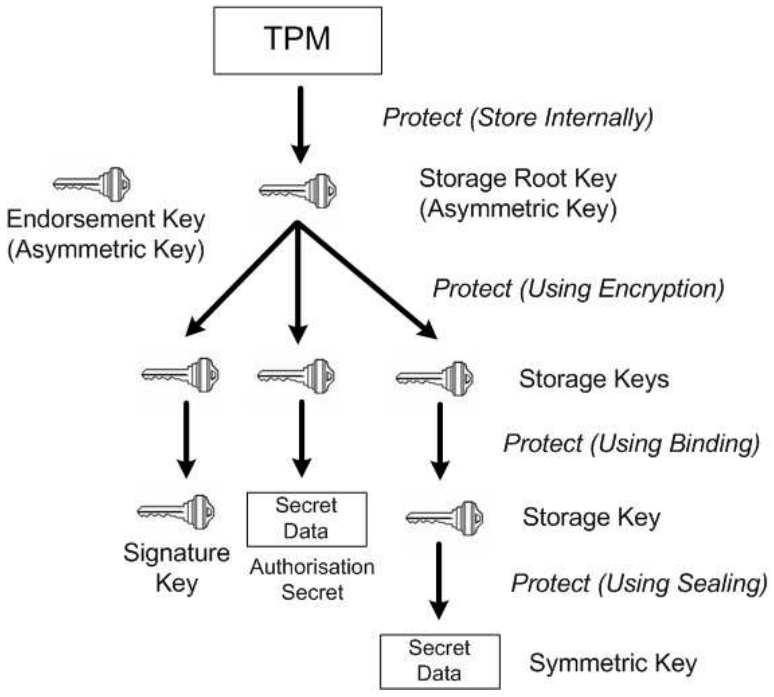

Figure 2.3: TPM Key Tree Hierarchy

as migratable, certified-migratable (see Section 2.4.4.1) or non-migratable. Private keys with either of first two designations are capable of moving from one TPM to another, whilst non-migratable private keys are inextricably bound to a single TPM instance. Only TPM-Keys that have the migratable bit set in their TPM_Key_Flags data structure are allowed to migrate from a TPM [150]. Key types are used to define which operations a TPM_Key is capable of performing. For example, a key could be of signing type or storage type, depending on the designation of its usage.

To add a new key to the hierarchy it must be "wrapped" (encrypted) using an existing storage key in the hierarchy, creating a key "blob". These blobs are opaque outside of the TPM and may be bound to the platform on which the TPM resides (depending on the attribute designation of the storage wrapping key). If the key is designated non-migratable then blobs may only be decrypted on the platform that created them, as the decryption key will be fixed to a particular TPM. If, however, both the wrapping key and the new key are migratable, then both blob and wrapping key can be transferred to a different platform.

Within a TPM there are two types of key pair that hold special significance. These are the Endorsement Key pair (EK) and the Attestation Identity Key pairs (AIK). Within a TCG-conformant platform, AIK key pairs act as aliases for the EK key pair and are responsible for attesting to platform states (see Section 2.4.5). AIK key pairs are used because an EK key pair is unique per TPM instance and its use is considered a possible risk to user privacy should the EK public key become connected with personally identifiable information. As there is no prescribed limit on the number of AIK key pairs that can be used within a platform, this provides an anonymity mechanism, whereby a TPM can, for example, use different AIK key pairs each time it attests to platform integrity metrics.

2.4.4.1 Certified Migration

Certified Migration is a new concept introduced in the latest TPM specifications [153]. Certified Migration is an operation that permits the secure transfer of Certified Migratable Keys (CMKs) from one TCG-compliant platform to another. This transfer occurs in such a manner as to allow the new platform full usage of the migrated key. The notion of a migratable key can be subdivided into two categories, a Migratable Key (MK) and a CMK. MKs are keys that, while not bound to a specific TPM, can be transferred to another TPM if appropriate local authorisation is provided. CMKs are similar, insofar as they also require appropriate authorisation in order to migrate. However, their migration is conditional on receiving permission from a relevant external third party called a Migration Selection Authority (MSA) or Migration Authority (MA). An MSA is a third party responsible for approving the migration of a CMK to a specified destination (but does not directly handle the key itself). An MA performs a similar function to that of an MSA; however, an MA will act as a temporary storage area for a CMK while the CMK transits to its ultimate destination. Without this approval, the TPM will refuse to release the private CMK for migration.

2.4.4.2 Binding and Sealing

Protected storage functionality uses asymmetric encryption to protect the confidentiality of data on a TPM host platform. Protected storage also provides implicit integrity protection for TPM objects. Both data and keys can be associated with a string of 20 bytes of authorisation data before being encrypted. When decryption is requested, the authorisation data must be submitted to the TPM. The submitted authorisation data is then compared to the authorisation data associated with the object, and the object will only be released if the values match.

The notions of *binding* and *sealing* are of fundamental importance to Trusted Computing. Binding refers to the encryption of data with a public key for which the corresponding private key is non-migratable from the recipient's TPM. In this way, only the TPM that manages the non-migratable private key will be capable of decrypting the message. Sealing takes binding one step further. Sealing is the process by which sensitive data can be associated with a set of integrity metrics representing a particular platform configuration, and encrypted. The protected data will only be decrypted and released for use by a TPM when the current state of the platform matches the integrity metrics to which the data is sealed.

Using sealed storage, an end-user can protect their private data by making revelation of that data contingent on a platform being in a particular state. For example, a user can seal credit card data to a state that requires a particular banking application to be running on the platform, and nothing more. The presence of additional memory-resident applications would change the platform state and prevent a sealed object from being decrypted.

Depending on how one constructs the key used for binding, binding can replicate the TPM's sealing mechanism. When generating a new non-migratable private key, if we specify a set of platform metrics (as represented by PCR values) that must be present before the private key can be used, any object encrypted with the public binding key will be bound to the state specified at key creation. The only difference between this approach and sealing is that it is a key that is bound to a platform state, whereas with sealing it can be any data object.

2.4.5 Reporting and Attestation

The final root of trust within a Trusted Platform is the Root of Trust for Reporting (RTR). Its function is to faithfully recount information held by the RTS to third parties. The mechanism through which this is achieved is referred to as "binary attestation". For the remainder of this thesis, we refer to this as simply attestation.

Attestation within the context of Trusted Computing is the mechanism whereby a TPM-enabled platform adduces certain state characteristics that govern a platform's trustworthiness. This evidence is consumed by integrity verifiers that use this evidence, in conjunction with the perceived trustworthiness of the entity performing the attestation, to establish a trust level in the attesting platform. The TCG attestation protocol is illustrated in Figure 2.4 and outlined below.

1. An external entity requests one or more PCR values.
2. A platform agent culls the SML for the events responsible for generating the requested PCR values.
3. The TPM attests (signs) the requested registers using an AIK private key by

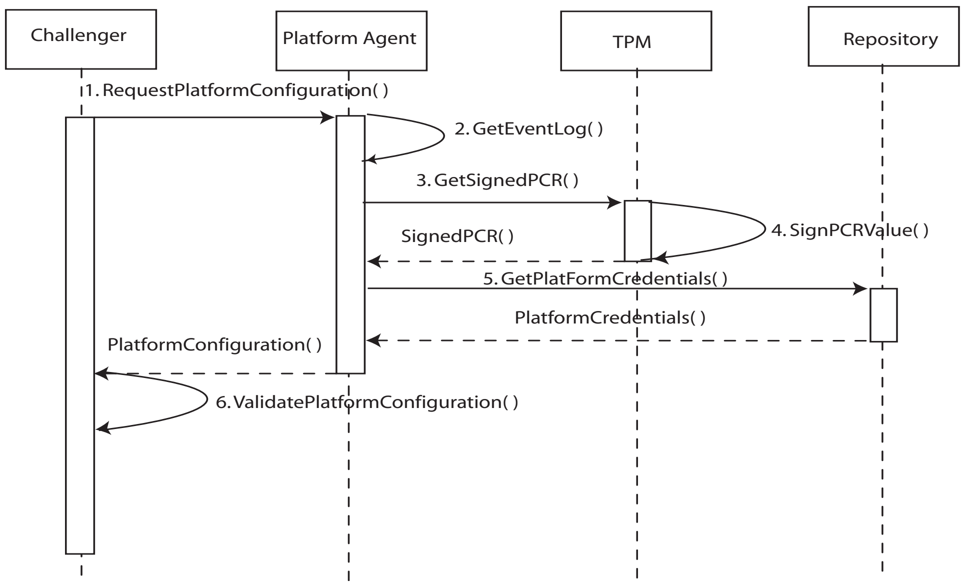

Figure 2.4: TCG Attestation Protocol [151]

calling the TPM_Quote command. [154].

4. The agent then obtains credentials that vouch for the TPM (see Section 2.4.6). These credentials, along with the relevant portion of the SML and the requested signed PCR values, are returned to the requesting entity. We will discuss TCG credentials in greater detail in Section 2.4.6.

5. The requesting entity then examines the credentials, checks signatures and compares a hash of the received SML entries to the attested PCR values. If they match the verifier can gain some assurances as to the state of the attestor's platform at the time the attestation was made.

However, in order for the values being attested to by a platform to have meaning outside of the confines of a challenger's platform, it is necessary for the challenger's platform to first obtain a credential for the AIK key pair (strictly the public component of this key pair) from a trusted third party recognised by the verifier. How this credential is obtained differs between version 1.1b and version 1.2 of the TCG specifications. Version 1.1b uses what is referred to as the Privacy CA model whilst version 1.2 introduced an (optional) new model, Direct Anonymous Attestation (DAA), whilst also retaining the Privacy CA model.

2.4.5.1 Privacy CA

The Privacy CA approach is employed as a means of issuing AIK credentials (see Section 2.4.6) to TPM-enabled platforms. These credentials are in turn used to vouch for the trustworthiness of an attestation originating from a platform. Credential acquisition is achieved as follows:

1. A requesting TPM-enabled platform first generates an AIK key pair and gathers all the necessary information for a Privacy CA to examine the platform. This information includes the public component of the newly generated AIK key, as well as various credentials that vouch for the trustworthiness of the TPM-enabled platform (see Section 2.4.6). Once collected, these are sent to the Privacy CA.

2. Provided the credentials presented by a user's platform can be validated by a Privacy CA, the Privacy CA will generate an X.509 certificate called an AIK credential. A Privacy CA will encrypt the newly created AIK credential with a symmetric key, which in turn is encrypted with the public component of the EK of the requesting platform and returned to the requesting platform.

3. Only the specific requesting platform is capable of "activating" (decrypting) the new AIK credential using its EK private component. Once decrypted, this then allows the AIK private component to be used to generate third-party verifiable signatures over platform integrity metrics (assuming the third-party trusts the Privacy CA which generated this AIK credential).

Significant concerns were raised over the level of "privacy" afforded by this solution [126]. In this model, the Privacy CA is capable of linking individual AIK public keys to a particular TPM. This is because in order to obtain an AIK credential it is necessary for a TPM to disclose its AIK-EK binding. Another potentially major issue with the Privacy-CA model remains the distinct lack of a business case for setting up and running such a CA.

2.4.5.2 Direct Anonymous Attestation

By contrast to the Privacy CA model, in the DAA approach the issuing authority should not be able to link multiple AIK credentials with a single platform identity. The DAA approach differs from the Privacy CA model in that it does not disclose AIK keys to an issuer, and hence aims to limit the exposure of AIK-EK bindings.

Enrolling a TPM-enabled platform with a DAA issuer occurs when a platform instigates a run of the DAA *join protocol*, see Figure 2.5. In this protocol, the platform obtains a credential from a trusted third party called a *DAA Issuer*. In practice, the role of the issuer might be played by the platform manufacturer or by a third party who wishes to offer services exploiting TPM features.

During the join protocol the platform asks to become a member of the group of platforms to which credentials have been issued by that issuer. During this process, the platform proves that it has knowledge of a specific, TPM-controlled, non-migratable secret value f, and is authenticated to the issuer via its EK credential (see Section 2.4.6). If the issuer accepts in this step, then it provides the platform with a credential in the form of a *Camenisch-Lysyanskaya (CL) signature* [29] on the secret value f. It is this credential that will later enable the platform to convince a verifier that it has previously successfully completed the join protocol with a particular issuer. Note that the issuer does not actually learn f in this process.

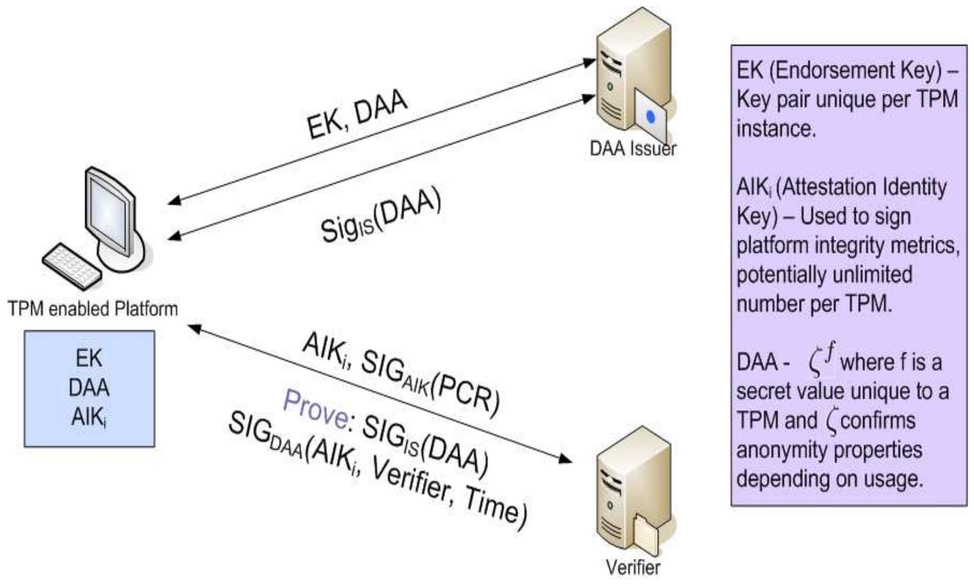
EK, DAA
Sig$_{IS}$(DAA)
DAA Issuer
TPM enabled Platform
EK
DAA
AIK$_i$
AIK$_i$, SIG$_{AIK}$(PCR)
Prove: SIG$_{IS}$(DAA)
SIG$_{DAA}$(AIK$_i$, Verifier, Time)
Verifier
EK (Endorsement Key) – Key pair unique per TPM instance.
AIK$_i$ (Attestation Identity Key) – Used to sign platform integrity metrics, potentially unlimited number per TPM.
DAA - ζ^f where f is a secret value unique to a TPM and ζ confirms anonymity properties depending on usage.

Figure 2.5: Simplified DAA Protocol

In order for a platform to obtain a credential for its secret value f, the platform must reveal a pseudonym N_I of the form ζ_I^f, where ζ_I is derived from the issuer's name and is an element of some suitable group. By maintaining a history of pseudonyms that it has seen, the issuer can check if credentials have already been issued to any given TPM. The issuer can also check for rogue platforms at this stage, by testing if $N_I = \zeta_I^f$ corresponds to an f appearing on a blacklist.

Once the join phase is complete, the TPM (with the help of the platform) can use the DAA-Signing algorithm to prove to any verifier that it has an appropriate issuer credential (in the form of a CL signature), and at the same time, sign a message m using the secret value f as a private key. Here, m is typically the public component of an AIK key pair, but it can also be an arbitrary external 160-bit string [27]. In the former case, the output of the DAA-Signing algorithm amounts to an attestation that the TPM is in possession of a valid issuer credential and has a particular AIK. This gives the verifier a similar assurance concerning that AIK's linkage to a particular platform to that gained using the Privacy CA approach described above. In turn, this allows any PCR value signed by the particular AIK to be checked as being genuine by the verifier. Corresponding to the DAA-Signing algorithm is a DAA-Verify procedure that is executed by the verifier.

In the DAA-Signing protocol, the platform is only identified by a pseudonym. As in the join phase, this pseudonym is of the form $N_V = \zeta^f$, where now the selection of ζ determines the anonymity properties of the attestation process. The value of the base ζ will typically be chosen by the verifier. In this case, if the verifier fixes on a choice of ζ (deriving ζ from its name, for example), then this will also fix the value of the pseudonym N_V used by a particular platform. This will allow all the transactions involving a particular TPM to be linked. As the secret value f is non-migratable, only the TPM that generated that value should be capable of using it. If a value of f is extracted from the TPM and distributed to other platforms, then the blacklisting approach used during the join phase can be used to detect rogue platforms, provided verifiers have access to the list of compromised f values. The value of ζ might also be selected at random by the platform for each transaction with the verifier — this would allow the platform's transactions to become completely unlinkable.

The full cryptographic details of DAA can be found in [27], while the TCG specification [154] contains implementation details.

2.4.5.3 Problems with Attestation

Given the potential difficulty for a verifier to assess if every operating system and every patch has a desired trusted configuration, verifiers may be forced to concentrate on a limited set of mainstream configurations. In this regard, the current attestation mechanism proposed by the TCG has been criticised for shifting "some of the costs of achieving interoperability onto those who have the least incentive to bear them" [129]. In addition, binary attestation has been noted as having the following deficiencies [65, 120]:

- At present attestation only tells a verifier what version of a particular application is running. It says nothing about how an application will behave.
- Due to the current nature of patch management, the application of patches may potentially result in a large number of distinct configurations for a single application.
- Revoking a particular version of an application may be difficult, given the non-commutative nature of patching.
- The challenger in an attestation gains exact information about the software configuration of a platform.

To combat some of these perceived inadequacies with binary attestation, the following proposals have been put forward:

Semantic Remote Attestation [65]: In order to separate behaviour-based attestation from binary attestation, semantic remote attestation introduces a trusted Java Virtual Machine (JVM) that is capable of attesting various properties of a Java application executing within that JVM (without revealing the code identity of the application). In this approach, software up-to-and-including the JVM must be measured using the binary attestation mechanism outlined in Section 2.4.5. This measured JVM can then be used to attest various properties of code running within it. For example, in the SETI@Home project, a server may wish to test the floating point behaviour of the system by having the JVM run a test suite of floating point number programs. The results of this test suite can be attested to by the JVM and provide assurances of how the system behaves with respect to its handling of such numbers. Unfortunately, there may be a limit on what semantic remote attestation can actually attest to using this approach as an attesting host may be unwilling to run arbitrary code of unknown provenance.

Property Based Attestation [120]: This approach introduces an additional layer of indirection into the attestation process. Instead of expecting a verifier to determine if a particular state is trustworthy or not, a platform state should be certified (by a trusted third party) as satisfying certain properties. A platform should then be capable of attesting that its current configuration matches such a property, allowing a verifier to infer whether a platform is trustworthy or not without knowing which particular software is running. The authors of [120] suggest that this approach could be realised either through a Trusted Attestation Service (similar to the trusted JVM approach of semantic remote attestation) or through modifications to the TPM hardware. Unfortunately, this approach shifts the problems with binary attestation to an entity other than the verifier. Many of the original problems persist, and, as with the Privacy CA, there is yet to emerge a clear business case for an entity to provide such a service. Additionally, exactly what properties can be satisfied using such an approach remains an open question. For the remainder of this thesis, we predominantly concentrate on binary attestation as specified by the TCG. Where indicated we will highlight how these alternative attestation proposals could be used to improve our solutions.

2.4.5.4 Subject Key Attestation Evidence

The Subject Key Attestation Evidence (SKAE) X.509 extension was proposed as a means of coping with difficulties in integrating TPM-controlled keys with standard security protocols [145]. An SKAE X.509 certificate, issued by an SKAE CA, provides a mechanism to allow a verifier to ascertain that an operation involving a private key was performed within a TCG-compliant TPM environment. Once issued, an SKAE certificate can be used as an aid to authentication in protocols such as SSL/TLS [42] or IKEv2 [105, 125].

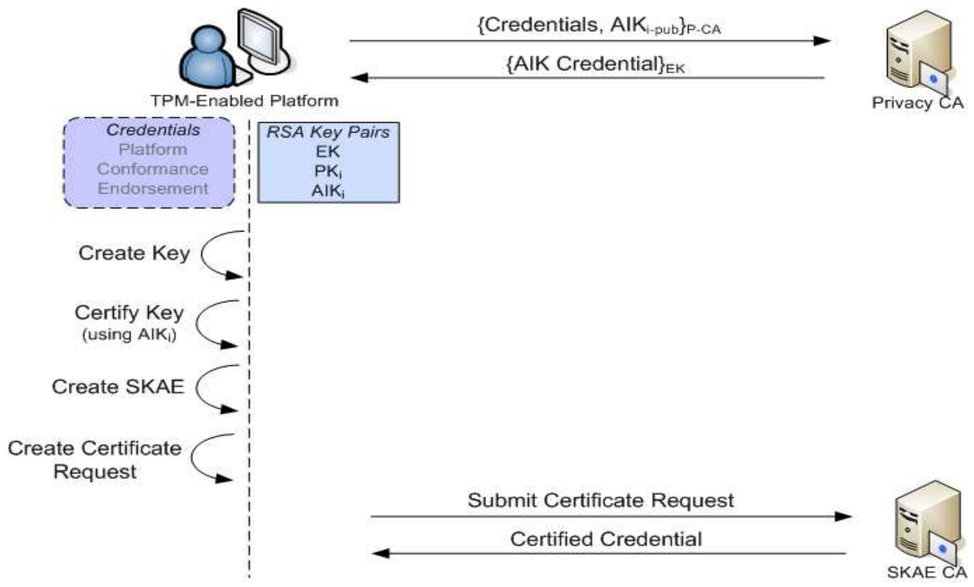

Figure 2.6: Simplified SKAE Protocol

The process of obtaining an SKAE certificate occurs as follows (see Figure 2.6):

1. A user instrusts their TPM to generate a new asymmetric key pair, K_i. In generating this key, the user specifies mobility, usage and authorisation constraints for the private portion of K_i.
2. A user's TPM signs (using the private component of a previously certified AIK key pair), the public component of K_i to produce a TPM_Certify_Info structure.
3. A user applies to an SKAE CA for certification of the public component of K_i by creating a certificate request package incorporating the TPM_Certify_Info structure.
4. If the SKAE CA is satisfied as to the AIK/public key binding, then an X.509 public-key certificate is issued by the SKAE CA to the platform. Here the certificate not only includes the public portion of K_i which has been cryptographically bound to a TPM, but also includes enough information for the

relying party to validate this binding.

2.4.6 TCG Credentials

The issue of certification plays an important role in defining what is meant by a Trusted Platform. In this regard there are five interrelated credentials (X.509 certificates) that adduce the existence of a correctly functioning TPM within a platform. These credentials and their relationships are shown in Figure 2.7 and discussed below.

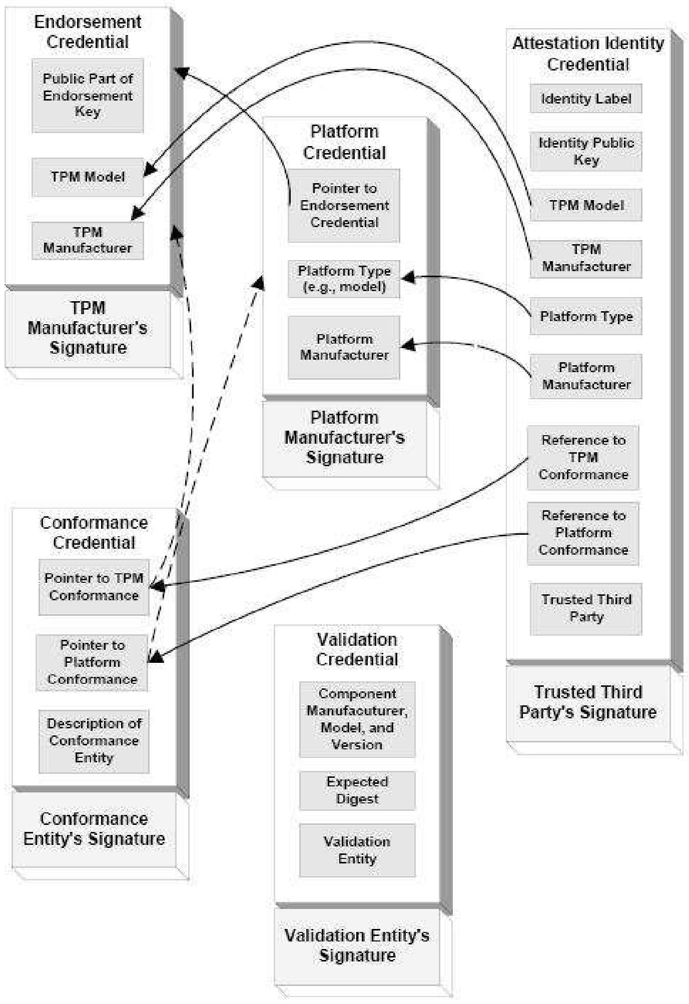
Endorsement Credential
Public Part of Endorsement Key
TPM Model
TPM Manufacturer
TPM Manufacturer's Signature
Platform Credential
Pointer to Endorsement Credential
Platform Type (e.g., model)
Platform Manufacturer
Platform Manufacturer's Signature
Attestation Identity Credential
Identity Label
Identity Public Key
TPM Model
TPM Manufacturer
Platform Type
Platform Manufacturer
Reference to TPM Conformance
Reference to Platform Conformance
Trusted Third Party
Trusted Third Party's Signature
Conformance Credential
Pointer to TPM Conformance
Pointer to Platform Conformance
Description of Conformance Entity
Conformance Entity's Signature
Validation Credential
Component Manufacuturer, Model, and Version
Expected Digest
Validation Entity
Validation Entity's Signature

Figure 2.7: TCG Credentials and their Relationships [151]

- **Endorsement credential:** This credential vouches that a TPM is genuine and is issued by whomever generated the EK pair, typically the TPM manufacturer or vendor.
- **Conformance credentials:** These credentials vouch that the TPM design conforms with established evaluation guidelines as laid out by the TCG. These credentials are issued by entities with sufficient credibility to evaluate a TPM, typically conformance testing facilities.
- **Platform credential:** This credential provides evidence that a platform contains a TPM and incorporates references to both the endorsement and conformance credentials. This is typically generated by a platform vendor.
- **AIK credentials:** These credentials are issued per AIK key pair and attest that a particular AIK key pair belongs to a specified TPM, and that the AIK key pair is tied to valid endorsement, platform and conformance credentials. These credentials are issued by Privacy CAs.
- **Validation credential:** This credential provides signed reference measurements (digests of the measured components, such as software, taken during development when the components are believed to be in a stable condition) which can be compared against load-time measurements to assure interested parties that their application is performing as intended. These load-time measurements will be signed by AIK private keys.

2.5 Trusted PC and Trusted Server Specifications

Both the Trusted PC [148] and Trusted Server [146] specifications provide generic implementation reference documentation for building Trusted Platforms at the client and server-side respectively.

The Trusted PC specification deals primarily with instantiating the various elements involved in transitioning a platform from its pre-boot to its post-boot state. Within this transition, various usage constraints are defined for TPM components, such as usage of PCR registers, as well as guidelines for handling option ROMs. The specification also outlines programmatic interfaces to the BIOS, as well as detailing how physical presence can be demonstrated on a TPM-enabled platform.

The Server specification details much of the same functionality as the Trusted PC specification, but introduces a number of new concepts that are of special interest in server architectures. In particular, it highlights the distinctions between partitions and platforms. The specification defines a partition as "the hardware and firmware environment, which provides the support for the execution of a single operating system image within a separated trust environment" [146]. A platform is later defined as "the complete package of physical hardware and firmware that a manufacturer produces for a single server product. This may be configured as one or more partitions" [146]. The issue of dynamic reconfiguration is addressed only insofar as stating that the measurement of asynchronous events is the responsibility of the post-boot software environment.

2.6 TNC Specification

The Trusted Network Connect (TNC) specification [152] forms a subclass of the Infrastructure Work Group (IWG) interoperability specification [147], and deals predominantly with enabling the enforcement of operator-controlled policies for endpoint security in determining network access.

TNC offers a way of verifying an endpoint's integrity to ensure that it complies with a particular predefined policy, for example, ensuring that a certain software

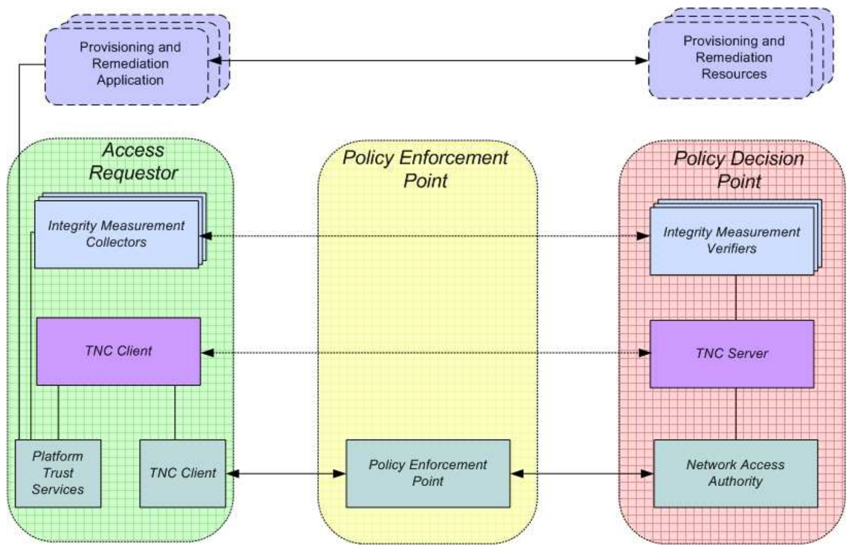

Figure 2.8: TNC Architecture

state exists on a platform prior to being granted network access, requiring that firmware or software patch updates be installed.

Attestation (as discussed in Section 2.4.5) forms an important focus of the TNC specifications. The process of assaying end-point integrity for compliance with policy occurs in three distinct phases: assessment, isolation and remediation.

The assessment phase primarily involves an Access Requester (AR) wishing to gain access to a restricted network (see Figure 2.8). In this phase, the Integrity Measurement Verifier (IMV) on a Policy Decision Point (PDP) examines the integrity metrics coming from the Integrity Measurement Collector (IMC) on the AR's platform and compares them to its network access policies. From this process of reconciliation the PDP informs a Policy Enforcement Point (PEP) of its decision regarding the AR's access request. The PEP is then responsible for enforcing the PDP's decision. As an extension to the assessment phase, in the event that the AR has been authenticated but failed the IMV's integrity-verification procedure, a process of isolation may be instigated whereby the PDP passes instructions to the

PEP which are then passed to the AR directing it to an isolation network. The final phase, remediation, is where the AR on the isolation network obtains the requisite integrity-related updates that will allow it to satisfy the PDP's access policy.

2.7 Operating System and Processor Support

Recently the definition of what constitutes Trusted Computing functionality has been revised and extended to incorporate the concepts of new OS design(s) and hardware-enforced isolation. Hardware-enforced software isolation enables the segregation of security-critical software and data so that it cannot be observed and/or modified in an unauthorised manner by software executing in parallel execution environments. Additionally, the presence of isolated execution environments can ensure that any infection that does occur is contained within the execution environment in which the infection happens.

A Trusted OS design will natively support TPM access as well as providing sandboxed execution environments in which applications can run. In this respect, processor and chipset extensions will provide the hardware support for the creation of these protected environments and act as a basis for enforcing application level sandboxing within main memory.

Isolation technologies have evolved from OS-hosted Virtual Machine Monitors (VMMs) [161] through stand-alone VMMs [57] to para-virtualisation techniques [24]. More recent developments in isolation technology, such as Microsoft's NGSCB OS [102, 103] and the PERSEUS framework [104] running with the L4 μ-Kernel[2], incorporate the concept of an isolation layer that can take advantage of initiatives in the domain of processor and chipset extensions described in Intel's LaGrande [68] and

[2]http://os.inf.tu-dresden.de/L4/

AMD's AMD-V [4] initiatives. This allows a platform to be partitioned into one or more isolated execution environments, typically implemented as one or more Virtual Machines (VMs) each of which may be running an individual OS or a virtualised application.

An isolated execution environment, independent of how it is implemented, should provide the following services to hosted software [103]:

- No interference: Ensures that a program is free from interference from entities outside its execution space.
- Trusted path: Ensures the presence of a trusted path between a program and an input device.
- Secure inter-process communication: Enables one program to communicate with another, without compromising the confidentiality and integrity of its own memory locations.
- Non-observation: Ensures that an executing process and the memory locations it is working upon are free from observation by other processes.

2.8 Summary

In this chapter, we have given an introduction to the basic concepts of Trusted Computing. We discussed the components of Trusted Platforms and provided an overview of the TCG specification set. For the interested reader, introductory texts on Trusted Computing can be found in [88, 100].

Part I

Securing Internet-Based CNP Transactions

Chapter 3

Card Payments

Contents

In this chapter we examine the payment model used for card-based transactions both at the Point of Sale (PoS) and for Card Not Present (CNP) transactions. We look at a number of threats intrinsic to CNP transaction processing and examine a number of protocols used to secure CNP transactions. Furthermore we examine the extent to which these protocols address the identified threats. We conclude this chapter with an overview of the Europay-Visa-Mastercard (EMV) standards for use in PoS environments. This chapter acts as a primer for the remainder of Part I.

3.1 Introduction

In many parts of the world, the use of magnetic stripe cards for Point Of Sale (PoS) transactions is slowly being phased out in favour of Integrated Circuit Cards (ICCs) compliant with the Europay-Visa-Mastercard (EMV) specifications [46, 47, 48, 49]. This process is essentially complete in Europe, well under way in the Far East, and is under active consideration for North America. This major technology change is motivated by the susceptibility of magnetic stripe cards to cloning, and the projected fraud losses associated with the continued use of such cards. ICC cards, having chips that are actively engaged in the transaction authorisation process, are much less easily cloned. The U.K., which began the transition process in 2004, has seen a 47% post-migration reduction in PoS fraud [9]. However, with this reduction in PoS fraud has come an accompanying increase in Internet-based Card Not Present[1] (CNP) fraud. For example, a recent report by the Association for Payment Clearing Services (APACS) on card fraud [8] showed that CNP transactions accounted for 41% of all card fraud perpetrated in 2007 in the UK (a 45% increase from the previous year). This translated into £223.8 million in losses for card issuers and merchants. Specific figures are harder to come by for other countries, but it is not unreasonable to suggest that the U.K. experience is illustrative of the global trend.

The level of fraud seen with CNP transactions has raised significant concern amongst the card processing community. Over the years a number of proposals have been put forward to address this problem. The most noteworthy of these are SET [133] and 3-D Secure [12]. However, neither of these proposals have gained widespread adoption. Indeed, the vast majority of CNP payments are protected today just using SSL/TLS to secure data in transit and to provide a limited form of merchant authentication. Customer authentication is provided through the cus-

[1]For the remainder of this thesis all references to CNP transactions refer to Internet-based CNP transactions.

tomer's ability to provide relevant card details such as the Personal Account Number (PAN) and the corresponding Card Security Code (CSC) over the established SSL/TLS session.

Thus, from the merchant's perspective, there is no guarantee that the customer is actually the legitimate owner of the card corresponding to the details being proffered in a payment transaction. This problem is exacerbated by the perpetual increase in "phishing" attacks [131]: through a combination of social engineering and technical subterfuge, customers may be tricked into revealing their card account details to an attacker. Once revealed, these card details may be used by criminals to instigate illegitimate transactions. In the past, these attacks have required some active participation from the user. However, as attacks become more sophisticated, the human element is becoming removed, with malicious software (malware) residing on a customer's platform being used to capture customer account details and manipulate customer transactions (including possibly instigating new transactions). We use the term Transaction Generator (TG), introduced in [73], to describe such malware in the remainder of this thesis.

On the other hand, from the customer's perspective, there is little, if any, assurance that a merchant will endeavour to protect sensitive cardholder data stored on the merchant's servers. In a PoS environment, a customer's suspicion may be aroused by environmental cues, such as damage to the housing of the payment terminal or the demeanour of the sales assistant. Based on this physical evidence, a customer may decide not to engage in a transaction. However, in an on-line setting, the environmental (browser-based) cues that are available are often either poorly interpreted, or not heeded [41]. In an attempt to instill greater confidence for customers, the Payment Card Industry Data Security Standard (PCI-DSS) [97] has been proposed. This standard details 12 mandatory requirements that merchants and third party processors must satisfy. A customer[2] then trusts that a (legitimate) merchant is in compliance with this standard and has adopted best practice procedures to protect their credit card details. However, PCI-DSS-compliance levels are still low, and there have been instances of companies passing a PCI-DSS conformance audit, only to be later shown to be non-compliant with the standard at the time of a breach [130]. Recently, concerns over merchants running vulnerable payment applications have become so great that beginning in January 2008, Visa began implementing a series of mandates to eliminate the use of non-secure payment applications from the Visa payment system [159]. Visa will now only accept payments from merchants using payment applications that adhere to, and have been validated against, Visa's Payment Application Best Practices (PABP) [160].

In summary, current CNP transaction processing cannot make use of the robust security features available from EMV-compliant ICC cards, and simply reverts to pre-EMV card authentication procedures. This weakness is now being ruthlessly and increasingly exploited by fraudsters, and closing this attack vector represents a significant challenge to the payment card industry.

3.2 Card Payment Processing Model

The processing model used for most card payment systems (including EMV) is typically referred to as the four-corner-model. Within this model, a number of steps are necessary to complete a given transaction (see Figure 3.1). Prior to a customer being able to interact with a merchant, it is necessary that they follow some issuer-specific enrolment procedure in order to obtain a physical payment card. A merchant, likewise, can only accept payments from a customer if they have preregistered to accept payments for that customer's particular card type with their acquirer. The dashed

[2]For the remainder of Part I of this thesis we use terms cardholder, client, user and customer interchangeably.

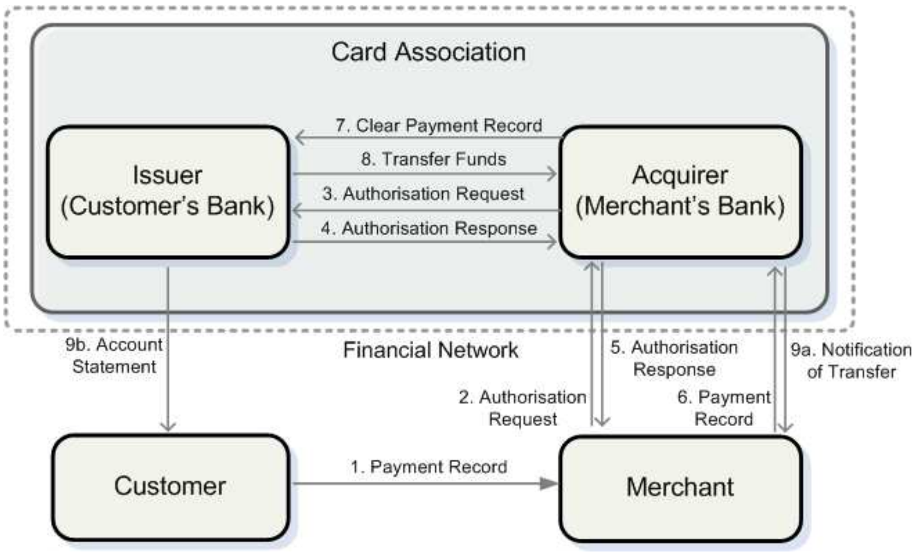

Figure 3.1: Generic model for card processing.

line in Figure 3.1 represents the boundary of the financial network domain. Payment clearing and processing is typically controlled by the card association. Payment clearing occurs as follows:

- **Step 1**: The process begins with a customer (cardholder) signalling their intent to purchase goods by forwarding a payment record to a merchant. The actual characteristics of a payment record differ depending on the environment in which the record is generated. For a PoS purchase, a payment record typically includes the information embossed on the customer's physical payment card, in conjunction with certain merchant supplied information (such as the invoiced amount). In a CNP setting, an additional CSC is included in the payment record. This CSC (also known by Visa as a CVV2 code) is a 3 or 4 digit number that is typically printed on the back of a physical payment card and is used as an additional cardholder authenticator (under the assumption that only the valid cardholder should know this value).
- **Steps 2-5**: These steps occur immediately after the merchant receives the

customer's payment record. In an "on-line" setting a merchant submits the customer's transaction details to its acquirer which will either have the transaction authorised or rejected based on its interactions with the customer's card issuer. After this, the merchant will either confirm payment or inform the customer that their transaction has been rejected. In an "off-line" setting (see Section 3.5) Steps 2-5 may be conducted directly between the card and the terminal and may be approved/declined based on the pre-set risk management routines of both.

- **Steps 6-9**: Based upon the transaction being approved as a result of a successful outcome from Steps 2-5, steps 6-9 represent the account settlement process through which funds are debited from a customer's account and credited to the merchant's.

A significant feature of on-line card-based payment systems is that a positive transaction authorisation (Step 5 in Figure 3.1) does not guarantee payment for a merchant. It is merely an indication that the card account details being proffered have not (yet) been reported stolen and that the customer has sufficient funds to cover the transaction amount. Indeed, unless the card has been reported stolen prior to a transaction, it is difficult for a card issuer, and by extension a merchant, to ascertain whether a particular transaction is fraudulent or not.

In a CNP environment, the merchant trusts (hopes) that the customer is the valid account holder for the presented payment record. This trust is largely underpinned by the level of indemnity offered by card issuers to their customers in the case of lost or stolen cards being used in illegitimate transactions. However, the level of indemnity afforded to merchants is dependent on their adherence to their acquirer-supplied Merchant Operating Guidelines (MOGs). The MOGs lay out the procedures that should be followed when processing CNP transactions. An example of such a procedure might be a requirement to use an Address Verification Service

(AVS) which compares the billing address, as entered by the customer, to that of the card issuer's records. If there is a match, this is seen as an indication that the customer owns the card being used. In many cases a merchant may be held liable for chargebacks associated with a transaction if the merchant does not properly perform cardholder verification. This verification is more difficult to do in a CNP setting, as the merchant is unable to physically inspect the payment card's security features or confirm that the customer is the genuine cardholder via a signature or PIN, as is the case with attended/unattended PoS transactions.

3.3 Threats to CNP Transactions

This section enumerates the threats faced by the customer, the customer's issuer, the merchant and the merchant's acquirer in a CNP payment system. These threats can broadly be generalised into the following categories (ordering does not reflect significance):

- **Loss of Confidentiality and Privacy:** Cardholders might wish to keep their personal account details from being disclosed to unknown third parties. Additionally, cardholders may like to keep multiple instances of personal information (in the form of transactions) from becoming personally identifiable information through transaction profiling. These concerns are manifest for both data-in-transit between the customer and the merchant and for data-at-rest, where account details are stored at the merchant and third party processing facilities. Should customer transaction details be captured, a fraudster will be capable of impersonating the cardholder and instigate new (fraudulent) CNP transactions.

  Privacy of account information at rest has received a lot of press of late with the break-ins at CardSystems Solutions [169] and TJX Incorporated [130]. The break-in at CardSystems Solutions occurred in spite of CardSystems Solutions having passed a conformance audit for the PCI-DSS [169]. It was later shown that CardSystems Solutions were not in compliance with the standard at the time of the breach. This raises an obvious question: how does one audit to ensure compliance as an ongoing concern? This type of rolling audit would be desirable from both a customer and PCI-DSS/PABP standpoint as it would allow verification prior to processing. We will examine this issue in greater detail in Chapter 5.

- **Lack of Authentication and Authorisation:** A customer might wish to authenticate the merchant's identity as being the valid identity of the organisation he is willing to do business with. This is becoming increasingly important as phishing and/or domain name resolution attacks become more prevalent.

  Merchants might also like to authenticate the customer as being the valid cardholder. If a merchant accepts a fraudulent transaction, it can be held liable for the value of the fraud as well as potentially having the cost of its processing rates increased.

  Finally, issuing and acquiring banks might wish to be able to authenticate customers' transaction data so as not to authorise illegitimate payments. That is, they might like to ensure that a valid customer with a valid card instigated and authorised the payment. An issuer can be liable if it authorises a fraudulent transaction.

- **Lack of Integrity:** It is important for all parties that the integrity of the transactional data being sent (and stored) is protected. No party privy to a transaction should be able to modify the details of a transaction.

  In addition to the integrity of transaction data in transit and at rest, it is important that the integrity of a customer's and merchant's software environment be protected. Changes to the software environment should be reported to interested parties so they can make the most appropriate decision as to whether to continue with transaction processing or not.

- **Lack of Non-repudiation:** Non-repudiation is closely allied with customer authorisation. The merchant might wish to have proof that the customer agreed to pay a specified amount for his goods or services. In the event of a dispute, to avoid being held liable and paying the associated chargeback costs, the merchant would need to submit this evidence to the issuer showing that the customer did indeed authorise a transaction.

- **Phishing & Malware:** As we mentioned in Section 3.1, phishing attacks, which fool a customer into revealing sensitive card account details, are becoming increasingly prominent. However, as attacks targeting customers become more sophisticated, new attacks are being deployed that directly target vulnerabilities in the customer's platform. These vulnerabilities, once exploited, allow malicious applications (crimeware) to be surreptitiously installed, leaving the platform prone to data exposure. In this setting, a transaction generator residing on a customer's platform can be used to capture customer account details and manipulate customer transactions (including possibly instigating new transactions).

  A crimeware attack, or more generally a malware attack, must typically pass through three stages to fulfil its goal [157]. These are *distribution*, *infection* and *execution*.

  - **Distribution**: Distribution refers to the means by which malware arrives at a platform. Traditionally, malware has been heavily reliant on social engineering as a means of distribution. Here a user is tricked into opening an e-mail or instant message attachment containing malware, or indeed, installing malware which has been integrated into an apparently useful application.

- **Infection**: Infection is the process by which malware penetrates a platform. Malware may be ephemeral and leave behind no lasting executable, as is the case with system reconfiguration attacks, such as DNS poisoning used in pharming. Alternatively, malware may persistently reside in memory or be executed upon loading an infected component. The infection may target user-space objects, as is the case with malicious browser help objects and application programs, or kernel objects, as is the case with malicious device drivers.

  In order to disguise an infection, rootkits are sometimes deployed. Basic rootkits simply replace user-level executables with trojanised versions, for example, replacing the Linux Process Status (PS) command with a version that incorrectly reports running processes. However, such attacks are trivially detectable given current anti-virus technology. Unfortunately, more recent rootkits such as variants of the FU rootkit [52] or new rootkits that exploit hardware-based virtualisation, such as Microsoft's proof of concept Subvirt rootkit [80], are becoming adept at circumventing antiviral defences. Subvirt attempts to modify a system's boot sequence so that the legacy Operating System (OS) is loaded into a virtual machine. This allows any system-call made by the OS to be observed and modified by the Subvirt application.

- **Execution**: It is during this stage that the malicious objectives of the malware are realised. The malware may attempt to gain unauthorised access to information, capture user-entered details or steal proprietary data. For example, the Bankash.G trojan horse attempts to steal user account details such as user names and passwords and credit card information from a compromised computer [141]. This data is collated by the crimeware and transmitted back to the attacker for processing.

Recent studies that examine the evolution, proliferation and propagation of both phishing and crimeware illustrate a marked and steady rise in both the number and complexity of applications used in the commission of cyber-crime [7, 142]. Given the (predicted) prevalence of phishing and crimeware targeting both end-user and their platforms, any proposed solutions for securing CNP transactions will need to consider both as a serious and viable threat.

3.4 Protocols for Protecting CNP Transactions

The proliferation of Internet-based commerce (and the increasing level of fraud associated with it) has resulted in a great deal of effort in developing methods for securing these transactions. In this section we discuss SSL, 3-D Secure and SET and examine how well they address the threats identified in Section 3.3.

3.4.1 SSL/TLS

SSL/TLS [42] is a protocol suite running on top of the Transmission Control Protocol (TCP) and provides a reliable, end-to-end secure communications service. Within this protocol, entities engage in a SSL/TLS Handshake Protocol to establish keying material and perform authentication. This keying material is then used in the SSL/TLS Record Layer Protocol for data integrity protection and encryption. SSL/TLS supports a wide variety of key exchange methods, including both anonymous and ephemeral Diffie-Hellman exchanges, supported by digital signatures, as well as key establishment based on RSA public key encryption. Both unilateral (server to client) and mutual authentication are supported.

SSL/TLS has become the *de facto* standard for the secure transmission of CNP transaction information such as PAN, CSC and relevant billing information. However, this use of SSL/TLS can be seen as pragmatic use of an existing technology rather than a systematic approach to securing CNP transactions.

One significant advantage of SSL/TLS, and perhaps the main reason for its pervasive deployment, is that it does not mandate customer certification. However, if we examine SSL/TLS under the headings of Section 3.3, we note a number of deficiencies in using SSL/TLS to secure CNP payments:

- **Confidentiality and Privacy**

  The confidentiality and privacy afforded by SSL/TLS only protects against an adversary eavesdropping on a transaction between a customer and a merchant. What occurs outside of an established transfer session is not within the scope of SSL/TLS's protection remit.

  SSL/TLS provides confidentiality by encrypting and integrity protecting data with an ephemeral shared key that is established by both the customer and the merchant during the handshake protocol. Successful attacks targeting either end-point allow an adversary access to CNP transaction data.

- **Authentication, Authorisation and Non-repudiation**

  Potentially the biggest deficiency in the use of SSL/TLS for CNP payments is the lack of customer authentication. Although SSL/TLS has a provision for client authentication using digital signatures supported by digital certificates, it is seldom, if ever, used due to increased costs and perceived difficulties in enrolling customers in the required PKI.

  Concerning authorisation, since the confidentiality/integrity mechanisms provided by SSL/TLS are session-specific, it is questionable if a merchant could definitively prove that the customer authorised a transaction. In fact, since the record layer protocol session uses symmetric key techniques to protect transaction messages, an unscrupulous merchant could fake an entire transaction session. In the event of a dispute, an arbitrator would be unable to determine which party generated the transaction; consequently, non-repudiation services cannot be obtained from the use of SSL/TLS.

- **Integrity**

  The communication integrity afforded by SSL/TLS only protects against attacks by parties not immediately privy to the current transaction. It says nothing as to the integrity of the data on either end-point in the connection or, for that matter, the security controls in place at either the merchant's or the customer's platform.

- **Phishing & Malware**

  Browser-based cues, such as warnings about merchant certificates, or the absence of a closed padlock indicating an SSL/TLS secured site, are often either poorly interpreted, or not heeded by customers [41]. This fact is exploited in phishing scams which rely on customers' ignorance and inability to distinguish real sites from fake ones. As a result customers continue to divulge credit card details to phishing sites in spite of the protections afforded by SSL/TLS.

  In addition to phishing sites, SSL/TLS is vulnerable to crimeware targeting both customer's and merchant's platforms. Crimeware may record CNP transaction details and/or surreptitiously instigate new transactions without a customer's consent [73]. Furthermore, the use of client authentication would do little to limit this threat. Without a means of securing private keys on platforms, a TG can simply use a client's private key to authenticate itself prior to transaction processing.

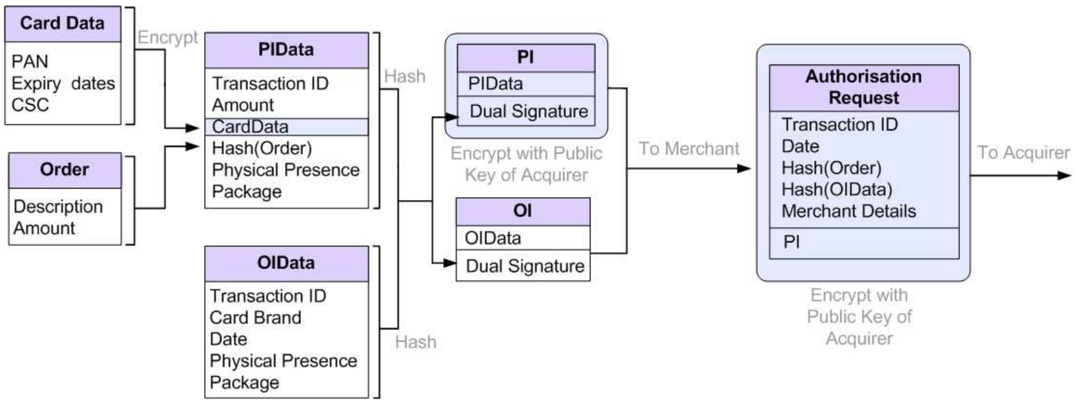

Figure 3.2: SET Message Structure

3.4.2 SET

SET [133] differs from SSL/TLS in that it was designed explicitly as a complete payment protocol and addresses a number of the deficiencies found in the SSL/TLS-based approach for facilitating on-line card-based commerce. SET allows every entity that is party to a transaction to be authenticated by employing a certificate-based PKI in which all participants are required to enrol. Certificates are exchanged between transacting peers to enable mutual authentication. In making a purchase using SET, a purchase order message is constructed by a customer so that only the merchant can see the Order Information (OI) and only the payment gateway can see the Payment Information (PI). This is accomplished though what is termed a "dual signature", whereby messages intended for the merchant and messages intended for the payment gateway can be linked without simultaneously having to reveal both. In the SET protocol, the PI comprises transaction related data as well as a transaction ID carried in a "digital envelope", see Figure 3.2. A one time session key is created for bulk encryption; this protects the PI, the dual signature and a hash of the OI. This key is then enciphered with the public key of the payment gateway. Only the gateway can decrypt the envelope and obtain the key to decrypt the enciphered PI. For a more through treatment of SET see [94, p.100-123].

With regards to the threats listed in Section 3.3, SET fares rather favourably in comparison to SSL/TLS.

- **Confidentiality and Privacy**

  Confidentiality is provided through the use of both symmetric and asymmetric cryptography. A purchase order message is constructed by the customer in such a way that only the merchant can see the OI and only the payment gateway can see the PI. This protects data-in-transit between the customer and the financial network and data-at-rest at the merchant, as the merchant never gets to see the account details of the customer. However, as is the case with SSL/TLS, transaction details are still vulnerable on the client platform, as are client private keys.

- **Authentication and Authorisation**

  SET allows for every entity that is party to the transaction to be authenticated using digital signatures and public key certificates. In order for a transaction to be authorised, payment needs to be approved by both the customer and his bank. The merchant forwards the encrypted PI information to the payment gateway along with its own authorisation information. This information comprises, amongst other things, an authorisation block containing a transaction ID encrypted with a session key and signed with the merchant's private key. This allows the payment gateway to verify both parties by their respective signatures. Also, the gateway confirms that they are referring to the same transaction by comparing transaction ID values.

- **Integrity**

  Message integrity in the SET protocol is achieved through the use of digital signatures. Messages are signed using the private key of the transacting peer. Signatures are verified using the public key obtained from the signing entity's certificate.

- **Non-repudiation**

  SET allows for non-repudiation of transactions — assuming the transaction gets authorised by the issuing bank. However, it remains questionable how much weight can be given to non-repudiable evidence produced during a run of the SET protocol, given the malware threat.

- **Phishing & Malware**

  In comparison to SSL/TLS, SET fares favourably in terms of its resiliency to phishing attacks. A faked merchant site is unable to extract customer account details from a customer-generated PI. However, like SSL/TLS, SET remains vulnerable to crimeware executing on the customer's platform. Without adequate protection for a customer's private key, a Transaction Generator application might be able to surreptitiously generate spurious transactions.

Despite improvements over an SSL/TLS-based approach, SET is no longer being deployed for use in CNP transactions. A number of theories have been put forward to explain why SET never became a success. These range from ease of use to the cost and difficulty of maintaining a stable PKI [64].

3.4.3 3-D Secure

3-D Secure is an optional adjunct to the SSL/TLS-based approach presented in Section 3.4.1, and attempts to provide cardholder authorisation for CNP transactions by requiring customers to authenticate themselves prior to transaction processing. As all communication in 3-D Secure is reliant on SSL/TLS for its security, we dispense with the analysis of 3-D Secure in terms of the threats presented in Section 3.3. Instead we provide an overview of how 3-D Secure's features can mitigate certain shortcomings identified with SSL/TLS usage in CNP transactions.

Before a 3-D Secure payment can be processed, a customer must preregister his account with his card issuer. During the registration procedure a cardholder typically chooses a secret password that will be used to authorise subsequent CNP transactions[3]. This authorisation forms an ancillary step in regular merchant checkout processing where, after receiving a customer's PAN and CSC, a merchant site constructs a Payer Authentication Request (PAReq) message and redirects the customer to a 3-D Secure Access Control Server (ACS). The customer next authenticates himself to this ACS using his username and password, as established in the registration phase. Based upon the correctness of the supplied username/password combination, the ACS formulates a Payer Authentication Response (PARes), confirming whether or not the customer has been authenticated, and signs it. This signed PARes message is then passed via the customer's browser and to the merchant plug-in. The merchant plug-in then verifies the ACS signature and decides if it wishes to proceed with the transaction. A validated response can later be used as evidence to show that the customer authorised a particular payment. If the customer account number (PAN) is not registered with any ACS, a Visa directory server informs the merchant plug-in and normal MOTO-based authorisation procedures are attempted. This approach aims to tackle the fraudulent acquisition of card account details for use in CNP transactions by providing a delineation between card authentication data and customer authentication data.

3-D Secure's real benefit comes in reducing the economies of scale possible with PoS card skimming attacks. An attacker obtaining a customer's card details, possibly by means of a compromised PoS terminal, will no longer be able to complete a fraudulent CNP purchase using the obtained information, as a PAN/CSC would no longer be sufficient to authorise a CNP transaction. Unfortunately, in this instance the use of an additional static authenticator may prove only a minor barrier to obtaining card account details. The use of an external, hardware generated one-

[3]Recently trials have begun that replace the use of a cardholder selected password with a one-time-password generated by a hardware token issued to the cardholder [85]

time-password can significantly help reduce the impact of a phishing attack by only allowing a single new transaction to be instigated (should the criminal trick the cardholder into disclosing a password). However, the presence of crimeware on a customer's platform or a phishing site that purports to provide a 3-D secure plug-in capability could potentially dupe cardholders into revealing additional authentication data.

3.5 EMV

In this section we describe the use of EMV (chip and PIN cards) in PoS transactions. EMV, as described in the current iteration of the specifications [46, 47, 48, 49], defines the full spectrum of interactions, from the physical to the logical, between an ICC and an integrated circuit enabled PoS terminal. EMV supports both cardholder authentication through new Cardholder Verification Methods (CVMs) and ICC authentication through either Static Data Authentication (SDA), Dynamic Data Authentication (DDA) or Combined DDA and application cryptogram generation (CDA). The introduction of new CVMs and SDA/DAA/CDA functionality aims to reduce credit and debit card fraud through improved authentication and authorisation mechanisms. Additionally, depending on the results of card and terminal risk management routines, which are designed to protect issuers and acquirers respectively, transactions can be completed either on-line or off-line. An on-line transaction is one in which the issuer is contacted to approve a transaction whilst an off-line transaction is one that is completed locally between the card and the terminal based on a reconciliation of issuer and acquirer risk routines as stored in the card and terminal respectively.

A readable introduction to the technical content of the EMV specifications can be found in [110]. This section aims to provide a brief overview of the current EMV specifications, particularly in relation to transaction processing and EMV key usage. The purpose of this section is to act as a primer for Chapter 5 in which we examine emulating EMV on a Trusted Platform for use with CNP transactions.

3.5.1 Transaction processing

The processing of an EMV transaction begins when the card is inserted into a PoS terminal and involves a number of distinct stages. The stages, as presented here, do not follow a linear path. Some of them operate under a "not before but not after" semantic and some of the steps, where indicated, are optional.

1. **Application Selection:** The terminal issues a SELECT command [46] to choose the appropriate EMV application from a multi-application card. The selection procedure can be either direct or indirect and is based on the same Application Identifiers (AIDs) being supported by both the terminal and the ICC.

2. **Initiate Application Processing:** The terminal commences application processing by issuing a GET PROCESSING OPTIONS command [48] to the card. In response to this the card returns the Application File Locator (AFL) and the Application Interchange Profile (AIP). The AFL provides a "table of contents"-like structure of the publicly available information provided by the card application. The AIP indicates the application functions that are supported by the card such as whether off-line SDA/DDA/CDA are available or if card holder verification is supported.

3. **Read Application Data:** A number of READ RECORD commands [48] are issued by the PoS terminal to the card to read the necessary data for the transaction to proceed. As part of this step, the terminal makes sure all

mandatory data elements are present on the card.

4. **Processing Restrictions:** This is a mandatory step performed by the terminal and does not require any interaction with the ICC. The purpose of this step is to ascertain the level of compatibility between the application running on the card and the application running on the terminal [48]. Based on this comparison, the terminal takes an appropriate action — including possibly terminating the transaction.

5. **Off-line Data Authentication:** This is an optional step in which the terminal issues an INTERNAL AUTHENTICATE command [48] to select one of SDA, DDA or CDA based on terminal and card capabilities.

6. **Card holder Verification:** This optional step [48] allows the terminal to potentially verify the authenticity of the cardholder in relation to the card based on a particular CVM. It is also used by the cardholder as a means to authorise a particular transaction. CVM methods range from no verification required to both off-line and on-line PIN verification.

7. **Terminal Risk Management and Action Analysis:** The purpose of terminal risk management and terminal action analysis is to protect the issuer, acquirer and payment system from fraud [48]. Based on a variety of measures, such as floor limits, random transaction selection, or velocity checking, a transaction may be forced to complete on-line.

   Floor limits ensure that if a transaction (including attempted split transactions) exceeds a certain amount, the transaction should be verified on-line. Random transaction selection uses a biased random selection to ensure that low value transactions are authorised on-line from time to time. Finally velocity checking forces an on-line check after a certain number of off-line checks as determined by an issuer-controlled parameter, the Lower Consecutive Off-line Limit.

8. **Card Risk Management and Action Analysis:** Details of card risk management are proprietary to card issuers and are outside the scope of the EMV specification. However, as a result of card action analysis [48] (which is based on the results of the card risk management routines), an ICC may decide to complete a transaction by either going on-line, remaining off-line, requesting a referral or rejecting the transaction outright.

9. **On-line Authorisation:** During normal transaction processing the terminal may decide to proceed with an on-line check. This could be as a result of a number of factors. The terminal may only support on-line transaction processing or the store attendant may have found the customer suspicious. It could also be as a result of the outcome of either of Steps 7 or 8 above. In any case, a transaction is deemed outside the risk profile for off-line completion and requires further scrutiny by the customer's card issuer. An Application Cryptogram (AC, see Section 3.5.2.1) is generated by the card and forwarded to the card issuer which, after examination, determines if the transaction should go ahead or not. The card issuer returns another AC informing the card whether or not to proceed with the current transaction.

10. **Issuer-to-Card Script Processing:** In order to allow card updates in the field, the card issuer can return scripts via the terminal to the ICC for processing. These scripts are not necessarily relevant to the current transaction and may be used to update applications during the utilisation phase of the ICC's lifecycle, or to transition the card into a blocked or unblocked state.

3.5.1.1 Cardholder Verification

The EMV standard specifies a number of CVMs including: no CVM required; signature-based CVM, and on-line or off-line PIN verification. In supporting off-line PIN verification, the ICC maintains a local copy of the cardholder's PIN and an

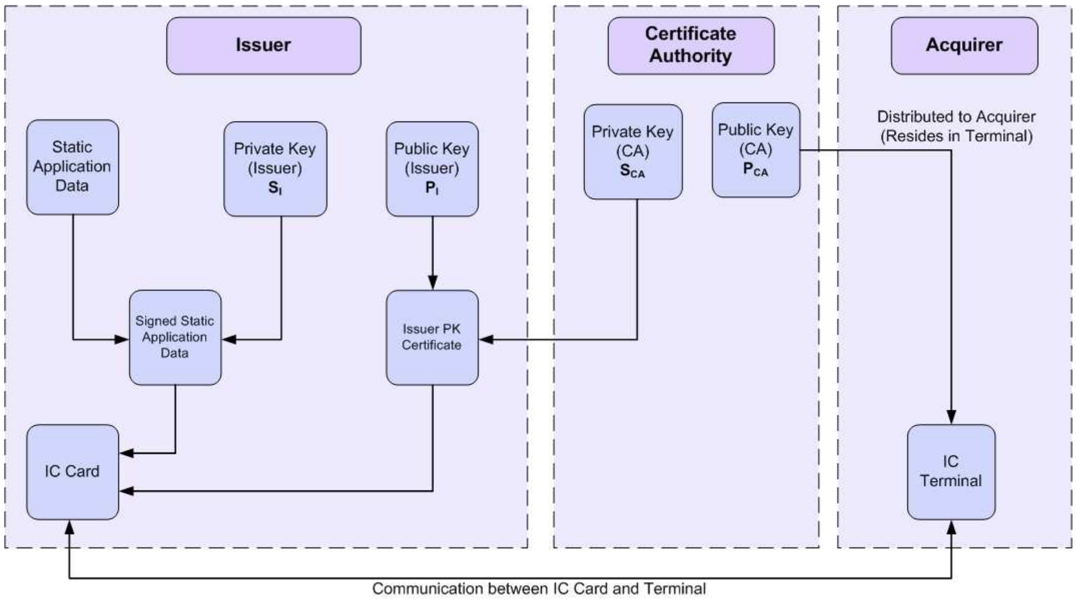
Issuer
Certificate Authority
Acquirer
Static Application Data
Private Key (Issuer) S_I
Public Key (Issuer) P_I
Private Key (CA) S_{CA}
Public Key (CA) P_{CA}
Distributed to Acquirer (Resides in Terminal)
Signed Static Application Data
Issuer PK Certificate
IC Card
IC Terminal
Communication between IC Card and Terminal

Figure 3.3: EMV - Static Data Authentication.

asymmetric key pair for PIN encryption. Typically, this PIN encryption/decryption key pair will be the same as the ICC key pair (see Section 3.5.2). A PIN Entry Device (PED) may use the public portion of this key pair (depending on terminal's support for either plain-text or encrypted PIN verification) to encrypt the PIN entered by the customer. The PED forwards the newly encrypted PIN to the ICC for comparison against the stored PIN of the customer. Based on the outcome of this comparison (success/fail), the ICC determines if it wishes to proceed with the transaction or not, based on card action analysis routines.

3.5.2 EMV Key Usage

The number and type of keys in an EMV-compliant ICC depends on whether the card is SDA or DDA/CDA compliant for off-line authentication. In the case of SDA only, the ICC maintains the public key of the issuer (in a certificate signed by a card association CA) as well as a signature of its static application data created using the

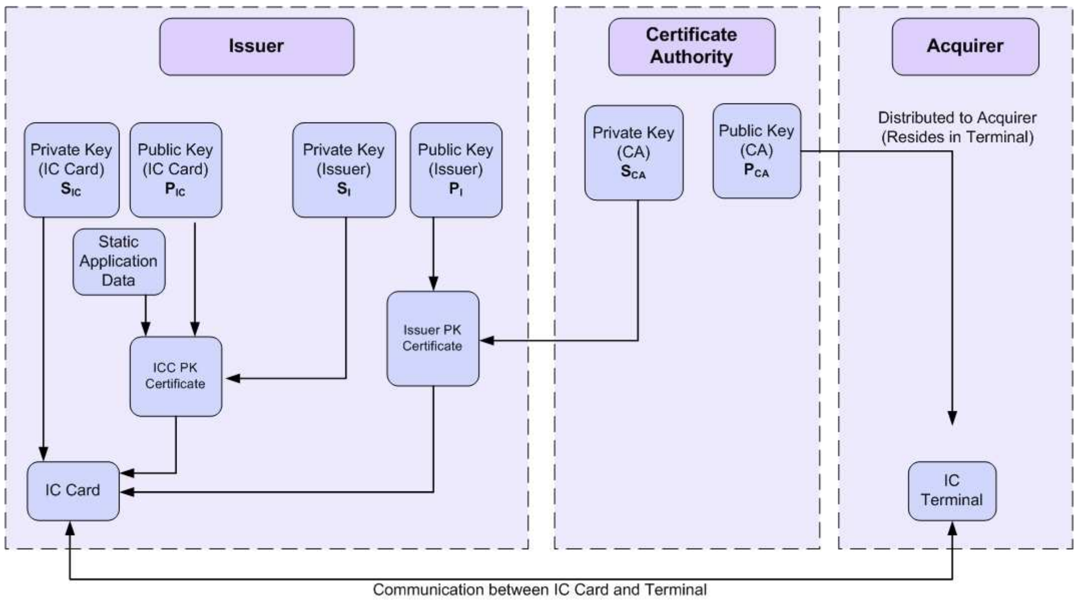
Issuer
Certificate Authority
Acquirer
Private Key (IC Card) S_{IC}
Public Key (IC Card) P_{IC}
Private Key (Issuer) S_I
Public Key (Issuer) P_I
Private Key (CA) S_{CA}
Public Key (CA) P_{CA}
Distributed to Acquirer (Resides in Terminal)
Static Application Data
ICC PK Certificate
Issuer PK Certificate
IC Card
IC Terminal
Communication between IC Card and Terminal

Figure 3.4: EMV - Dynamic Data Authentication.

card issuer's private key, see Figure 3.3. The terminal, in turn, stores the public key of the card association CA in order that it can verify the signed static application data.

DDA is similar except that, in addition to the SDA keys, the ICC has its own key pair (the ICC key pair), of which the public part is stored in an ICC certificate. This certificate also contains the card's static application data and is signed with the private key of the card issuer, see Figure 3.4. Like SDA, the terminal uses its embedded CA public key to verify the issuer's public key certificate. This newly verified issuer's public key is then used to verify the ICC certificate.

In DDA, the ICC private key is used to generate a signature over the static application data as well as a random number provided by the terminal. This challenge-response procedure allows the terminal to determine whether the card is genuine or not as well as to ascertain if the data personalised in the card has been altered since card personalisation, as per SDA. CDA-enabled cards are similar to DDA cards except that the data over which a signature is generated includes an additional AC (see Section 3.5.2.1).

3.5.2.1 Application Cryptograms

For SDA/DDA/CDA conformant EMV ICCs, ACs are used to protect transaction messages generated by the ICC using AC session keys. ACs are generated by the ICC and card issuers using Message Authentication Codes (MACs) based on a shared secret. The ICC and its issuer share a long-term MAC master key, the ICC Application Cryptogram Master Key, MK$_{AC}$. An Application Cryptogram Session Key, SK$_{AC}$, is derived from this master key using a session key derivation function (as defined in Annex A1.3 of [47]) that uses a card's Application Transaction Counter (ATC) as diversification data. An ATC is a counter that keeps track of the total number of transactions performed by a card. In this way the key SK$_{AC}$ will be different for every AC generation.

To avoid the issuer having to store a unique master key for every card the master key is itself a result of a derivation process. The issuer uses the ICC's PAN, the PAN sequence number, as well as its Issuer Master Key (IMK) to dynamically generate the shared MAC master key, MK$_{AC}$.

EMV defines a number of different types of transaction messages. Depending on the reconciliation of card and terminal risk management routines one of the following will be generated by either the card or the cardholder's issuer: an Application Authentication Cryptogram (AAC), an Authorisation Request Cryptogram (ARQC), an Authorisation Response Cryptogram (ARPC), or a Transaction Certificate (TC). If the transaction is approved, the ICC will generate a TC which will be passed to the terminal and can be used to claim payment during the clearing process. If a transaction is declined, the ICC will generate an AAC. In the event that the transaction needs to be approved on-line the ICC generates either an AAC or an ARQC, which will be forwarded to the issuer. The issuer will then reply with an ARPC indicating whether the transaction should be approved or declined, in which case a TC or an AAC will be generated by the ICC.

3.6 Summary

In this chapter we have looked at the general payment model employed for securing card-based payments. We identified a number of threats to CNP payments and examined three of the main protocols used to secure CNP transactions. We note that, given current advances in attack sophistication and given the (predicted) future prevalence of crimeware, attackers may circumvent the protections afforded by SSL/TLS, SET and 3-D Secure by directly targeting customers' platforms. On the other hand, we have seen that EMV offers a number of attractive security features for PoS transactions. However, EMV was not designed for use in CNP transactions and currently there is no easy way of making its security services amenable to CNP transactions. We will look at a number of proposals for bringing the benefits of EMV to CNP transactions in greater detail in Chapter 5.

Chapter 4

Trusted Computing and CNP Transactions

Contents

In this chapter we demonstrate how Trusted Computing technology can be used to enhance the security of Internet-based Card Not Present (CNP) transactions. We take a pragmatic approach, focusing here on exploiting features of Trusted Computing as it is being deployed today. Thus we rely only on the presence of client-side

Trusted Platform Modules, rather than upon the "idealised" deployment in which Trusted Computing functionality is fully integrated with OS and CPU. In essence, our approach uses features of the Public Key Infrastructure (PKI) that is inherent in Trusted Computing to build lightweight client-side enrolment and certification processes; public key certificates are then used to underpin authentication for CNP payments. Using this approach we demonstrate how Trusted Platform Module (TPM) enabled platforms can integrate with SSL/TLS and 3-D Secure. We discuss the threats to CNP transactions that remain even with our enhancements in place, focussing in particular on the threat posed by malware, and how it can be ameliorated.

4.1 Introduction

This chapter examines how Trusted Computing can be used to enhance the security of existing protocols (SSL and 3-D Secure) in the provision of secure CNP transactions. We operate from the sole assumption that client platforms are equipped with Trusted Platform Modules (TPMs) having limited but trusted cryptographic functionality. We use the TPM's trusted capabilities to build lightweight client-side enrolment and certification processes. These effectively bind a platform, and by extension its owner, to a particular card. The resulting public key certificates and TPM signing capabilities are then used to underpin authentication for CNP payments.

One of the most salient issues in the development of our approach is the problem of customer enrolment, during which a customer/card binding is established. We examine different system architectures and discuss the pros and cons of their associated enrolment procedures. Another major issue is the increasing threat of malware (or, more specifically, crimeware) and its ability to create spurious transactions or modify on-going ones. We also study the malware/crimeware threat, explaining how it can be reasonably addressed within our architecture using the secure attention sequences (physical presence) that are a mandatory part of Trusted Computing (see Chapter 2).

An oft-cited reason for the failure of SET was its reliance on PKI and client-side certificates. Our approach would appear to suffer from the same problems. Given the rise of Internet-based CNP transactions and the associated fraud levels, we believe that the economic conditions are now far more ripe for the deployment of solutions based on client-side certification. Moreover, we are able to take advantage of the TPM certification infrastructure to support Trusted Computing. This PKI needs only to be augmented with certain CA functionality, provided by a card issuer in our approach.

We stress here that we do not need to make any further assumptions about the deployment of Trusted Computing in order for the approach in this chapter to provide useful security benefits for CNP transactions. In particular, we need not rely on a full-blown deployment of Trusted Computing in which the TPM is fully integrated with the host platform's processor, boot process and trusted Operating System. The next chapter studies how such a deployment can be exploited to enhance the security of CNP payments.

This chapter is structured as follows. In Section 4.2, we look at some related work. In Section 4.3, we examine the issue of customer enrolment with particular emphasis on the establishment of customer-centric credentials within a TPM-enabled platform. In Section 4.4, we discuss the issues of client-side certification in the face of the increasing threat of malware. In Section 4.5, we examine the role TPM-enabled customer credentials can play in supplying additional security to SSL and 3-D Secure. Finally, we conclude with Section 4.6.

4.2 Related Work

The idea of using Trusted Computing to enable client-side certification has previously been discussed in [2, 16, 145] as well as in the as-yet-unpublished Trusted Computing Group's TLS extensions for carrying attestations. However, none of this work takes into consideration the threat posed from malware nor the infrastructural requirements necessary to support client-side certification. The threat from malware is examined in greater detail in [53, 58, 72, 73]. There, Virtual Machines (VMs) are used to constrain the use of malware to an individual VM "compartment". The authors of [53, 72, 73] also suggest using visual cues as to the trustworthiness of the VM with which an end-user interacts.

Other related work includes the use of Trusted Computing as an adjunct to securing connected card readers for generating digital signatures, presented in [14] and [135]. However, both approaches suffer from costs associated with the provision of card readers to end users. Additionally, both proposals assume the presence of trusted software to interact with the readers.

Recent works that takes a pragmatic approach to Trusted Computing, like we do in this chapter, includes [84] and [123]. In [84], McCune *et al.* attempt to address the problem of user-level malware in the absence of Trusted Computing processor and chipset support. They propose the use of an external trusted mobile device to establish an encrypted and authenticated channel between the user and a TPM host. In [123], Sarmenta *et al.* implement virtual monotonic counters to prevent replay attacks on an untrusted machine aided solely by TPM support. They suggest a number of possible applications of their approach, such as n-time use keys and n-copy migratable objects.

Finally, the use of additional hardware tokens for securing on-line commerce has been proposed in [165]. Here an additional universal serial bus token is deployed which directly interfaces with a customer's bank. The token monitors connections from the customer's PC and asks the customer for confirmation before committing to a transaction. This approach is functionally similar to ours, except that we look to use common cryptographic hardware that is becoming increasingly prevalent on modern computer systems to achieve our solution.

4.3 Architectures for Customer Enrolment

In this section we will look at the issue of customer enrolment with a view to obtaining certification of a TPM-controlled (non-migratable or certified migratable) key. We present a number of different system architectures through which enrolment may occur and discuss the issues of client-side certification in the face of the omnipresent threat of malware.

4.3.1 Enrolment

This section aims to explore various architectural options for enrolling a platform, and by extension its owner (cardholder), using a card-issuer-controlled Trusted Computing CA. The goal here is for a cardholder to obtain an X.509 certificate incorporating both card account details as well as a cardholder's public key, with the corresponding private key being bound to the cardholder's TPM. This certification by the card issuer will effectively bind a cardholder's hardware platform to a particular card. The cardholder can later demonstrate this binding when authenticating himself to a merchant during a CNP transaction. Thus the TPM acts as both a secure storage area for the cardholder's private key as well as providing a means by which the use of the private key can be controlled.

In order for a card issuer to provide an enrolment facility for their customers' platforms, it will be necessary for the card issuer to provide some form of CA functionality. Given the strictures of the TCG specifications, this functionality can take the form of a Privacy CA, an SKAE CA or possibly a hybrid of the two. As we saw in Section 2.4.5, in order for a platform to obtain an X.509 certificate for a TPM resident key, it is necessary to go through a number of steps. A platform at the behest of its owner (the cardholder) first makes a request to a Privacy CA to certify an AIK public key. The corresponding AIK private key is then used to sign the public key of a non-migratable (or certifiable migratable) TPM key pair. This signed public key is then sent to an SKAE CA which certifies that the private portion satisfies certain key type and attribute designation constraints (as evidenced in the TPM_Certify_Info structure) before issuing an X.509 certificate for the customer's public key.

4.3.2 Deciding on an Architecture

Figure 4.1 shows the general certificate enrolment hierarchy in which customers can enrol with multiple card issuers, who in turn can enrol with multiple card associations. The cardholders themselves have no direct dealing with the card association but instead interact with the enrolment interfaces exposed by their card issuers. In defining these interfaces there are various design decisions related to a card issuer providing Privacy/SKAE CA functionality. These can be broken down as follows:

Privacy CA: By acting as a Privacy CA, a card issuer can issue AIK certificates to its customers' TPM-enabled platforms. Unfortunately, the usefulness of this approach is limited by the fact that an AIK private key can only used to sign integrity metrics and non-migratable/Certified Migratable keys.

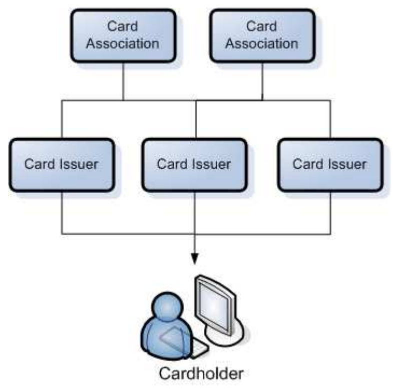

Figure 4.1: Certificate Enrolment Hierarchy

Note that in discussing the use of a Privacy CA here, we are not suggesting that our approach should actually benefit from the privacy-enhancing features available from this choice. Rather, we see the card-issuer playing the role of a Privacy CA as providing a convenient mechanism for achieving client-side authentication within the limitations of the TCG specifications. Indeed, there is a potential privacy concern for customers in the disclosure of a platform's EK public component to a non-manufacturing entity: since an EK is unique per platform instance, it may act as a 'super-cookie' in identifying subsequent platform actions across multiple domains.

SKAE CA: By acting as an SKAE CA, a card issuer can issue X.509 certificates on non-migratable/Certified Migratable keys to customers' TPM-enabled platforms. Once this certificate is received it can be used in future transactions, either during an SSL handshake (see Section 3.4.1) or in support of a 3-D Secure authentication (see Section 3.4.3). In providing this service, a customer's card issuer does not need to provide Privacy CA functionality. This can be provided by an entity that is in the best position to do so, typically a TPM manufacturer. However, a card issuer would need to trust the outcome of the Privacy CA's AIK credential issuance procedure that precedes a customer's SKAE application. This solution can be seen to offer additional anonymity to a customer's platform, as it breaks the link between an EK and

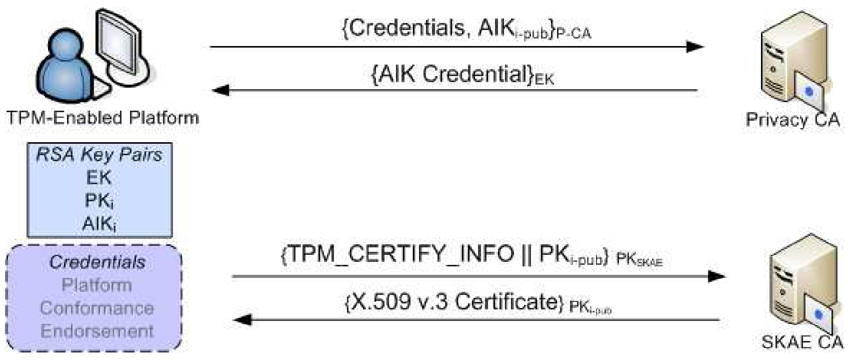
{Credentials, AIK$_{i\text{-pub}}$}$_{P\text{-}CA}$
{AIK Credential}$_{EK}$
TPM-Enabled Platform
Privacy CA
RSA Key Pairs
EK
PK$_i$
AIK$_i$
Credentials
Platform
Conformance
Endorsement
{TPM_CERTIFY_INFO || PK$_{i\text{-pub}}$} PK$_{SKAE}$
{X.509 v.3 Certificate} PK$_{i\text{-pub}}$
SKAE CA

Figure 4.2: Card Issuer Controlled Privacy/SKAE CA

an AIK by having a Privacy CA (outside of the bank's domain) handle this mapping.

Hybrid CA: The final option is to have a card issuer act as a dual Privacy/SKAE CA. This is perhaps the most pragmatic solution for customer enrolment as it avoids the assumption that Privacy CAs are widely available. It also has the added benefit of shortening the customer enrolment procedure. Instead of making two separate CA requests, a customer generates an AIK key pair and a non-migratable/Certified Migratable key pair and signs the public component of the non-migratable/Certified Migratable key using his private AIK key. The AIK/SKAE certificate request package is then bundled and sent to the Hybrid CA, which processes each component individually before issuing an AIK credential for the AIK public key and an X.509 certificate for the non-migratable/CMK public key.

Figure 4.2 shows the most general case where a Privacy CA and SKAE CA are distinct entities. Obtaining a X.509 certificate for a TPM-bound non-migratable key in our system is a result of the following process:

1. The cardholder instructs his TPM to create an AIK key pair, AIK_{i-pub} and AIK_{i-priv} for the public and private components respectively.

2. The cardholder instructs his TPM to generate a certificate request package for

his card issuer's Privacy CA in order to obtain an AIK credential for his newly generated AIK key, AIK_{i-pub}.

3. The Privacy CA validates the cardholder's request and issues an AIK Credential to the cardholder's TPM.

4. The cardholder's TPM receives the AIK Credential and the cardholder instructs his TPM to activate his AIK, AIK_{i-priv}.

5. The cardholder instructs his TPM to generate a key pair K_{i-pub} and K_{i-priv} with K_{i-priv} having the following properties: its type should be non-migratable (or certified migratable), its attribute designation should be signing only, and the use of the key should always require authorisation[1] which the cardholder now supplies, in the form of a passphrase.

6. The cardholder instructs his TPM to certify (sign) K_{i-pub} generated in Step 5 using AIK_{i-priv} generated in Step 1. This creates a signed TPM_Certify_Info structure [150] describing the security properties K_{i-priv} from Step 5.

7. The cardholder instructs his TPM to create an SKAE extension. This extension acts as a receptacle for a TPM_Certify_Info structure from the preceding step.

8. The cardholder instructs his platform to create a certificate request package incorporating the SKAE extension from the previous step. During this process the cardholder authenticates himself to his card-issuer-controlled SKAE CA. This authentication would involve demonstrating knowledge of the payment card's PAN, CSC, address as well as a secret Personal Identification Number (PIN) or password[2].

[1]We assume that such authorisation data can be observed by malware. Its inclusion is to mitigate against the risk of this key later being misused in the event of a platform being stolen.

[2]We assume a secret PIN or password would be provided to cardholders using an out-of-band mechanism, similar to that currently used in Internet banking.

9. If the card issuer SKAE CA is satisfied with the above information, then the SKAE CA issues an X.509 v.3 certificate containing a customer's PAN (within the subject alternative name extension) with an additional SKAE extension incorporating the K_{i-pub} of the non-migratable key pair generated in Step 5. The inclusion of the PAN in the certificate provides a mechanism through which a card can be demonstrably bound to a platform and by extension the platform's owner (cardholder). The omission of the CSC from the certificate removes certain security issues with respect to CNP processing. Without a CSC/PAN combination, an adversary cannot engage in traditional MOTO-based payment authorisation. Thus the absence of the CSC from the X.509 certificate significantly reduces the value of the PAN to an adversary. Finally, including a validity period can further constrain a card's usage, as is common in physical deployments.

10. The cardholder's platform receives the certificate from the cardholder's issuing bank.

Whilst it may appear that the burden for a cardholder is significant in the above protocol, in reality an application such as a card issuer supplied applet that interacts with a platform's TCG Software Stack [149] could perform the majority of the cardholder's interactions with a TPM. The cardholder would only need to select and enter an authorisation string at Step 5 and enter a PIN/password at Step 9.

4.4 Client-Side Certification and Malware

The concept of client-side certification, as outlined in Section 4.3.1, works well if we assume an attack model that centres around external threats. However, as we have seen in Chapter 3, a model which only considers external threats is no longer appropriate in CNP transactions. In order for a cardholder to generate a signature using the private component of the key referenced in the X.509 certificate, the cardholder needs to send authorisation data to his TPM to activate his signature key. It is desirable that we secure the "channel" over which this authorisation travels. However, Trusted Computing as it exists today cannot secure this channel from the keyboard to the TPM. Thus, authorisation information may be observed and replayed by malware, in order to obtain access to the private key and generate signatures on unauthorised transactions. Moreover, malware would be capable of modifying subsequent transaction data that is sent to the TPM for signing.

4.4.1 Using Physical Presence

Our proposed mitigation for the malware problem is to use the current TCG requirement for TPM-enabled platforms to be able to demonstrate physical presence through a secure attention sequence to a TPM. Physical presence as defined by the TCG is a signal from the platform to the TPM that indicates operator-instigated hardware manipulation of the platform. Examples of such manipulation would include "depressing a key on the keyboard or some other such action" [153]. The combination of customer-provided card account details and evidence of the successful completion of a secure attention sequence can demonstrate that an authorised customer instigated the transaction. Only a person physically present at a platform can demonstrate physical presence and only an individual who knows the correct password for K_{i-priv} can load the key for use in a transaction. Even if malware were to surreptitiously observe cardholder authentication data, it would be impossible for it to generate new clandestine transactions without the presence of a user, as malware would be incapable of generating a corresponding secure attention sequence. Thus secure attention sequences can protect against transaction generating malware of the type discussed in [73].

As we saw in Section 2.4.1, the demonstration of physical presence on a platform is typically associated with administrative functions of the TPM. However, physical presence may also be demonstrated via the TPM_SetCapability command and the TPM_GetCapability command [154]. These two commands can be used to set and retrieve bits in the Deferred Physical Presence Bit Map (DPPBM) that forms part of a TPM's TPM_STCLEAR_DATA structure [150]. We examine this in more detail next.

In order for a user (or more precisely an *untrusted* piece of software operating on a user's behalf) to produce verifiable evidence of a (physical) commitment to a transaction, a user needs to issue a series of commands to his TPM. A user opens an *exclusive and logged transport session* [153] and calls the TPM_SetCapability to clear a single bit in the DPPBM. This command does not require a demonstration of physical presence and is used to prevent a bit from a previous transaction being reused by malicious software. Following this, a user again calls TPM_SetCapability, but this time to set the newly cleared bit in the DPPBM (here the setting of the bit requires the user to demonstrate physical presence). The user next calls TPM_GetCapability to read the newly set bit indicating that physical presence has been demonstrated. Finally, a user calls TPM_ReleaseTransportSigned to generate a *physical presence certificate*. The TPM_ReleaseTransportSigned produces a signature (using K_{i-priv} for which the user must supply the correct authorisation data) over a data structure that includes: a hash of the transport session log (consisting of the inputs, commands, and outputs encountered during the entire transport session) and a merchant-supplied anti-replay nonce. In our approach, this nonce will be a hash of the current payment record concatenated with a merchant-supplied random number. This physical presence certificate, together with the merchant's nonce, and the transport session log, can be used to construct a *physical presence package* which a third party can verify.

4.4.2 Migration

Our architecture allows a customer's private key that is bound to a particular TPM to migrate in a controlled manner to another TPM-enabled platform. We achieve this using the TPM's certifiable migration functionality. Certifiable migration is a TPM-specified operation [153] that permits the secure transfer of CMKs from one TPM-enabled platform to another. This transfer occurs in such a manner as to allow the new platform full usage of the migrated key. The process of migrating a CMK requires the inclusion of additional infrastructure components, in the form of either (or both) a Migration Selection Authority (MSA) and a Migration Authority (MA) (see Section 2.4.4.1).

In our architecture, irrespective of whether an MSA or MA is used, it is important that the customer's card issuer is specified as the controlling entity when the key is created. Without specifying the card issuer as either the MA or MSA, malware would be capable of migrating the CMK directly to an untrusted platform controlled by a fraudster.

Since the mechanism used to demonstrate physical presence is independent of the CMK used to sign this demonstration, once the CMK has been migrated it can be used as described in Section 4.4 to generate the physical presence package.

4.5 Augmenting Existing Protocols with Trusted Computing

This section examines the integration of client-side certification and the use of physical payment packages in SSL and 3-D Secure. At the end of this section we analyse the benefits of our approach in securing CNP transactions.

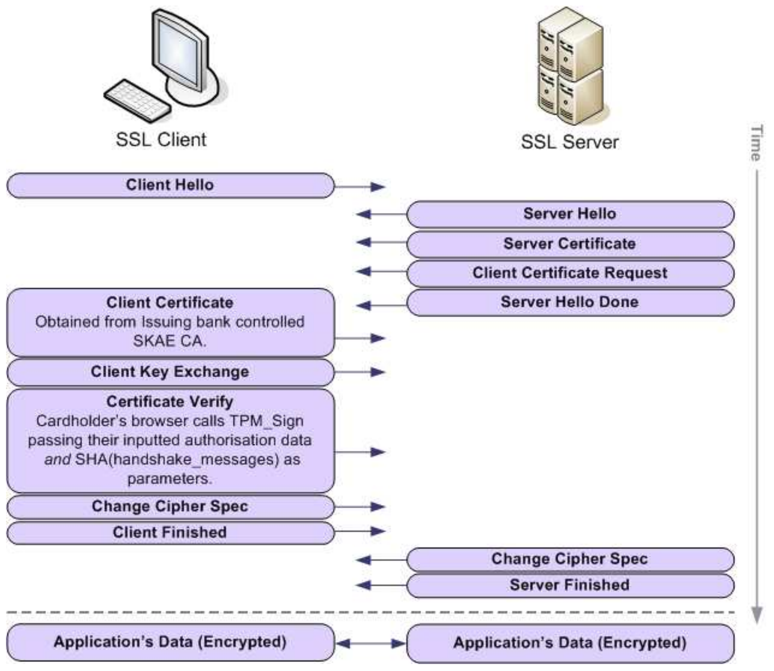
SSL Client
SSL Server
Time
Client Hello
Server Hello
Server Certificate
Client Certificate Request
Server Hello Done
Client Certificate
Obtained from Issuing bank controlled SKAE CA.
Client Key Exchange
Certificate Verify
Cardholder's browser calls TPM_Sign passing their inputted authorisation data *and* SHA(handshake_messages) as parameters.
Change Cipher Spec
Client Finished
Change Cipher Spec
Server Finished
Application's Data (Encrypted)
Application's Data (Encrypted)

Figure 4.3: Augmenting SSL Authentication

4.5.1 SSL Augmentation

SSL augmentation involves the addition of client (customer) authentication as supported (but seldom used) in standard implementations of SSL. Under the assumption of ubiquitous TPMs and the corresponding infrastructure that will be necessary to support them, we can use the enrolment mechanism outlined in Section 4.3 to provide a bootstrapping mechanism for providing client-side SSL certification.

The SSL process described here is identical to that of a standard SSL handshake in which client (cardholder) certification is requested by the server (merchant), see Figure 4.3. Here the server requests a certificate by sending a list of certificate authorities with which it is willing to participate. These requested certificate authorities may take the form of one or more root CAs (each CA run by a card association) or of one or more subordinate CAs (run by card issuers) depending on the constraints a merchant's acquirer wishes to place on the type of payment cards it is willing to accept.

If the client is in possession of an X.509 certificate that satisfies the merchant's request, then the cardholder's platform forwards this certificate to the merchant along with a certificate verify message (see Figure 4.3). This certificate verify message provides a proof of possession for the private key, K_{i-priv}, corresponding to the public key, K_{i-pub}, referenced in the client certificate. Here a customer's TPM is responsible for performing customer authentication prior to using K_{i-priv} to generate the certificate verify message. The process of generating this certificate verify message requires the authorisation data for K_{i-priv} (as supplied by the cardholder) as well as all the handshake messages exchanged thus far. These two parameters are input to a TPM_Sign command [154]. This command checks to see if the provided authorisation data matches the authorisation data stored with the private component of the requested non-migratable key. If they match then the TPM uses K_{i-priv} to generate a signature over the provided handshake messages. This signature is then passed to the merchant server for validation, subsequent to which the SSL handshake protocol proceeds as normal and the client can send its commitment to a transaction via the physical presence package. We note, however, that in implementing this modified SSL handshake the SSL session time-out parameter should be set *reasonably* high to accommodate processing delays resulting from accessing the client's TPM.

4.5.2 3-D Secure Integration

As we reported in Section 3.4.3, there has been some movement recently in bringing unconnected card-readers to market. The idea behind the provision of an 'unconnected' card-reader approach is that the reader (mirroring POS terminal functionality) would be capable of generating a one-time authorisation code, on a per

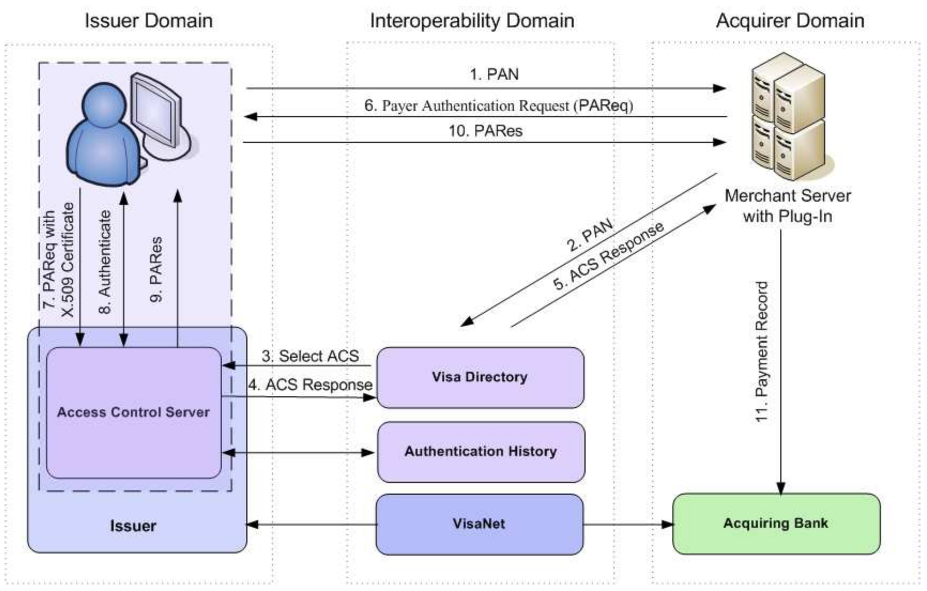

Figure 4.4: 3-D Secure Authentication

transaction basis. This could then be fed into a VbV or SecureCode environment in order to authenticate a customer to their card issuer.

An alternative approach could be to use Trusted Platforms as a means of authorising transactions. In this respect, given the rapid roll-out of TPM enabled platforms, the integration of Trusted functionality in a 3-D secure environment can be seen as an organic development in the current deployment of Trusted Platforms.

Enrolment into a 3-D Secure-like environment would occur as laid out in Section 4.3. We now outline an approach for a TPM-enhanced 3-D Secure purchase transaction flow

Initiate Purchase: This stage is representative of a 3-D secure initiation procedure revolving around the merchant plug-in component. In Step 1 of Figure 4.4 the customer forwards his PAN to the merchant. The merchant forwards this PAN to the Visa Directory Server which provides the address of an appropriate Access Control Server (ACS), (Steps 2-4 of Figure 4.4). The ACS's response is then forwarded to the merchant (Step 5 of Figure 4.4). The merchant plug-in constructs a PAReq containing cardholder, merchant, and transaction-specific information and sends this information to the ACS via the customer's browser (Step 6 of Figure 4.4).

Payment Authentication: This stage is representative of the 3-D secure payment authentication process. The customer forwards the merchant's PAReq as well as his own SKAE X.509 Certificate to its ACS (Step 7 of Figure 4.4). The ACS will then ask the customer to authenticate himself and commit to the transaction by generating a physical presence package for this transaction (Step 8 of Figure 4.4). This approach allows the ACS server to be sure that a valid customer is proffering their valid account details (as the customer's PAN forms part of the X.509 certificate) and the customer has physically performed an act that indicates approval of the transaction. The ACS validates the physical payment package and compares the transaction details contained in the PAReq to those contained in the physical presence package. If they refer to the same transaction, the ACS produces a PARes indicating that the customer has been authenticated and the transaction has been approved (Step 9 of Figure 4.4). This PARes is then forwarded to the merchant (Step 10 of Figure 4.4).

Payment Validation: Payment validation results from an examination, by the merchant plug-in, of the PARes generated by the ACS server. If the ACS signature validates correctly, then the merchant server can be assured that the transaction has been approved by the customer's issuer. Finally, the merchant forwards the customer's payment record to his acquirer (Step 11 of Figure 4.4).

4.5.3 Security Analysis

The primary advantage of our augmented SSL and 3-D Secure protocols over their unaugmented counterparts (the security analyses of which can be found in Sections 3.4.1 and 3.4.3 respectively), is that a remote verifier (be they merchant, acquirer or issuer) can gain (implicit) assurances about the protection levels associated with a customer's private key. In this instance, the private key used to authenticate a customer (and authorise a transaction) is not exportable to applications outside of a specific TPM-enabled platform.

Integration of Trusted Computing's physical presence signals with SSL and 3-D Secure can enhance the security of CNP transactions and address the threat posed by malware. Unfortunately, with our solution, user education surfaces as a potential weak link in the security chain. Regrettably, the manner in which physical presence functionality is presented to an end-user is entirely dependent on how a manufacturer chooses to implement it. Lack of customer education may be a difficult obstacle to surmount given the heterogeneity of physical presence implementations amongst manufacturer devices. Attesting to physical presence may be better suited to constrained devices such as mobile phones that conform to the upcoming Trusted Mobile specifications [155]. Here, the range of mechanisms available for this would be restricted by functional limitations and users would presumably become more aware of when they are being instructed to attest to physical presence.

A second significant drawback is that the use of secure attention sequences will not prevent malware from modifying an on-going transaction (as opposed to generating multiple new transactions). Here we have to rely on the lack of a strong economic incentive for malware to behave in this way — we can assume that it will simply not be beneficial for malware to modify individual transactions, since this would lead to rapid detection for little benefit (from the malware's perspective).

Stronger guarantees are unlikely to be available via Trusted Computing until recent initiatives such as Intel's LaGrande and AMD's AMD-V come to fruition (see Section 2.7). These efforts aim to provide additional Trusted Building Blocks in the form of hardware features, which can be exploited by next generation Operating Systems to provide strong security properties (such as non-interference, non-observation, and trusted I/O paths) for executing processes.

4.6 Conclusions

The use of the payment cards as an avenue for e-commerce is increasing at an unprecedented rate. In the physical world, the introduction of EMV for card-based payments at point of sale terminals has seen a dramatic reduction in the level of chargeback-related fraud. This is primarily due to the wide-spread deployment of tamper-resistant cryptographic hardware, preventing the cloning of cards.

Unfortunately, the benefits seen in the physical deployment of EMV for card payment transactions cannot be so easily gained in CNP scenarios. In this setting, knowledge of customer account information is all that is required to perform a transaction. This makes it impossible for a merchant or a customer's card issuer to determine if the valid owner of the account details being proffered is the one that actually instigated the transaction.

This chapter has attempted to address this imbalance by analysing the role Trusted Computing can play in augmenting two different mechanisms for securing CNP transaction details. We showed how the integration of Trusted Computing physical presence signals with SSL and 3-D Secure can enhance the security of CNP transactions and partially address the threat posed by malware. In doing so, we we note that solutions based on 3-D Secure are perhaps more amenable to the inclusion of physical presence signals since the merchant plug-in does not need to be modified to support customer-supplied attestations of physical presence. Instead all the verification logic can be embedded in the issuer's access control server. With SSL, much greater heterogeneity of implementations of server logic is possible, since it is the merchant and not the financial network domain that decides on the actual implementation. By tying payment authorisations to Trusted Computing hardware, in the form of a TPM, we provide similar benefits to those obtained with EMV. That is to say, knowledge of a customer's account details is no longer sufficient to complete a transaction; rather, a customer would need to demonstrate possession of a private key which is physically bound to a piece of hardware under their direct control. This approach can be easily adapted to other payment protocols such as SET, or indeed any protocol where it is important that a human presence be determined. However, a threat remains that a naïve user may be fooled into demonstrating physical presence by a sophisticated piece of targeted malware.

In the next chapter, as a natural extension of this chapter, we will examine how the extended security properties made available by potential future deployments of Trusted Computing technology (in which Trusted Computing functionality is fully integrated with OS and CPU) can be used to further enhance the security of card payments.

Chapter 5

e-EMV: Emulating EMV using Trusted Computing Technologies

Contents

In this chapter we show how the functionality associated with EMV-compliant payment cards can be securely emulated in software on platforms supporting Trusted Computing technology. We describe a detailed system architecture encompassing *user enrolment, card deployment (in the form of software), card activation, and subsequent transaction processing. Our proposal is compatible with the existing EMV transaction processing architecture, and thus integrates fully and naturally with the already deployed EMV infrastructure. We show that our proposal, which effectively makes available the full security of PoS transactions for Internet-based CNP transactions, has the potential to significantly reduce the opportunity for fraudulent CNP transactions.*

5.1 Introduction

In this chapter we propose e-EMV, a system that makes use of the full array of Trusted Computing technologies to securely emulate EMV for CNP transactions. We describe a system architecture encompassing user enrolment, deployment of software cards to customer platforms, card activation, and subsequent transaction processing. Our e-EMV proposal uses a combination of application software, a Trusted Platform Module (TPM) [153], a processor (with chipset extensions) [68] and Operating System (OS) support [121, 102] to securely emulate the functionality of a standard EMV-compliant card in software. We provide a detailed description, at the level of individual TPM commands, showing how this emulation is achieved. We also explain how the security features provided by Trusted Computing are used to obtain an appropriate level of security for our system.

Our approach of emulating EMV on Trusted Platforms for CNP transactions provides the following benefits:

1. It is possible to demonstrate e-EMV card ownership and authentication for CNP transactions as in standard EMV card authentication procedures. Thus a merchant can ensure that a customer claiming to posses a particular e-EMV

card is the legitimate owner of that card.

2. A merchant can obtain a payment guarantee through being able to demonstrate customer authorisation of a transaction.

3. A customer can gain an assurance prior to transaction initiation that a merchant will endeavour to protect sensitive cardholder information. Such a feature is is absent from EMV's use at Point of Sale (PoS) terminals, a fact which is now allegedly being exploited by criminal gangs [156].

4. An executing e-EMV application can mitigate the threat posed by malware Transaction Generators (TGs), by hosting e-EMV cards in their own isolated memory partition, free from observation and interference.

5. Our proposal supports EMV cards that are more cryptographically capable and offers enhanced authentication and transaction authorisation procedures, compared to the cards and procedures used in current EMV deployments for PoS transactions. While these enhanced security features (Dynamic Data Authentication (DDA) and Combined DDA and application cryptogram generation (CDA)) are specified in the EMV standards [47], they are not ubiquitous on today's EMV ICCs because of their cost implications; instead, some banks find it more economical to work with cheaper cards and accept the higher level of risk implied by the use of Static Data Authentication (SDA) [47]. However, our proposal, involving only software running on mass-market consumer computing platforms, is not restricted in this way.

6. In e-EMV, modifications and enhancements (resulting from, for example, changes to the EMV specifications) can be realised through relatively straightforward software update processes. In contrast, traditional EMV requires complicated and expensive card upgrades to achieve the same thing. This lends our e-EMV approach an inherent degree of "future proofing".

The motivation for emulating EMV, rather than designing a new protocol from scratch, comes from real-world, practical constraints. By creating an environment where EMV transaction flows can be mapped directly to e-EMV transactions we can avoid any expensive re-engineering of the back-end financial network. Indeed, our proposal is compatible with the existing EMV transaction processing architecture, and thus integrates fully and naturally with the already deployed EMV infrastructure. Additionally, as a consequence of EMV having been developed and deployed over many years, many of the protocol bugs should already have been ironed out.

Our e-EMV proposal could result in a mutually beneficial arrangement for TPM manufacturers and card issuers. Indeed, our e-EMV proposal could be a "killer application" for Trusted Computing in the consumer space, something that seems to be currently lacking.

Our e-EMV architecture closely follows the generic four corner model presented in Chapter 3. The only differences in our approach are that the communication channel between the customer and the merchant is now Internet-based (instead of being based on physical proximity of card and terminal), and that both card issuers and acquirers must provide enrolment facilities for their e-EMV clients, separate to the facilities provided for the physical issuance of an EMV card. For a card issuer this means providing a mechanism through which customers can establish their e-EMV cards on their platforms. Similarly, for an acquirer this means providing a facility whereby a merchant can download an application that can interface with an customer's e-EMV application. In doing so, we assume the presence of a Public Key Infrastructure (PKI), which can be an extension of the one that currently exists for EMV. In this regard, enrolment facilities should be able to be authenticated by customers and merchants via public key certificates issued by a particular card association.

This chapter is structured as follows. In Section 5.2, we present an overview of related work. In Section 5.3 we provide a high-level overview of our e-EMV architecture. Section 5.4 explains in detail the procedures and processes involved in establishing an e-EMV card within a Trusted Computing enhanced platform. Following on from this, Section 5.5 highlights how a PoS EMV transaction flow can be mapped to an e-EMV transaction. Section 5.6 examines how the threats posed to CNP transactions in Chapter 3 are mitigated by e-EMV. We conclude with Section 5.7.

5.2 Related Work

Herreweghen and Wille [66] present a detailed evaluation of the security requirements for card-based Internet payments. These include: payment guarantee for merchant, mutual authentication between customer and merchant, customer privacy and anonymity, and transaction authorisation.

The real world roll-out of EMV and the accompanying desire to replicate the fraud reduction seen with PoS transactions has resulted in a number of proposals that utilise EMV's functionality for Internet-based payments [1, 45, 66, 78, 110]. However, these approaches have not seen any real traction in the market place. A possible explanation for this is these proposals' underlying assumption that customers would make use of card readers connected to their PC platforms. This assumption engenders an additional cost in the form of distributing card readers to end-users. Even if the cost issue could be surmounted, these approaches alone offer only limited security gains. This is due to the lack of a trusted path between the card-reader and host, as well as a lack of OS support for application isolation. Without these, a Transaction Generator could passively observe Personal Identification Number (PIN) entries, actively modify transaction data and possibly generate new transactions, enabling criminals to remotely take control of users' payment cards. There has, however, been a recent development in integrating EMV with CNP transactions that aims (albeit indirectly) to address the TG issue. This is the proposed use of "unconnected" card readers [85] which have been rolled out by a number of prominent UK banks. Unfortunately, this approach also suffers with respect to the additional costs associated with distributing card readers to end-users and, once deployed, cannot be updated to address new threats as they emerge.

The use of Trusted Computing to combat phishing has been proposed in [2] and [53]. The primary threat considered in [2] is external attack, whereby a credential needs to be extracted from the client's platform in order for it to have any value to an attacker. No consideration is given to the ever-increasing threat posed by malware TGs. The threat from malware is examined in greater detail in [53, 58, 72, 73]. Here, much like in our approach, Virtual Machines (VMs) are used to constrain the use of malware to an individual VM "compartment" (see Section 2.7 for a discussion of the security properties of a VM). These authors also suggest using visual cues as to the trustworthiness of the VM with which an end-user interacts, an idea originating in [14]. Such work is complementary, but orthogonal, to our own: these cues could be adopted in e-EMV and might help enhance customer protection from TGs.

5.3 An overview of e-EMV

This section presents a high-level overview of our e-EMV proposal. Our overall aim is to provide functionality akin to that of a standard EMV card by replicating that functionality through procedures and capabilities natively supported by a host that is augmented with Trusted Computing. This allows us to provide a secure and extensible architecture for CNP transactions.

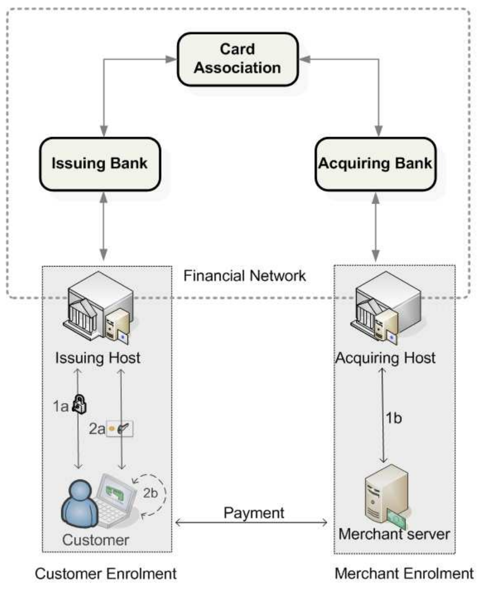

Figure 5.1: Enrolment Procedure.

Establishing an e-EMV card on a Trusted Computing platform is a two-stage process consisting of an account activation stage and an application delivery stage. In describing this architecture we are making the following important assumptions: we assume the presence of a PKI extending the one currently in existence for EMV, we assume that Trusted Computing Platforms are ubiquitous within the merchant/customer domain, and additionally, that both processor and OS support are available to all platforms within this domain. Furthermore, we make the underlying assumption that a card issuer has already made the decision to extend credit or debit facilities to a particular customer.

5.3.1 Enrolment

Enrolment in our e-EMV architecture involves a customer formally registering as a legitimate cardholder, allowing him to obtain an e-EMV card. Much like enrolment in the traditional EMV architecture, a card issuer within our system is responsible for enrolling cardholders as well as later authenticating their transactions (and possibly the cardholders themselves). Acquirers can be seen as providing similar functionality to their merchant customers. The stages for establishing an e-EMV card on a TPM-enabled platform are as follows.

Account Activation: Account activation is the process through which a customer becomes a member of a card-issuer-controlled group. In this case, group membership is indicative of a customer activating their account within the system (Figure 5.1, Step 1a). In the context of Trusted Computing, this means enrolling with a Privacy CA (see Section 2.4.5) or optionally becoming a member of a card-issuer-controlled Direct Anonymous Attestation (DAA) group (see Section 2.4.5). This is achieved by the customer demonstrating the presence of a non-migratable TPM-controlled secret over which certification is requested, an AIK public key (for which the corresponding private component is maintained and managed by the TPM) in the case of the Privacy CA or a secret value f for DAA. Note, as with Chapter 4, in discussing the use of a Privacy CA and DAA here, we are not suggesting that our approach should actually benefit from the privacy-enhancing features available from either of these choices. Rather, we see the card-issuer playing the role of either a Privacy CA or DAA issuer as providing a convenient mechanism for achieving client-side authentication within the limitations of the TPM specifications.

At this point, the actual binding between the customer and his platform can be established through a mechanism of the issuer's choosing. For example, the customer might provide information such as a passcode supplied to them in the pre-enrolment stage (in which the card issuer agreed to extend debit/credit facilities), communicated using an out-of-band mechanism. In addition to this authentication information, a platform will send various platform credentials (describing the binding of the TPM to the platform) as well as evidence of the existence of a non-migratable TPM-controlled secret. After receiving this information, the card issuer performs due diligence to satisfy itself as to the relationship between a customer and his platform. This is achieved through a reconciliation of the provided authentication information with an examination of evidence supporting the existence of a TPM-controlled secret. If the card issuer is satisfied by this evidence, the customer's platform will receive certification of their TPM-controlled secret. This certification will later be used to demonstrate the customer's membership of a particular card scheme.

Enrolment for the merchant (Figure 5.1, Step 1b), by contrast, involves satisfying the requirements for payment processing as laid down by the relevant acquirer's Merchant Operator Guidelines (MOGs). During the merchant enrolment procedure, the merchant's acquirer also becomes a certificate issuer, again in the context of the Privacy CA or DAA models.

Secure Application Delivery: The customer downloads a small secure application bundle (Figure 5.1, Step 2a) that fulfils the role of an e-EMV card as well as acting as a guide through the process of creating/installing the requisite TPM managed keys (Figure 5.1, Step 2b) [47]. This bundle, once installed, will enable a platform to perform electronic transactions analogous to those carried out by an EMV card at a PoS terminal. Our fully realised e-EMV application contains all the keys required to perform CDA authentication. However, unlike an EMV card, our e-EMV application will need to provide some of the functionality typically seen in PoS terminals, particularly when it comes to cardholder verification (see Section 5.5).

The merchant will need to download and install a terminal plug-in application, similar in function to a 3-D Secure merchant plug-in [12], but capable of emulating certain EMV terminal functionalities. The most important of these will be the authentication of customer-supplied credentials. Much like terminals in the physical setting, merchants will require card issuer's public key certificates in order to verify

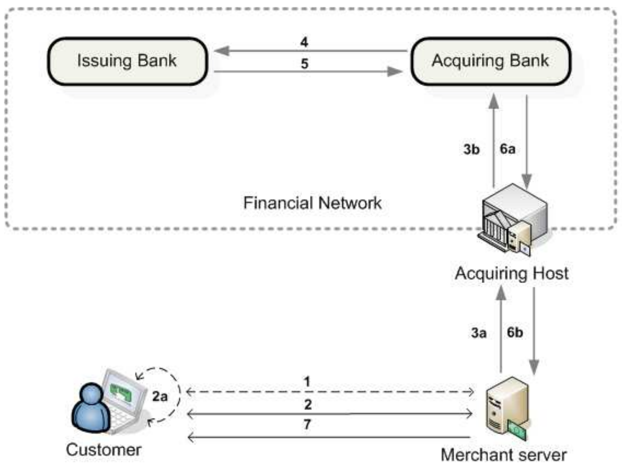

Figure 5.2: Transaction Architecture for e-EMV Payments.

customer transactions. This plug-in should be compliant with Visa's PABP and the PCI-DSS standards. We do not discuss the operation of the merchant terminal plug-in in any greater detail in this chapter as this information is outside the scope of the EMV specifications and is largely proprietary to individual acquirers.

5.3.2 Transaction Architecture

In order for an e-EMV application (card) to be launched, a Trusted Platform's OS sends an instruction sequence to the DRTM to create a new isolated memory partition. The DRTM launches a VM into this newly created memory area, which in turn executes our e-EMV application, free from observation by and influence of TGs. For the details of launching a secure VM, we refer readers to [68]. With the exception of Step 1 (see Figure 5.2), the basic flow for e-EMV transaction processing follows EMV's transaction processing at a PoS terminal. The steps are as follows:

Step 1 represents the browsing phase in which a customer peruses a particular merchant site. During this step, the customer verifies that the merchant is a member of a valid group of merchants, corresponding to a particular Privacy CA/DAA issuer (acquirer). Additionally, the customer's e-EMV application, acting on the customer's behalf, may verify that the merchant is in a state that satisfactorily addresses privacy and confidentially concerns (for example, conformance with the PCI-DSS or Visa's PABP) via Trusted Computing's attestation procedures.

Step 2 represents the typical EMV ICC and terminal interaction, except that the communication channel between the customer and the merchant is now exclusively Internet-based.

Step 2a represents the creation of an AC. As part of this step the customer's platform generates a signature over its AC (using the certified non-migratable TPM-controlled secret established during the account activation phase) as well as the Platform Configuration Register (PCR) values that provide evidence as to the platform's current state at the time of transaction authorisation. These data items are communicated over a secure and authenticated channel to the merchant server.

Steps 3 to 7 are executed if the optional decision to go "on-line" has been exercised. As a result of either terminal or card risk management procedures, the AC (possibly in conjunction with the information adducing the current platform state) is forwarded to the customer's card issuer (Steps 3a–4). After examining the received data, the card issuer returns an AC of its own (Steps 5–7). This AC informs both the card and the merchant as to whether the request is to be approved or declined, in which case the card application will either proceed with the transaction or reject it. It is important to note that all e-EMV messages exchanged in Steps 2–7 are standard EMV message flows.

Where additional Trusted Computing platform state information has been added to EMV's AC messages in our architecture, it is optional for a card issuer to examine this state information as part of its decision making process. The Tag Length Value (TLV) encoding mechanism used in EMV makes it easy for a card issuer to ignore any extraneous information from a transaction referral request. Significantly, if the card issuer does decide to take a customer's platform state into consideration, the customer's card issuer would not need any Trusted Computing hardware to examine these characteristics. It would only need to be capable of hashing some supplied records, comparing them against known good metrics for a particular card application and verifying a signature (as we shall see in Section 5.4). Indeed, such functionality could be provided by a third party facility or performed by the merchant plug-in.

5.4 Installing and Instantiating an e-EMV card

This section explains in greater detail the process involved in establishing an e-EMV card within a Trusted Platform.

5.4.1 Account Activation

In order for a customer's e-EMV account to be activated, the customer's Trusted Platform must become a member of a card issuer controlled group. This process is mirrored for the merchant server account activation procedure with respect to a merchant's acquirer. In both cases, account activation is achieved by successfully obtaining/generating a credential issued for an AIK public key. This credential could be in the form of X.509 certificate [70] issued by a Privacy CA or a credential issued by a DAA Issuer [154].

In either case, the choice of a Privacy CA or DAA Issuer will end with a similar result: the customer will have their account activated, later allowing them to demonstrate physical/logical possession of an active card within the system. Through the establishment of a customer-centric credential embedded within a platform, the customer will be able to attest to the existence of a key capable of replicating EMV's SDA/DDA/CDA functionality.

In obtaining this credential, the credential issuer needs to ensure that a request is coming from a particular customer. This can be achieved, for example, through a customer demonstrating knowledge of a shared secret created in the pre-enrolment stage over a secure channel. However, the actual mechanism used is orthogonal to our discussion.

5.4.1.1 Privacy CA Approach

Here the process involves a customer's TPM generating an attestation key pair (S_{AIK} and P_{AIK} for the private/public portions respectively) and having P_{AIK} incorporated into an ICC Public Key Certificate (in our context an AIK credential). When creating this key, the customer specifies authorisation data in the form of a password which will be required every time S_{AIK} is loaded for use. The process involved in the generation of a new attestation key within a platform maps to the generation of an AIK key within a TPM, that is, as a result of performing the TPM_MakeIdentity command [154]. The P_{AIK} portion of this AIK, along with various platform credentials and customer authentication data, are encrypted with the Privacy CA's public key and sent to the card-issuer-controlled Privacy CA. After authenticating the particular customer and satisfying itself that the request is coming from a genuine TPM, the Privacy CA will issue an AIK certificate (ICC Public Key Certificate) to the customer's TPM-enabled platform. This credential can then be used to provide evidence of card activation within the system. The AIK certificate could be further enhanced through X.509v3 extensions. For example, it is possible to add key and policy information to the credential, such as setting a private key usage period during which the AIK signing key will be valid.

5.4.1.2 DAA Approach

The DAA enrolment procedure occurs exactly as laid out in [27]. However, here the role of the DAA Issuer is provided by the card-issuer-controlled issuing host. After successful completion of the DAA_Join command [154], the customer will instruct his TPM to create a new AIK pair (S_{AIK} and P_{AIK} for the private/public portions respectively). As part of this key generation process, the customer specifies a password for S_{AIK}, entry of which will be required every time the key is loaded for use. The customer then instructs his TPM to sign the P_{AIK} component using his newly enrolled DAA key to create the ICC Public Key Certificate. This will later allow the customer to be able to demonstrate that his account has been activated with respect to a particular card issuer.

Subsequent to a successful completion of either the DAA join procedure or upon receipt of an AIK certificate from a Privacy CA, the card issuer activates the customer's account within its systems. This enables the soon-to-be-downloaded e-EMV application to be used in ensuing financial transactions.

5.4.2 Secure Application Delivery

The delivery of the e-EMV application to a customer's platform involves an interactive process between a customer and his card issuer. The delivery of the software itself necessitates a number of checks to ensure the binding between a customer and his platform. The software set-up utility requires the inclusion of a unique EMV ICC AC Master Key per card issuance, and this key cannot be generated by the platform itself as it is derived from a card issuer Master Key. Instead the ICC AC

Master Key needs to be injected into the platform as part of the application delivery process. In addition to the secure delivery of the application, there is an underlying customer-driven requirement to ensure the authenticity and the "behaviour" of the application once delivered. The dual concerns of authenticity and behaviour can typically be addressed using what is termed a validation credential in the Trusted Computing literature [151]. In this way, load-time metrics can be compared against known good values to assure customers (and merchants) that their applications will perform as intended.

The issue of secure delivery of an application using Trusted Computing has previously been examined in the context of conditional access in mobile systems [55]. Much like the approach adopted in [55] in which a third party specifies the state to which a broadcast application must be bound, we require the card issuer to be able to specify the state on the customer's platform to which the e-EMV application will be "sealed".

Sealed messages in the context of Trusted Computing are messages that are bound to a set of platform metrics (specified by one or more PCR values) and that can only be opened when the platform is in a certain state. This is normally achieved with the TPM_Seal [154] or the TPM_Sealx command [154]. However, the TPM_Seal and TPM_Sealx commands only allow messages to be sealed locally to a customer's platform. To overcome this problem, a card issuer can specify a state to which a private key must be sealed on the customer's platform. The corresponding public key can be sent to the card issuer, which can satisfy itself that the private component of this key is indeed sealed to a specified platform state (by examining the TPM_PCR_Info structure associated with the signed public key). The card issuer can then encrypt the e-EMV application with this public key and send it to the customer.

In our proposal, the delivery of an application to a customer's platform, as well as the corresponding requirement of securely storing the application upon receipt, is handled in three stages. In the first stage, the customer generates an asymmetric key pair, specifying security constraints for the private key. In the second stage, the customer sends the public key of this key pair to the card issuer. The card issuer will then encrypt the customer's e-EMV application with the customer's public key. In the third stage, the customer decrypts his e-EMV application (provided the security constraints specified for the private key are satisfied by the platform). The details of these stages are as follows.

Stage 1: In the first stage, the customer's TPM generates either a CMK or a non-migratable key pair. The set-up phase of this is illustrated in Algorithm 5.1.

Algorithm 5.1 e-EMV Application Download Set-up Phase

```
1: Issuer → Cust: Sealed State || Issuer$_{Cert1}$ || Guide || $Sign_{Issuer}$(Sealed State || Guide)
2: IC_Key := TPM_CreateWrapKey(TPM_Auth_Priv_Use_Only, digestAtCreation, diges-
   tAtRelease)
3: TPM_Certify_Info := TPM_CertifyKey(IC_Key)
```

Notation: Here Issuer is the card issuer and Cust is the customer. $Sign_X$(Y) denotes a signature with the private key of entity X over data item Y and X$_{cert1}$ is the public key certificate for entity X for which the corresponding private key is used for signing only. Finally, Guide is a small software applet that guides the customer through the creation of the necessary TPM keys for secure application delivery.

In **step 1**, the customer's card issuer specifies a state to which it would like the customer to "seal" a private key. This state will typically be a representation of the MVMM and the minimal VM software components necessary for e-EMV application virtualisation. In this step the customer also receives the card issuer public key certificate (from the EMV PKI [47]) and a small signed application that will guide the user through the creation of the requisite TPM keys.

In **step 2**, after having verified the signatures received in step 1, the guide application, acting on the customer's behalf, creates a new asymmetric key pair, IC_Key. The application calls the TPM_CreateWrapKey command to generate a new asymmetric key pair, and asks the customer to enter a password (*usageAuth*) which will be required every time the private component of this key pair is loaded (TPM_Auth_Priv_Use_Only). The application captures the current platform state at the time the key pair is created (digestAtCreation) as well as the future platform state required for private key usage (digestAtRelease). Here digestAtRelease is specified to be the *SealedState* provided by the card issuer.

In **step 3**, the newly generated IC_Key is certified by S_{AIK} using the TPM_CertifyKey command [154]. Here the customer's TPM is effectively producing a signature over the public component of IC_Key using a signature key that the card issuer knows to be bound to a particular customer's TPM. The ability to demonstrate use of the private component of IC_Key can provide the same level of assurance as results from the TPM_Seal command. This is because the "digestAtRelease" parameter is specified during the generation of the private component of the IC_Key. In this case, specifying digestAtRelease is semantically equivalent to sealing. The output of the TPM_CertifyKey command produces a signed TPM_Certify_Info structure which includes a TPM_PCR_INFO structure describing the state to which the private component of the IC_Key key pair is sealed.

Stage 2: Once the set-up stage is complete, the download of a customer's e-EMV card application can proceed as illustrated in Algorithm 5.2:

Algorithm 5.2 Downloading the e-EMV Application

```
1: Issuer → Cust: $R_{Issuer}$ ‖ $Issuer_{cert2}$ ‖ [Optional Platform Attestation]
2: Cust → Issuer: $E_{Issuer}$(TPM_Certify_Info ‖ $P_{IC\_Key}$ ‖ Platform Attestation ‖ $R_{Issuer}$
   ‖ $R_{Cust}$ ‖ $Sign_{S_{IC\_Key}}$(Cust ‖ Issuer ‖ $R_{Cust}$ ‖ $R_{Issuer}$))
3: Issuer → Cust: $R_{Cust}$ ‖ $E_{P_{IC\_Key}}$(e-EMV application)
```

Notation: Here R_x is a random number, P_{IC_Key} is the public key of the IC_key key pair generated in the set-up stage and S_{IC_Key} is the private key of the IC_key key pair generated in the set-up stage. $Sign_{S_{IC\_Key}}(Y)$ denotes a signature with the IC_Key private key of data item Y, $E_X(Z)$ denotes encryption with the public key of entity X over data item Z, and X_{cert2} is a public key certificate issued by entity X, for which the corresponding private key is used for decryption only.

In **step 1**, the card issuer sends a challenge, its public key certificate and, optionally, its own platform state.

In **step 2**, the customer verifies this platform state (if present) as well as his card issuer's public key certificate. The customer returns the TPM_Certify_Info structure (generated in the set-up phase) which contains a hash of the public component of IC_Key and a description of the security constraints (of the private component) of IC_Key. In addition to this, it sends the public component of IC_Key and signed platform metrics, responds to the challenge of the card issuer, and sends a challenge of its own, all encrypted with the public key of the Issuer.

In **step 3**, the card issuer examines the received data. Provided the storage semantics for the private key (IC_Key) match the card issuer's security policy and the customer correctly responded to the earlier challenge, then the card issuer sends a reply to the customer's challenge and the customer's e-EMV application encrypted with the public key of the customer's IC_Key pair.

Stage 3: The final stage of the protocol for secure application delivery is applica-

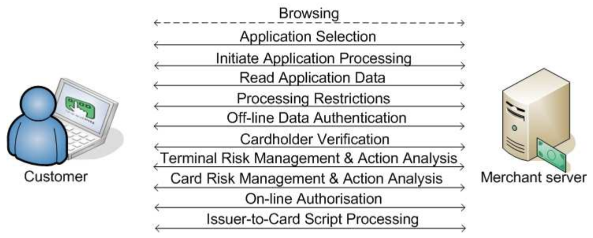
Browsing
Application Selection
Initiate Application Processing
Read Application Data
Processing Restrictions
Off-line Data Authentication
Cardholder Verification
Terminal Risk Management & Action Analysis
Card Risk Management & Action Analysis
On-line Authorisation
Issuer-to-Card Script Processing
Customer
Merchant server

Figure 5.3: e-EMV Customer-Merchant Interface

tion retrieval and launch. Providing that the current platform state matches the requirements for the sealed data (as specified by the card issuer) and the incoming authorisation data matches the authorisation data set for the private key during the set-up phase, then the application is unsealed by the TPM. If one or both of these two conditions is not met then the application will remain sealed until both conditions can be fulfilled. Following successful delivery of the e-EMV application, the customer is now able to utilise their card on their TPM-enabled platform.

5.5 e-EMV in Operation

Once our e-EMV application (card) is launched into its own isolated VM by the platform's MVMM, transaction processing proceeds as follows.

5.5.1 Customer to Merchant Interaction

Customer-to-merchant interaction in our e-EMV system uses the same transaction processing used for EMV at PoS terminals. EMV transaction flows can be mapped directly to e-EMV transactions flows as follows (see Figure 5.3):

Application Selection: Typically, application selection occurs in response to a terminal-initiated command to select the appropriate EMV application from a multi-application card. In this case we assume a single application instance in which customer/merchant application matching is dependent on finding a suitable set of validation credentials supported by the platform, or a reported set of one or more PCR values in the form of a platform attestation, representing a valid application execution.

Initiate Application Processing: The customer's application next commences application processing. In the physical world this would yield a response in which the card returns its application interchange profile and its application file locator. However, in the e-EMV setting as the cards are virtualised and free from the constraints of typical ICC cards, all e-EMV cards would have CDA capabilities. Additionally, as part of this step the merchant terminal plug-in provides to the e-EMV application any terminal-related information pertaining to the business environment at the point of sale. An example of such information would be terminal capabilities or the country in which the terminal is operating.

Read Application Data: The steps involved in issuing one or more READ RECORD commands [48] in our setting is redundant as we assume the application is executing solely in main memory on a customer's platform.

Processing Restrictions: This mandatory step is performed by the merchant and does not require any direct interaction with the customer's execution environment. This step is primarily concerned with judging the compatibility of the e-EMV application with that of the terminal application. This process can be handled in an e-EMV environment through a process of reconciliation between terminal-supported applications and customer-supplied validation credentials/platform attestation.

Off-line Data Authentication: The three options for off-line data authentication available in EMV-compliant cards are SDA, DDA and CDA. These all involve a PoS terminal issuance of an INTERNAL AUTHENTICATE command [48]. We achieve this functionality in our e-EMV application through the platform's attestation mechanism. We now discuss how this is achieved for each of the three options. In doing so, we do not suggest that either SDA or DDA mechanisms be implemented. However, as CDA builds upon the functionality of SDA and DDA, we present them independently for ease of exposition.

SDA: SDA is relatively simple process whereby a validation credential (generated by a card issuer) providing metrics for a correctly functioning e-EMV card application is compared against load-time PCR metrics reported in an attestation challenge to a merchant. An outline of this process is as follows (here M is the merchant and Req is a request for one or more PCR registers):

$$1{:}\ M \rightarrow Cust : \mathrm{M}_{Cert} \parallel \mathrm{Req} \parallel \mathrm{S}_M(\mathrm{Req}) \parallel \text{Platform Attestation}$$

$$2{:}\ Cust \rightarrow M : \mathrm{E}_M(\mathrm{Cust}_{Cert} \parallel \mathrm{P}_{AIK} \parallel \text{Platform Attestation})$$

In step 1, the merchant server requests one or more PCR values corresponding to the PCRs to which the e-EMV application is bound. In addition to this request the merchant attests to its own platform metrics for examination by the customer's platform. The customer's platform then examines the attested merchant metrics and determines whether the merchant adheres to certain desirable policies, such as the PCI-DSS standards for card processing or Visa's PABP. If satisfied, a customer's platform agent culls its SML for the events responsible for generating the requested PCR values. The TPM then loads the AIK signing key using customer-provided authorisation data. The e-EMV application, using the now loaded AIK key, calls the TPM_Quote command [154] to sign the requested PCR registers. This will either be the AIK for which a certificate was obtained during the e-EMV account activation stage (as per the Privacy CA model) or an AIK over which a DAA signature has been generated (as per the DAA model). This information is returned to the merchant server for verification, as the platform attestation part of message 2. The merchant then examines the AIK credential, checks signatures and compares a hash of the SML entries to the attested PCR values. If they match, then the merchant can be sure as to the current state of the VM in which the e-EMV application is executing.

In effect, this procedure achieves the same function as SDA does in standard EMV.

DDA: DDA requires a slight modification to the SDA mechanism as outlined above. During the operation of SDA's TPM_Quote, in which the PCR registers are signed, the customer incorporates a merchant-supplied 160-bit random challenge in the attestation. This challenge takes the place of the operand *externalData* which is of type TPM_NONCE [150] in the PCR attestation process.

CDA: In the EMV PoS environment, CDA involves computing an ICC private key signature over the dynamic application data of which an AC (specifically a TC or an ARQC) and a random challenge (as per DDA) are integral parts. Replicating this in the e-EMV environment is not substantially more complicated than either the SDA or DDA approaches. The e-EMV application carries out the AC generation process. The AC, along with its data output (TC/ARQC) can be integrated into a PCR register. The actual examinable output, that is the TC or ARCQ, is then correspondingly appended to the SML and a signature is generated over the representative PCR values using S_{AIK}. In this way the merchant, upon receipt of the attestation bundle, can examine the SML for the inclusion of an AC and verify its correctness through a signature verification.

Cardholder Verification: In the typical PoS EMV operation, this step allows the terminal to verify the authenticity of a cardholder. This authentication is based on the card and terminal both supporting a particular Card-holder Verification Method (CVM). In addition to the CVM being used to verify a customer, some CVMs, particularly PIN authentications, are used as a means of authorising transactions. In this instance, the PIN may be either verified “off-line” by sending the PIN from the PIN entry device to the card for local verification or by encrypting the PIN and sending it to the card issuer for “on-line” verification.

Through the use of Trusted Computing functionality we can make PIN authentication and authorisation intrinsic to transactions. This is achieved through the TPM key authorisation mechanism, whereby certain keys require 20-bytes of authorisation data to be supplied prior to being loaded. Use of an AIK requires authorisation data to be securely communicated to a TPM. In our architecture, this authentication data is communicated over a secure Object-Independent Authorisation Protocol (OIAP) session [153]. The use of an OIAP session allows authorisation information to be sent to a TPM without revealing the data on the channel over which it is sent. The necessary authorisation data is itself specified during the account activation process.

The ability to actually use a key as part of a transaction demonstrates to a merchant that a CVM has occurred successfully. By verifying the state of the platform, a merchant can be assured that only a valid customer would be capable of using the AIK private key generated in the account activation stage.

To prevent malware from launching a dictionary attack against key authorisation data, it is important that the TPM provides some form of resiliency to such an attack. The current iteration of the TPM specifications [153] details an example mechanism where a count of failed authorisation attempts is recorded. If this count exceeds a threshold, the TPM is locked and remains non-responsive to further requests for a predetermined time-out period. However, at present, the implementation of a dictionary attack resistant authorisation mechanism is vendor-specific and is not always implemented securely [119].

Terminal Risk Management and Action Analysis: The purpose of terminal risk management and terminal action analysis is to protect the issuer, acquirer and payment system from fraud. The existing variety of measures used by EMV in terminal risk management, such as floor limits, random transaction selection and velocity checking can all be replicated in the merchant terminal plug-in in our e-EMV architecture.

Card Risk Management and Action Analysis: Details of card risk management are proprietary to card issuers and are outside the scope of the EMV specification and as such would remain proprietary to individual card issuers within our e-EMV system.

On-line Authorisation: During EMV transaction processing the merchant terminal may decide to proceed with an on-line check; this again can be replicated in our e-EMV environment via standard EMV message exchanges.

Issuer Script Processing: To allow updates to EMV cards in the field, the card issuer can return scripts via the terminal to the ICC for processing. These scripts are not necessarily relevant to the current transaction and may be used to update applications during the utilisation phase of the ICC's lifecycle, or to transition the card into a blocked or unblocked state. Updates to e-EMV cards can be performed in a similar manner. An issuer can send update scripts via the merchant plug-in to the customer's e-EMV card. The issuer can then distribute new state measurements for updated e-EMV cards to merchants via new validation credentials. Merchants should only allow e-EMV cards whose attestation matches the values contained in the latest validation credential to perform e-EMV transactions. Card blocking scripts are a more difficult EMV feature to replicate in an e-EMV setting. It may be illegal in certain jurisdictions for a card issuer to force an update on a customer's platform without gaining customer consent.

5.5.2 Payment

As we saw in Section 3.5, in EMV, an ICC uses session keys derived from the ICC Master Key to protect the transaction messages. In our e-EMV architecture, in conjunction with the information typically sent in the AC message, the e-EMV application sends the current platform state as signed by its issuer-validated signing key (S_{AIK}). After examining the AC message as well as the supplied state, the verifier (a merchant or a card issuer) can make a decision as to whether or not to proceed with a transaction. In the instance where an AC is an ARQC, as mentioned in Section 5.3.2, the TLV encoding scheme used in EMV allows the issuer to ignore extraneous information if they so choose. The card issuer can then respond with an ARPC indicating whether the transaction should be approved or declined, in which case a TC or an AAC will be generated by the customer's platform. The transition to Internet-based communications in e-EMV requires that ACs be protected in transit between the customer and merchant. Such protection can be achieved by establishing a TPM-centric SSL tunnel between customer and merchant, as per Chapter 6 or [58]. We note, however, that as these messages are already integrity protected using a shared session key, in certain environments additional confidentiality protection may not strictly be necessary.

5.5.3 Migration

Enabling the migration of an e-EMV card from one Trusted Platform to another may be desirable in our architecture. We can achieve such a feature by using the TPM's certifiable migration functionality. If the customer creates a CMK during the application delivery process (see Section 5.4.2), a customer may later migrate his application bundle to another TPM-enabled platform. However, as neither DAA secrets nor AIK private keys obtained in the enrolment phase are migratable from a

TPM, the customer would need to rerun the account activation phase (see Section 5.4.1) in order for their card to be usable on the new platform. In this instance, the cost of providing additional infrastructural elements to support a trusted migration service [144] may be somewhat prohibitive for issuers, and it may just be simpler in practice for the customer to enrol his new platform with his card issuer from scratch.

5.6 Security Analysis

The semantics of trust enforced by Trusted Computing functionality enables both parties, the merchant and the customer, to obtain certain guarantees that were hitherto unrealisable in past proposals [133, 12]. Both merchant and server can be sure as to the integrity of their communicating peer's platform, i.e. that each peer will behave in the expected manner (in this case, adhere to, and faithfully adduce, state characteristics corresponding to legitimate transaction states). It is important to point out that we do not expect the customer to be able to recognise or validate software states within our system. This function can be fulfilled by the application software itself and should be reported to the customer in a way that is understandable.

We now analyze the effectiveness of our e-EMV architecture as a means of securing electronic payments. This is achieved by examining how well it satisfies the security requirements of Herreweghen and Wille [66], described in Section 5.2.

Mutual authentication of customer and merchant: In our architecture both customers and merchants are authenticated via their card issuer/acquirer supplied credentials, providing a much stronger form of authentication than that currently employed for CNP payments. As we have seen in Chapter 4, these credentials can be used to authenticate the establishment of SSL/TLS sessions, thus providing an additional layer of protection for the transport of e-EMV transaction messages over the Internet, if required.

An added benefit of our approach is that it allows the customer to examine a merchant's platform state prior to transaction authorisation. This enables the customer to satisfy himself that the merchant will behave in a manner that will protect his sensitive card data. Such a feature is lacking with current PoS transactions, a fact which is now allegedly being exploited by criminal gangs [156].

Transaction authorisation & payment guarantee for merchant: In our architecture, the use of the private transaction authorising key is contingent on a particular platform state being present on the customer's platform. Much like PIN entry at a PoS, the completion of a transaction is conditioned on the input of correct authorisation data, ensuring the logical presence of the cardholder in the remote transaction. Additionally, as part of the transaction authorisation process, a customer's platform must attest to the VM in which the e-EMV application is executing. Any divergence from intended operating state (due to unwanted memory resident applications) will be picked up in the attestation, allowing merchant risk management routines to terminate the transaction.

Assuming a transaction goes ahead, payment guarantee for the merchant is provided by CDA. CDA provides a signature over an e-EMV card's Transaction Certificate, providing the merchant with non-repudiable evidence of payment. Cardholder non-repudiation is dependent on the degree to which the cardholder's private signing key is securely generated and stored in the cardholder's Trusted Platform.

Customer privacy and anonymity: Customer privacy is somewhat problematic in any CNP transaction as the customer will typically provide significant amounts of personally identifiable information, such as name and billing address. Protection of customer privacy is not provided by EMV, and therefore not provided by our e-EMV application.

5.7 Conclusions

The use of the Internet as an avenue for electronic commerce, in the form of CNP transactions, has grown rapidly in recent years. However, CNP transactions are currently far from secure. This chapter proposed a new security architecture for securing CNP transactions. By creating software-based EMV cards running on Trusted Platforms, our e-EMV proposal replicates many of the features of standard EMV-compliant cards for use in CNP transactions. Through our account activation and secure application delivery procedures, we established how cards can be remotely provisioned within our system. We showed how card ownership can be demonstrated by the customer through the use of an OIAP session enabling secure PIN entry. We also showed how EMV transaction messages can be mapped to e-EMV transaction messages. We demonstrated how EMV keys can be generated and bound to a particular TPM-enabled platform. Through these various measures, we can achieve a significant improvement in the level of security afforded to CNP transactions.

Our work raises a number of areas for further research. In this chapter, we have focused on describing architectural and technical aspects of our e-EMV proposal. Our future work will examine security and system management issues. A prototyping activity based on the OpenTC framework[1] with SVM [4], L4 μ-Kernel[2] and Xen [24] support is also likely to be useful in terms of revealing unforeseen practical issues, operational problems, and the like. Additionally, as part of future work we also look to examine the role of physical presence in providing enhanced transaction authorisation (see Chapter 4) within our e-EMV framework. At present, our approach is reliant on MVMMs and Trusted Computing augmented processors being present

[1]http://www.opentc.net/

[2]http://os.inf.tu-dresden.de/L4/

in commodity platforms. Unfortunately, such support is not widely available today. A more immediate avenue for adoption would be Trusted Computing enhanced mobile phones [155] which do not require MVMM support. However, at present only preliminary details of the TCG's trusted mobile architecture are available [155]. We believe that Trusted Computing can provide an enhanced level of security compared to EMV's deployment at PoS terminals. However, given the perpetual increase in attacks (both in terms of number and sophistication) targeting end-user systems, we see an increased role for both terminal and card risk management routines to control transaction risk in our e-EMV architecture.

Whilst outside of the immediate scope of this chapter, in developing our architecture we note two weaknesses in the current attestation mechanisms adopted by the TCG. Firstly, as noted in [65], the current TCG attestation is static, inexpressive and has poor handling of patches and upgrades to system software. Secondly, the TCG attestation mechanism only concerns itself with load-time measurements of applications. Recent approaches that attempt to address these issues have been proposed in [65, 120, 134] and [83, 132, 134] respectively. Investigating these proposals in the context of e-EMV will be of interest. In the absence of more expressive attestations we must rely on the properties of the MVMM to ensure that our executing e-EMV application cannot be interfered with by outside applications.

We further note the possibility of extending our system by exploiting additional features of the DAA protocols to support pseudonymous payment cards. In such an extension, we envisage the information identifying individuals being removed from the merchants' view of transactions, with acquirers still being able to obtain the necessary payment guarantees.

Part II

Other Applications of Trusted Computing

Chapter 6

Trusted Computing and Peer-to-Peer Systems

Contents

In this chapter we study the application of Trusted Computing to securing Peer-to-Peer (P2P) networks. We identify a central challenge in providing many of the security services within these networks, namely the absence of stable, verifiable peer identities. We employ the functionalities provided by Trusted Computing technology to establish a pseudonymous authentication scheme for peers and extend this scheme to build secure channels between peers for future communications.

6.1 Introduction

The concept of Peer-to-Peer (P2P) networking covers a diverse set of network types, supporting a wide variety of applications. The common feature shared by almost all P2P networks is the lack of any centralised control. In this respect, P2P networks are the antithesis of the traditional client-server model. P2P networks have most famously become popular in the form of P2P file-sharing, providing a means of distributing (often copyrighted) material such as music and video. Commercially-oriented P2P networks are now coming to the fore. Useful introductions to P2P networking can be found in [87, 95].

Aside from availability, security issues for P2P networks have not yet been widely addressed. A major conflict arises from the perceived requirement to provide anonymity for users of P2P networks and an increasing need to provide robust access control, data integrity, confidentiality and accountability services. These services are increasingly important as industry moves towards using P2P technology, as applications such as Skype[1], and P2P e-commerce, emerge [37]. The security situation for P2P networks is made worse because, by definition, they lack any cen-

[1]`http://www.skype.com/`

tralised authority which can vouch for identities or security parameters. Without the foundation of stable, verifiable identities, it is difficult to build any of the desired security services. In particular, pseudospoofing attacks, in which malicious parties claim multiple identities and disrupt the operation of P2P networks, are difficult to prevent.

In this chapter, we demonstrate how features of the TPM specification (as described in Chapter 2) can be employed to enhance the security of P2P networks. In particular, we show how the TPM protocols for Direct Anonymous Attestation (DAA) can be used to enforce the use of stable, platform-dependent pseudonyms and reduce pseudospoofing in P2P networks. Further, our use of DAA provides a means of building entity authentication and simple access control mechanisms. Taking this a step further, we show how runs of the DAA protocol, providing authentication of pseudonyms, can be securely linked to the establishment of secure channels with known endpoints. Such channels allow the protection of data in transit in P2P networks. We show how the cryptographic mechanisms we provide can be integrated with standard SSL and IPSec protocols. An important feature of our work emerges here: while earlier authors [57, 79, 124] have posited the application of Trusted Computing in P2P networks, we do not halt at that point. Rather, we actually describe the specific Trusted Computing mechanisms and commands which enable us to establish security in P2P networks.

After establishing how the basic security features can be provided by using Trusted Computing mechanisms, we go on to discuss two distinct approaches to using these mechanisms to enhance the security of P2P networks. The distinction arises from the nature of the “root of trust” for DAA in the two versions.

In the first version, the root of trust for DAA is assumed to be provided by a credential issued in a controlled manner at the time of platform manufacture, by one of a small number of platform manufacturers. This model is matched to the widespread deployment of Trusted Computing technology on end-user platforms and is most interesting when it is exploited to secure traditional P2P networks. In this context, our research has a quite unusual implication. Trusted Computing is often portrayed as a technology which has the potential to provide mechanisms to control the spread of digital content. But in our architecture, it becomes a mechanism that can be used to enhance the security of, for example, a P2P file-sharing network distributing content, some of which may be proprietary. The security provided does not prevent a content owner from purchasing a TPM-enabled platform and joining the network, but it does prevent casual eavesdropping on network communications, and gives a degree of anonymity (strictly, pseudonymity) for the users. While it is still possible for content owners or their agents to flood the network with poor quality or bogus material through pseudospoofing, the cost of doing so quickly becomes prohibitive as this would require the use of multiple TCG-compliant platforms in order to claim multiple identities. As we shall report in Section 6.2, such attacks have been well documented as a mechanism by which the value of P2P file-sharing networks can be reduced. In addition, the originators of such material can be pseudonymously identified and barred from the network, using, for example, a reputation mechanism founded on the stable pseudonyms assured by the use of TCG mechanisms. Going further, our DAA-based access control mechanism can be used to add strict controls on who can access the network. In the limit, completely private P2P networks can be constructed, with peers restricting access to a closed group of pseudonyms.

The second version of our security architecture is more commercially-oriented. Here, the root of trust for DAA is a company offering content or information services to customers who are willing to pay for the service. Now DAA can be used to register customers, authenticate registered customers, and prevent customers sharing registrations. The accesses made by a particular customer (or rather platform) to different service providers are unlinkable because of the properties of DAA. This kind of usage of DAA has already been anticipated in [28]. Our work can thus be seen as filling out the details of this approach by specifying which TPM commands need to be issued in order to register and authenticate customers. In addition, we present a P2P twist on this fairly routine application of Trusted Computing. We propose that registered customers can act as nodes in a P2P network for distributing the content owned by the service provider. Because of the trust properties of a TPM-enabled platform, the service provider can allow this to happen without fear of losing control of its content. At the same time, the service provider can reduce its requirements for storage and bandwidth, instead relying on peers to do this work. One potential incentive for peers to become content distributors could be a reduction in access fees; further possible incentives are discussed in Section 6.5.2. We also discuss mechanisms by which the service provider can continue to collect revenue from customers who obtain their content from network nodes rather than the service provider itself. Note, in discussing the use of a DAA in this chapter, we are not suggesting that our approaches should actually benefit from the privacy-enhancing features available from DAA. Rather, we see DAA as providing a convenient mechanism for achieving client-side identification within the limitations of the TPM specifications.

This chapter is structured as follows. In Section 6.2, we provide a short overview of P2P networks and discuss a number of security concerns that arise in these networks. In Section 6.3, we discuss how DAA can be used to provide stable pseudonymous identities in P2P networks. We build upon this work in Section 6.4 and describe how these identities can be used in the establishment of secure channels between peers. In Section 6.5, we discuss two approaches to DAA credential issuance and examine the impact that these may have on P2P networks. We close this chapter with Section 6.6, in which we discuss of some of the issues and open problems raised by our work. We also point to areas which are opened up by our work and which seem ripe for further exploration.

6.2 Overview of Security Issues for P2P Networks

There is a multitude of definitions for the concept of P2P networks in the literature. The one we adopt for the purposes of this chapter is taken from [87]:

> *A peer-to-peer network is a class of systems and applications that employ distributed resources to perform a critical function (usually in a decentralised manner).*

Resources here could be computational power, data, or network bandwidth, while critical functions could include data or information sharing, distributed computing, communications or collaborative working. Within this broad definition of P2P networks, one can include systems such as Napster, Gnutella, BitTorrent, Distributed Hash Table (DHT) based systems, as well as systems less traditionally thought of as being P2P, such as social networking sites or eBay. While eBay itself is very much a centralised architecture, the reputation system within eBay exhibits P2P aspects, with peers contributing to the reputation of other peers.

Despite the initial and still main use for content sharing, P2P networks have come a long way in a short time and are gradually being adopted in commercial and corporate applications such as network storage [81], content distribution [109] and web caching [71]. P2P networks represent something of a paradigm shift from the traditional client-server model for service provision. The P2P approach can bring many benefits: scalability, efficiency, fault resilience and, of course, reduced reliance on central servers. It is through the entities in these networks, called *peers*, that resources are replicated and shared.

6.2.1 Security Issues

Whilst many aspects of P2P networks have been thoroughly researched, security within these networks still remains a challenge. It is not our intention to exhaustively review security issues for P2P networks here. Indeed, good overviews of this topic can be found in [35] and [164]. Rather, we focus on identifying some key security issues in P2P networks and pointing towards some of the approaches to solving them that have been identified in the literature.

The security architecture [69] associated with the ISO/ITU Open Systems Interconnection (OSI) reference model serves as a useful framework for assessing security issues in networks. According to [69], a secure system is governed by the set of security services it provides, and the mechanisms put in place to cater for these services. The set of potential security services in [69] is divided into five main classes:

- Confidentiality,
- Integrity,
- Authentication (both data origin and entity authentication),
- Access Control, and
- Non-repudiation.

Additional services not explicitly included in [69] could be added to this list. Accountability is an important service in networks where users are to be charged for their use of resources. Anonymity in various forms is often cited as being desirable, especially in the context of P2P networks. Anonymity can refer to obscuring all identifying information of an individual, or it can refer to making the actions of a particular individual unlinkable. Availability is sometimes considered separately from security. Since many malicious attacks on networks are directed at reducing availability for legitimate users, we explicitly include it here as a security service. Any particular network may require a combination of all, some, or even none of these services.

In traditional client-server systems, users are typically identified with a user account, and platform or system-specific controls can be applied to these accounts to provide security mechanisms. Security services such as access control and accountability can be implemented in this manner, with the accounts providing a form of stable identity. Other services such as authentication and non-repudiation clearly also rely on the establishment and preservation of stable identities. Moreover, if we want to establish cryptographic keys for confidentiality and integrity of data in transit, then entity authentication of one or more of the communicating endpoints is usually necessary for security. Often, entity authentication is enabled using Trusted Third Parties (TTPs). For example, one might rely on a Public Key Infrastructure (PKI) and the certificates issued by a Certification Authority, or one might use the services of a dedicated TTP, as in Kerberos [91]. Thus most (though not all) the generic security services we might wish to provide are reliant on the provision of stable identities.

In a P2P environment, there are by definition no centrally-administered accounts or trusted third parties who can provide assurances as to the identities of entities in the network. Instead, each peer is typically identified based on a *handle* or *pseudonym* that is selected by the peer for itself. In most P2P networks, there exist no standard registration procedures for these pseudonyms, and indeed entities can claim multiple pseudonyms, including those of other entities in the network. The use of pseudonyms provides a degree of anonymity for peers and eases discovery and routing of resource queries in P2P networks, but is clearly in conflict with the stable identities that are needed, as noted above, to provide many other security services. This problem of providing entity authentication in the face of untrusted networks of peers seems to be a central challenge in providing wider security services for P2P networks. Having identified this challenge, we now go on to make a more detailed examination of the security issues in P2P networks, and their relationship to the identity/authentication issue.

6.2.1.1 Authentication and Pseudospoofing

Pseudospoofing, a term coined by Detweiler [40], refers to a peer creating and handling more than one pseudonym at once. Potentially, an attacker might manipulate tens or even hundreds of pseudonyms to his advantage. An attacker might also make use of the pseudonym of an existing peer, preferably one with a good reputation. This type of attack has been named *ID stealth* in the literature [33], but we prefer the term *pseudotheft*. An attacker who engages in pseudospoofing or pseudotheft in a P2P network may do so in order to manipulate a reputation mechanism in place in that network. A peer, having gained a bad reputation, can discard the corresponding pseudonym and re-join the network with a fresh pseudonym. In online auction systems, a peer can use a pseudonym to generate false bids, an attack known as *shilling*. In general, a peer can create a number of pseudonyms, build up the reputation of one or all of the pseudonyms using the others, and then make use of the now reputable pseudonyms to persuade other peers to trust him. Clearly any P2P network attempting to rely on a reputation system for building trust in peers would need to address the problems of pseudospoofing and pseudotheft. It is evident that pseudotheft can also lead to loss of confidentiality. It has been argued that in any P2P system without a centralised point of trust, such attacks on identity are endemic and can never be effectively combatted [43].

Lest these attacks seem theoretical, we note that there are many concrete cases citing the issue of pseudospoofing as a potential problem. We briefly discuss two examples. In the first case, we note the reputation system used in eBay, essentially a P2P system, has constantly been under pseudospoofing attack. A particularly well-documented case is that of the philatelic frauds perpetrated by the Saratoga gang on eBay.[2] In the second case, the Recording Industry Association of America (and their agents) have been accused of setting up bogus identities in order to spread poor quality music files in certain P2P file sharing networks, notably Kazaa and networks based on Gnutella and Bittorrent, with the alleged aim of disrupting these networks to prevent sharing of protected materials.[3] These activities amount to attacks on the authenticity of resources on the networks.

The key aspect to note in the above examples is the relative ease with which peers can create, use, and abuse identities. In the absence of strong registration and entity authentication procedures, it becomes hard, if not impossible, to enforce stable identities or pseudonyms. While the problem is widely recognised, research directed towards preventing pseudospoofing has been sparse, perhaps reflecting the truly decentralised nature of P2P networks. For example, the problem is discussed in [33], but it is simply assumed there that identifiers are tamper-resistant. A statistical, quorum-based approach to building a PKI suitable for supporting P2P security services, including entity authentication, was proposed in [37]. Another important contribution to this area is the proposed use of Cryptographically Generated Addresses (CGA) for IP routing [13]. CGA is a method for binding a public key to an IPv6 address. Here the public key belonging to some entity is hashed, the output of which forms the interface identifier for their IPv6 addresses [38]. In this way an entity can assert ownership of an address by signing messages sent from this address. Additional work examinging this problem is given by Advogato[4] where trust relationships are modelled as a directed flow graph. Trust between peers is modelled as

[2]An interesting account of this case, "The Saratoga Fakes", has been compiled by the Stamp Collectors Against Dodgy Sellers (SCADS) institute, `http://www.scads.org/alterations/Saratoga.htm`.

[3]Content posted on (`http://www.mediadefender.com/`) specifically refers to this kind of disruptive activity.

[4]`http://www.advogato.org/trust-metric.html`

the presence/absence of an edge in the graph and the amount of weight assigned to this edge determines how much one peer trusts another. How trustworthy a peer is in relation to the graph as a whole is calculated as the maximum flow that can be pushed from a source (trusted) peer to another peer. When a peer joins the graph it is isolated from other peers and so there is little reason to trust it.

6.2.1.2 Accountability and Reputation Mechanisms

In the P2P context, accountability means the ability to hold peers responsible for their actions and the resources that they share/consume. This aids in achieving efficient utilisation of resources, discourages peers from consuming resources whilst only providing nominal resources in return (a practice known as *free-riding*), and gives peers protection against fraud.

Despite these known benefits, it is hard to ensure accountability due to the inherently autonomous and transient nature of peers in P2P networks. A peer can join and leave the network at any time. Network topologies are dynamic and it is uncommon to have a known history of all peers, nor are there any contractual obligations between peers. Most obviously, anonymity and accountability requirements are in clear opposition. Moreover, these problems are exacerbated by the lack of stable identities in P2P networks.

There are various schemes in the literature aimed at addressing the issue of accountability in P2P networks. A common approach is to use some kind of reputation mechanism to build trust in P2P entities and the resources they hold. The rationale for this approach is that knowledge that peers will consider each other's past interactions while pursuing present and future transactions constrains peer behaviour in the present, thus imposing accountability in the form of non-abuse of resources [114].

A classic example of such a reputation mechanism is the web-of-trust approach adopted in PGP [170]. Here, a user can calculate a trust level associated with another user's public key, where the calculation depends on a number of factors, including which other users are prepared to trust that key, and how well they themselves are known to the relying party. The scalability and applicability of this kind of approach for large and widely distributed P2P networks has been questioned in [37], where an alternative approach based on distributed PKI was suggested. Another interesting approach, based on micropayments, was proposed in [89]: users are made accountable for their use of resources via micropayments. The micropayment tokens are generated through work performed by the users. The P2P network Mojo Nation[5] uses its own currency called "mojo" to handle micropayments between peers. On the other hand, Free Haven [89] requires participants to provide storage space for other peers to use, a form of payment-in-kind for accessing the network.

However, as we have already seen, these reputation mechanisms are open to manipulation through pseudospoofing and other attacks.

6.2.1.3 Anonymity

Anonymity in a P2P context, in its strongest sense, means that there should be no mechanisms that link peer identities, aliases, actions or responses within a network. Anonymity greatly appeals to the P2P user-base because it not only offers them privacy protection, enabling facilities such as censorship resistance and freedom of speech, but also the ability to break copyright laws or publish libellous material without fear of litigation.

Anonymity for P2P networks has been classified into four distinct categories [89] — *author anonymity* (which user created which resource), *server anonymity*

[5]`http://sourceforge.net/projects/mojonation`

(which peers store a particular resource), *reader anonymity* (which peers access which resources) and *document anonymity* (which resources or documents are stored at a given peer node). As we have already noted, the provision of these forms of anonymity conflicts with other security goals and hampers the design of strong reputation mechanisms. Moreover, providing complete anonymity usually impacts on network performance and routing [31, 82]. As examples, the P2P networks Freenet[6] and Publius[7] have made efforts to provide anonymity. Freenet, in particular, encrypts data between its nodes and routes messages through other nodes. In the process, it is made difficult to determine which node is requesting data and what the data contents are. Also, peers have to contribute a portion of their disk space to the network, but peers do not know what is stored in their local data stores. These techniques are, in general, adapted from existing anonymity techniques such as MIX networks [30, 113], crowds [112] and onion routing [143].

An absence of anonymity implies a loss of privacy. Therefore present systems typically employ a middle ground whereby each peer within the network is recognised by a pseudonym. Here, the actions associated with a particular pseudonym can be linked, but there is not a strong binding between the pseudonym and an underlying identity.

6.2.1.4 Access Control

Another security problem faced by peers within a P2P network is access control. Within a P2P environment, this would mean restricting access to authorised peers, for example, peers who have paid for the queried resources. This would certainly be desired by companies managing intellectual property and digital rights, but may also be of interest in more traditional P2P networks. The (deliberate) absence

[6] `http://freenet.sourceforge.org`
[7] `http://cs1.cs.nyu.edu/~waldman/publius/`

of such controls has led to widespread abuse of copyright laws and a consequent fusillade of litigation against companies making P2P-based software and individuals operating on P2P file-sharing networks. However, it is certainly not the case that access control is needed in every P2P network. A careful analysis would be required to evaluate what impact an access control mechanism would have on peer reputations. For example, a peer restricting access to queried resources may get a bad reputation with other peers. Moreover, building robust access control mechanisms once again depends on authentication: without stable identities, traditional access controls cannot be implemented.

6.2.1.5 Confidentiality and Integrity

Confidentiality and integrity can be classified broadly as applying to storage (data at rest) or communications (data in transit). In the context of P2P networks, the former refers mainly to resources stored at peers, while the latter refers to transactions that occur between peers, for example, resource queries and replies, distribution of reputation information, and so on. Attacks on confidentiality and integrity can be achieved by a variety of means. For example, it may be possible to manipulate the routing protocols used in a P2P network so that data of interest to an attacking peer is routed via that peer.

A common method for ensuring integrity of resources in transit and at rest in P2P networks is to calculate and append Message Authentication Code (MAC) values or digital signatures to the resources. While an integrity service guarantees that a resource has not been altered (either in transit or while at rest), it may say little about the original source of that data. Questions of that nature can be settled using a data origin authentication service in combination with, for example, a time stamping service. This is referred to as file authenticity in [35], but we prefer the terminology of [69].

Confidentiality and integrity are not often discussed in the context of P2P security. Some research and implementation has already taken place in certain networks, like Microsoft's Groove[8], but these services are optional there. If one regards PGP as providing a secure P2P messaging system, then it provides a good example of the careful implementation of confidentiality and integrity services in a P2P context. Confidentiality and integrity are clearly fundamental to security, and we believe they are set to become more important with the emergence of commercial P2P networks. The requirement to protect valuable resources at rest and in transit is likely to outweigh the processing and management overheads associated with these services, and will be a key enabler for P2P e-commerce.

Once again, a barrier to the development and deployment of these security services for data in transit in P2P networks is the lack of stable identities. The problem is not that one cannot implement the services using well established encryption and MAC mechanisms, but that there is little point in doing so if peers do not know with whom they are establishing communications. Otherwise a peer may fall victim to man-in-the-middle attacks or send sensitive data to unintended recipients. Once again, authentication of peers is a vital precursor to providing security services in P2P networks.

6.2.2 Summary of Security Issues

Research on P2P security has covered a variety of problems and is summarised above and elsewhere [35, 164]. We have identified what we believe to be a fundamental issue which must be addressed if better than trivial security is to be added to P2P networks: the lack of stable identities and the inability to provide strong authentication

[8]`http://www.groove.net/home/`

services, which in turn underpin the security of reputation systems, accountability, access control, confidentiality and integrity services in P2P networks. We have also seen that full anonymity in P2P networks clashes with other security services. A reasonable compromise is to work with pseudonyms, shielding the binding between on-line and "real-world" identities, but to find methods to prevent pseudospoofing and pseudotheft attacks.

6.3 P2P Pseudonymous Authentication Using Trusted Computing

In this section, we consider how the use of stable pseudonyms can be enforced in any P2P network in which each peer is TPM-enabled and has acceptable DAA credentials. In subsequent sections, we will examine how this basic property can be leveraged to provide security services in P2P networks.

The mechanism for enforcing stable pseudonyms is simple. We assume only that the P2P network has a particular name, that is, a unique identifier. The pseudonyms used in our network are strings of the form $N_P = \zeta^{f_P}$, where ζ is derived from the network name by applying a hash function, and f_P is the secret associated with the TPM located on the peer P claiming the pseudonym.

Whenever a peer claims a particular pseudonym, the peer verifying that claim can challenge the claimant to supply a DAA signature on, say, a random message M. Later, we will see how to enable other security services by selecting M to include additional values. The verifying peer can check, using the DAA-Verify algorithm, that the claimant has a valid credential supplied by a particular issuer and can be convinced that the value f_P used in forming the claimed pseudonym N_P is the same as the one claimed at the time of credential issuance. The use of a random message M prevents replay attacks in which a rogue peer captures and replays credentials. Through these checks, the pseudonym N_P that can be claimed by peer P is fixed, and is determined as a function of the network name and the particular f_P certified during the joining phase. In summary, we have built an *pseudonymous authentication mechanism* from the DAA algorithms. Through this mechanism, peers can be authenticated by other peers, but platform identities are not revealed in the process.

To make this a workable method for enforcing stable pseudonyms, we need to ensure that a particular platform will only ever obtain one credential from one of a set of authorised issuers that are recognised by the verifying peer. Otherwise, it may be difficult to prevent a peer obtaining multiple credentials for different f values, from one or more issuers, and using these to produce multiple pseudonyms. One mechanism for ensuring that a platform can only ever obtain a single credential is to assume that an initial credential is issued to the platform in a controlled fashion at the time of manufacture. The set of authorised issuers is then the set of platform manufacturers, and the peer software can be configured to insist on seeing a credential issued by one of these manufacturers. Here, then, the manufacturers become the root of trust for DAA. This is a likely scenario in the deployment of Trusted Computing. We explore this situation further in Section 6.5.1. We relax the assumption about issuance of single credentials in our second model for the use of DAA in P2P networks, described in Section 6.5.2.

We now examine the extent to which our approach prevents pseudospoofing and pseudotheft attacks. We note that, if our condition on credential issuance is met, then a peer cannot claim multiple pseudonyms. Nor can a peer claim the pseudonym of another peer. Thus these attacks, so destructive in P2P networks, are prevented. The usual blacklisting mechanisms can be used by peers to detect the use of a rogue TPM, provided the relevant information can be distributed and kept up-to-date.

Of course, nothing in our approach prevents an attacker from purchasing multiple TCG-compliant platforms and using these to create multiple pseudonyms. However, this kind of attack has an economic cost for the attacker.

It is clear that, by enforcing the use of pseudonyms, a given peer's actions on the network become linkable, so our approach does not offer a full level of anonymity. We have already established that this is necessary if other security services are required. Moreover, the use of DAA never reveals the platform identity (in the form of the endorsement key) to verifying parties, so the peer's actions cannot be linked to a particular TPM. Notice too that the pseudonym used during the join phase is different from that revealed during DAA-Signing (because $\zeta_I \neq \zeta$ in general). Thus the pseudonyms do shield the actions of a peer effectively. We also note that the pseudonyms we propose for use do not relate to particular individuals; rather they are linked to particular platforms (more precisely, TPMs).

A given peer may wish to operate in more than one P2P network. Because of the use of the network name in determining the value of ζ used to form pseudonyms, we obtain the nice feature that a peer's actions in different networks will remain unlinkable.

Finally, we note that an alternative authentication mechanism can be built using the Privacy CA approach. The idea is to extend PCRs using nonces exchanged between peers as measured data. The PCRs can be signed using AIKs, which are in turn signed by the Privacy CA. The signed nonces can be exchanged by interleaving two runs of the TCG attestation protocol (see Section 2.4.5). We omit the details since they are closely related to the techniques introduced in the next section. Obviously, this approach would depend upon the emergence of an entity sufficiently qualified to provide Privacy CA facilities to a P2P network.

6.4 Securing P2P Networks Using Trusted Computing

In this section we will demonstrate how a range of security services can be built from our DAA-enabled authentication mechanism and the stable pseudonyms which it enforces.

6.4.1 Access Control and Accountability

As outlined in Section 6.2.1.2, access control is particularly problematic in large unrestricted P2P networks. Given the authentication mechanism outlined in Section 6.3, it would be trivial to implement a simple access control mechanism based on access control lists. In this case, an access decision could be made based on the peer pseudonym presented to (and verified by) the peer granting access to resources. Clearly a more sophisticated and application-dependent access control system could be implemented on top of the stable pseudonyms.

As we have reported in Section 6.2.1.2, accountability and robust reputation models are hard to build in the presence of pseudospoofing and pseudotheft. Now that we have the ability to enforce stable pseudonyms in a P2P network, it becomes possible to build robust reputation systems that will not be undermined by pseudospoofing and pseudotheft attacks. Such a reputation system provides a means to hold each peer accountable for its actions. We note that the deployment of reputation systems seems to be a necessary precursor for the development of P2P e-commerce as envisioned in [37]. The description of specific reputation systems is beyond the scope of this chapter. For the interested reader, a survey of reputation systems can be found in [76].

6.4.2 Authenticated Key Establishment and Secure Channels

Whilst authentication (even at the level of pseudonyms) is a very useful service to provide in a P2P network, its usefulness is limited if it cannot be extended to provide protection for an entire communication session between peers. We address this next, showing how our authentication mechanism can be extended to an authenticated key establishment protocol. From there, it is a short step to integrating our DAA-based mechanisms with SSL and IPSec, these being two widely used existing methods for providing secure channels for communicating entities.

6.4.2.1 Two Generic Approaches

Suppose we have two peers P and Q, with pseudonyms N_P and N_Q, in a particular P2P network. We assume these peers wish to authenticate one another[9] and establish keying material for protecting further communications.

In the simplest form of our approach, the peers first exchange random nonces R_P, R_Q, ephemeral Diffie-Hellman values g^{x_P}, g^{x_Q} (where g is a generator of a suitable group negotiated in advance or by peer software configuration), and then exchange signatures on the 160-bit hash[10] of the messages

$$M_P = N_P \| N_Q \| R_P \| R_Q \| g^{x_P} \| g^{x_Q} \text{ and } M_Q = N_Q \| N_P \| R_Q \| R_P \| g^{x_Q} \| g^{x_P}$$

The signatures are obtained by each peer using their TPM and platform to execute the DAA-Signing algorithm on the hash of M_P (or M_Q). Using the DAA-Verify algorithm, each peer can check that the other has a valid credential supplied by a

[9]The authentication need not be mutual, and the alterations to our methods needed in this case are straightforward.

[10]The TCG specification limits externally generated messages that are to be DAA signed to be at most 160 bits in length.

particular issuer and can be convinced that the values f_P, f_Q used in forming the claimed pseudonyms N_P, N_Q are the same as those claimed at the time of credential issuance. Here, we are essentially re-using the pseudonymous authentication mechanism from Section 6.3, but with M made up of a longer string including pseudonyms, nonces and Diffie-Hellman values. If both signatures are confirmed by DAA-Verify, then the parties can confidently compute common secret keying material $K = g^{x_P x_Q}$, and then derive keys for MAC and symmetric encryption algorithms from K.

Here, we have used the facility that a platform can ask the TPM for the DAA signature on any message M. However the TPM may be configured to only execute DAA-Signing on AIKs. If this is the case, we can use an alternative (but slightly less efficient) approach involving PCRs and AIKs. Each peer can use the string $M = N_P \| N_Q \| R_P \| R_Q \| g^{x_P} \| g^{x_Q}$ as measured data to extend a PCR, using the process outlined in Chapter 2, Section 2.4.5. Then each peer can request its TPM to produce an attestation to the PCR value, which takes the form of a signature on the PCR value produced using the private component of the peer's AIK. Next, at each peer, the DAA-Signing algorithm can be used to sign the peer's AIK. Finally, the peers can exchange the DAA signatures on their AIKs and the AIK signatures on the string M. Checking these signatures provides the necessary authentication for the Diffie-Hellman key exchange. This approach can be realised in practice by first extending the PCRs and then interleaving two runs of the TCG attestation protocol outlined in Chapter 2, Section 2.4.5.

We now go on to investigate a third approach, better suited to integration with existing secure protocol suites, in Section 6.4.2.2. After that, we examine which TCG commands are needed to implement our ideas.

6.4.2.2 Integration with SSL/TLS and IPsec

We have outlined two mechanisms by which DAA can be bootstrapped to build secure communications between pseudonymously authenticated endpoints. Our motivation here is to build on existing protocols to achieve the same aim, still using DAA and attestation of PCRs as a basis for authentication, but re-using protocol components where possible. We focus on SSL/TLS and IPsec as starting points.

As we saw in Section 3.4.1, either one or both parties in SSL/TLS will have a key pair that is used in the Handshake Protocol. Validity of these public keys is ensured through the verification of X.509 certificate chains back to a trusted root. We show how to replace this process with pseudonymous authentication in a TCG-enabled P2P architecture. The main idea, once again, is to extend a PCR, this time using the public key(s) as measured data. By signing the extended PCR using an AIK, and then signing the AIK using the DAA-Signing algorithm, a chain of signatures extending from the SSL/TLS public key(s) to the DAA issuer's public key can be constructed. Thus, in this approach, the role of the SSL/TLS root CA would be filled by the issuer of DAA credentials.

One complexity here is that at least one of the signatures in this chain is not of the type traditionally seen in SSL/TLS. Namely, at least one of the signatures is a Camenisch-Lysyanskaya signature representing the issuer credential. Another complexity is that the chain of signatures is not arranged as an X.509 certificate chain. For this reason, it may be simpler to just use self-signed X.509 certificates in the SSL/TLS Handshake Protocol and follow this protocol run with a second phase comprising an exchange of DAA credentials and PCR attestations using two runs of the TCG attestation protocol. Ciphersuite negotiation in SSL/TLS would determine which keys needed to be incorporated into PCRs in the TCG attestation protocol. Nonces exchanged in the SSL/TLS Handshake Protocol should be included as part of the measured data in the PCR extension step to bind the SSL/TLS key exchange and pseudonymous authentication steps.

This approach is sufficiently flexible to support any of the SSL/TLS standard key exchange methods. As options, the issuer names in the self-signed certificates could be the peer pseudonyms N_P, N_Q and the subject name field could be a meaningful handle that the peer (user) wished to be known as on the network. This last option would help to address the issue of associating a meaningful name with a platform's pseudonym, an issue we will return to in Section 6.6.1.3. All of this could be made transparent to the users in the same way that SSL/TLS hides the intricacies of its protocol in web browsers today.

Using similar ideas, we can also build on the Internet Key Exchange Protocol (IKE) protocols within IPSec, as specified in [105, 125], to establish secure channels between peers. Further traffic between peers can then be protected using IPSec's AH and ESP protocols in appropriate combinations. In the simplest version, we can use either of the generic key exchange mechanisms from Section 6.4.2.1 to establish a shared key, then use the shared key version of IKE Phase 1 to establish a secure Internet Security Association and Key Management Protocol (ISAKMP) channel over which further Security Associations can be exchanged. Alternatively, any of the IKE Phase 1 versions using public key techniques can also be supported. We can use the technique of extending PCRs, with measured data being replaced by the relevant public keys, followed by TPM attestations, to establish authenticated public keys at the communicating endpoints. This can then be followed by the appropriate version of IKE Phase 1, again using self-signed certificates to simplify certificate processing, and using DAA pseudonyms N_P and N_Q as initiator and responder identities. After this, IKE Phase 2 can be used to establish multiple further SAs containing keys for use in AH and ESP protocols between the peers.

Some care needs to be taken with these approaches to ensure that the either TLS or IKE, and TPM attestation protocol runs are bound in an appropriate cryptographic manner. We also have not sought to optimise the protocols to any great degree. Recent work in examining Trusted Computing and TLS has highlighted many of the issues faced in successfully binding security protocols with TPM functionality [58, 11]; much scope remains for developing these ideas further.

6.4.2.3 Implementation

In order to make our proposals for TPM-based authentication, access control and secure channel establishment mechanisms as concrete as possible, we give here an indication of the specific TPM commands that are needed to implement our ideas.

In order to understand the mechanisms for extending a PCR register we need to examine the event logging mechanism for Trusted Computing in greater detail. We will base our discussions on the assumption that our P2P application is running on a TCG-conformant platform. The general principles and commands discussed here are transferable to any TPM-enabled device. As we discussed in Section 2.4.5, the SML is not really a log. It is better to envisage it as an array of events in which each entry is in the form of a `TSS_PCR_EVENT` structure [149]. In a TPM-enabled device it is a `TSS_PCR_EVENT` that provides information regarding individual PCR extension events. The `rgbEvent` parameter in a `TSS_PCR_EVENT` is the one we are interested in, as it is a pointer to event information data, in our case a string formed from cryptographic parameters. The TCG Core Services (TCS) Event Log services are responsible for maintaining the SML. These services are responsible for allowing PCR extension events to be logged and allowing challengers interested in these events access to them.

To incorporate an event into a PCR two things must happen. A call to the

TPM_Extend command [154] is needed. This is the command that is responsible for extending a PCR. Then an additional call to a function such as Tcsi_LogPcrEvent is needed. This function is responsible for adding the new event to the end of the array associated with a named PCR [149]. When a challenger wishes to verify a system by examining one or more PCRs, the platform must perform a number of operations to satisfy this request. A call to Tcsi_GetPcrEventsByPcr returns an event log of all events in the SML that are bound to a single PCR [149]; this can be called multiple times depending on the number of requested PCR values. The event log is returned as an ordered sequence of TSS_PCR_EVENT structures. Attestation of the chosen PCR values occurs after a call to TPM_Quote [154]. This operation is responsible for providing cryptographic reporting of PCR values, using an AIK key to sign the current value of a chosen PCR.

The TPM_DAA_JOIN command [153] obtains the issuer's Camenisch-Lysyanskaya signature on the DAA secret f, while the TPM_DAA_SIGN command [153] is responsible for running the DAA-Signing algorithm and, through this, proving that the TPM in question does indeed possess valid credentials. By performing the TPM_DAA_SIGN command upon the AIK key used in the TPM_Quote command, a TPM can prove to the verifier that it is in possession of the private component of the AIK key that was used to sign a requested PCR value.

6.5 Two Approaches to Securing P2P Networks Using Trusted Computing

In this section, we investigate further the application of TPM functionality (particularly DAA) to securing P2P networks. We consider two distinct approaches here. In the first, we consider augmenting the security of today's "unmanaged" P2P networks by exploiting the presence of manufacturer-issued DAA credentials. The second is more tuned to existing commercial services for distributing content, but we sketch how secure P2P features can be added to these networks, giving potentially significant benefits to service providers.

6.5.1 Securing Unmanaged P2P Networks

As we have seen in the previous sections, DAA can form the basis for a number of security services in P2P networks in which peers are equipped with TCG-compliant platforms. Foremost amongst these services is pseudonymous authentication, from which other useful services (access control, reputation mechanisms, secure communications) can be built.

To achieve this, we have made the single, but vital, assumption that, in any given P2P network, the TPMs embedded in peers are guaranteed to only possess a DAA credential from one of a set of issuers that will be recognised and accepted by other peers. We stress that there can be many issuers of DAA credentials for the platforms in our P2P network, but we need to be sure that each platform only has one credential from one of the issuers from a specified subset of all possible issuers, and not multiple credentials from a single issuer or single credentials from each of multiple issuers. Otherwise, misbehaving peers will be able to create and prove possession of multiple pseudonyms.

As discussed above, one way of ensuring this assumption holds is to assume that manufacturers will provide DAA issuer functionality for the platforms that they produce. This seems to be an attractive supposition, since, in reality, customers would gain a privacy advantage if they were to obtain their credentials from their original equipment manufacturer. Either the DAA credential could be embedded into the platform at the time it is shipped to the customer, or it could be obtained at a later stage through a run of the DAA join protocol. A manufacturer can keep track of which EKs it has injected into the TPMs that it has produced and to which EKs it has already issued credentials. Since during the join protocol a TPM is authenticated with respect to its endorsement key EK, the issuer can be sure that it is dealing with a given TPM that it has manufactured and not previously issued a credential to.

If this situation holds, then we can configure our P2P software with a list of manufacturers' public keys used for issuing credentials (much in the same way that SSL is usually configured with a list of root public keys). Alternatively, manufacturers may choose to obtain certificates for their public keys from other certification authorities, in which case some form of PKI would be involved. Here we are assuming that the number of manufacturers of TPM-enabled platforms would be a small and relatively static group. This would allow an owner of a platform created by one manufacturer to transparently verify a DAA signature from a platform assembled by a different manufacturer while maintaining a high level of confidence in the authenticity of the end platform.

The obvious question is what does this mean for P2P networking? The traditional view of P2P networking is that it is an all-you-can-eat buffet of file sharing, where content flows as fast as connection bandwidth permits. At the same time, some believe that Trusted Computing represents a threat to this free-for-all as a tool for enabling Digital Rights Management (DRM) [5]. However, our analysis shows that the opposite may be the case. If TPM-compliant platforms become prevalent on end user systems, then the techniques developed here would make it expensive for copyright holders or their agents to engage in pseudospoofing and the flooding of networks with poor-quality content. With the development of suitable reputation mechanisms, the reputations of the offending peers would rapidly decline and they could be barred altogether. Our techniques would also enable users of P2P networks to maintain secure end-to-end communications, immune from monitoring by Internet

Service Providers or other entities. This could significantly hamper the prosecution of individuals operating in undesirable ways on P2P networks. Using simple access control lists based on pseudonyms, small groups could set up completely private, access-controlled subgroups within a given P2P network. Users could form their own groups based on content of choice or personal preferences, and so an existing P2P network could be utilised to bootstrap multiple private P2P networks. There is naturally the danger of such a feature being abused by certain groups. It would be difficult to prevent the formation of such groups, given that any platform can act as an issuer (and not just manufacturers).

6.5.2 Securing Commercial P2P Networks

There is an alternative model to the one presented above which is more commercially-orientated in application. In this model we reposition the root of trust for the issuance of TCG credentials from the platform manufacturer to a service/content provider.

We will begin by sketching a relatively simple example using a fictional application herein referred to as TrustedPeer. In this example we also make use of a fictional TTP named ContentCorp. In our scenario ContentCorp provides both the TrustedPeer application as well as acting as an issuer of DAA credentials.

Peers download the application from a ContentCorp server and install it on their TCG-enabled devices. In order for a peer to become a member of the network, it must register with ContentCorp. DAA can be used for the registration. A successful run of the DAA join protocol provides a peer with a credential in the form of a Camenisch-Lysyanskaya signature on its secret value f (see Section 2.4.5.2). This credential can later be used by the peer to authenticate itself to the service provider and to other peers through the DAA-Signing protocol. Thus this credential effectively becomes a proof of registration.

During this registration procedure the platform must reveal a pseudonym $N_I = \zeta_I^f$. Here the quantity ζ_I is set by ContentCorp and allows them to associate a handle with a peer as well as to check for rogue platforms. The handle may be used later by ContentCorp, for example at the stage of collecting payment for access (in which case, it may be necessary to collect additional payment details at the time of registration). Once this stage is complete a peer can gain access to the TrustedPeer network.

So far, the approach we have described is broadly similar to a Trusted Computing application scenario described in [28]: we have a centralised infrastructure making content available to subscribing peers. The benefits this approach offers are twofold. The benefit to service providers is that only platforms that have a credential issued by the service provider, i.e. that are registered, will be permitted access to the network and its content. It benefits registered participants insofar as other platforms will not be able to impersonate them and gain access to content that the registered participants have paid for.

This type of content distribution application could start to evolve into a more decentralised P2P architecture by allowing peers to obtain content from other peers as the content spreads out from ContentCorp to subscribing peers. In this scenario a peer looking for specific content sends out a query onto the network, receives a list of matching hits, examines them for relevance and then decides which peer to use to download the required content. In order that peers can distinguish genuine ContentCorp content from rogue content, ContentCorp could digitally sign its content and embed their public key in the TrustedPeer application. As more and more content becomes distributed, query results for content could start to be returned by other peers on the network, thus reducing the load at ContentCorp servers.

The TrustedPeer application could be configured to demand that the requesting peer provide a proof (in the form of a Camenisch-Lysyanskaya signature) that it is a peer registered with ContentCorp before supplying the requested content. It could be further configured to supply the requesting peer with the content in an encrypted form, with the session key used for encryption further encrypted under ContentCorp's public key, perhaps along with the responding peer's pseudonym and some metadata describing the content. It would then be incumbent on the requesting peer to contact the ContentCorp server to obtain the session key, at which point ContentCorp could collect payment for the content. Here, the communication with ContentCorp would involve only short messages and low bandwidth rather than the high bandwidth required for content distribution. Further, ContentCorp would have visibility of which peers were supplying content, and those peers could be rewarded appropriately (perhaps with reduced fees for subsequent downloads). Thus ContentCorp could generate revenue from its content without being directly involved in its distribution. Such models of distribution predate this work and are becoming increasingly popular with content providers. In particular, this model of distribution underlies the open mobile alliance digital rights management model of content distribution [93].

ContentCorp's content distribution could be further enhanced by adding a reputation system, enabling peers to search out peers with fast connections or rich content. Building on the stable pseudonyms that are enforced by the DAA protocols, such a mechanism would naturally resist pseudospoofing and pseudotheft attacks. ContentCorp could also keep state regarding which EKs have already been registered in order to prevent multi-credentialised platforms joining the network.

The security measures that we present here are not intended to provide a solution to all the possible problems that may be potentially encountered in a real world setting, but they do address some of these issues, such as discouraging bad behaviour by peers.

One important issue that remains to be addressed is the prevention of "leakage" of content from the ContentCorp network. There is nothing in our system as described so far that prevents a peer registering, obtaining content, and then re-distributing it on traditional, anarchic P2P networks. There is a form of economic incentive to prevent this activity: a peer, having paid for the content, may be less likely to distribute it freely to others. Moreover, a peer who knows that ContentCorp will give him credit of some kind if other peers download content from him, is more likely to conform and use the encrypted session key mechanism that we have described above. In turn, this forces peers wishing to obtain content to pay ContentCorp for it.

A technical measure that could further discourage unrestricted re-distribution of content would be to use watermarking, marking content with the requesting user's DAA pseudonym before distribution. In the event that a peer offers content from the TrustedPeer network on another network, then the peer could be identified via the watermark and potentially permanently banned from the TrustedPeer network.

A far stronger mechanism would be to rely on the ability of a TPM-enabled platform to lock availability of content to particular software configurations: with this in place, ContentCorp could be assured that the correctly behaving TrustedPeer application is in operation before content is available at a peer. This would limit a rogue peer's ability to re-distribute content. However, this kind of mechanism would depend on the extension of the domain of control of TPM from the OS level to applications themselves. This is certainly something we expect to see in the years ahead.

6.6 Issues and Open Problems

We briefly summarise some issues with our approach to securing P2P networks using Trusted Computing. We also highlight some open problems and areas suitable for further development.

6.6.1 Issues

6.6.1.1 Credential Replacement

Replacement of credentials is a non-trivial task in DAA. If it is necessary to issue replacement credentials, then the previous DAA history of the platform gets erased, at least as far as the DAA credentials from that issuer are concerned. This potentially allows a rogue platform to rejoin a system from which it has previously been barred. However decisions on credential replacement are a matter of policy for a given issuer. For example, an issuer can consult a black list before issuing a credential to a platform.

6.6.1.2 Credential Migration

Migration of credentials is not possible within our architecture. Should a platform owner decide to sell their platform there is no obvious way for them to migrate DAA credentials from one platform to another.

6.6.1.3 Data Representation

Another issue in our approach to securing P2P networks is that of data representation. There is something of a semantic gap between dealing with a DAA pseudonym (a sequence of ones and zeros) and working with the more traditional “handles” to which users of P2P networks are accustomed. One possible solution to this problem is to link a local name space with a network name. If a peer on a particular network maintains a list of pairs (ζ, ζ^f) then with each of these pairs it can associate a particular name or handle. In this scenario ζ represents the network and ζ^f a particular pseudonym on that network. This mapping would be akin to that used in SPKI/SDSI[11], in which every principal maintains a name space where names are bound to public keys, possibly in other principal’s name spaces. This kind of mechanism is suitable for use in a P2P network as it is not necessary for peers to know every other peer on a network. Instead, they need only know those peers with whom they have previously interacted or are intending to interact. In our application, we simply replace the public keys seen in SPKI/SDSI with pairs (ζ, ζ^f).

6.6.2 Open Problems

6.6.2.1 Reputation

One area of our work where further development is possible is that of reputation systems for TPM-enabled P2P networks. For example, it may be possible to set up a recommendation system whereby peers pass on their experiences to other peers by sending a subset of their named handles to other peers following a SDSI naming convention. In this case peers could provide DAA-signed reference subsets based on successful interactions with other peers. It would be interesting to see various

[11]See for example `http://www.ietf.org/html.charters/spki-charter.html`

established rating schemes examined in the context of TCG-based P2P networks: it would seem that they could be far more robust than is currently the case in standard P2P networks. Given the network based unlinkability property established in Section 6.3, developing the ability to securely transfer a peer's pre-existing reputation in one network to another seems like an interesting challenge.

6.6.2.2 Confidentiality and Integrity of stored data

We have mainly concentrated on confidentiality and integrity of data *in transit* in P2P networks. A TCG-enabled platform provides *sealing* mechanisms which, in combination with attestation, can provide confidentiality and integrity for stored data. Such secure storage mechanisms could be exploited to further enhance the security of P2P networks. In [122], Zhang *et al.* consider the use of a TPM with processor and OS support to build a Trusted Reference Monitor (TRM) that could mediate access to data in P2P networks. By monitoring and verifying integrity properties of executing (P2P) applications, the TRM could enforce access policies on behalf of object owners.

6.6.2.3 Anonymity

Anonymity challenges other security goals within P2P networks such as accountability and reputation. DAA provides a certain degree of anonymity in the form of unlinkability of transactions arising from a user, but by fixing the name base (ζ) in our approach we have restricted the provision of such anonymity. However, anonymity within multiple networks is still present as long as different name bases are used in the different networks. Providing complete anonymity would bring with it problems relating to routing and discovery of resources, but it would be worthwhile continuing research in this area.

6.6.2.4 Authenticity

Another issue within anarchic P2P networks is the question of how to guarantee the authenticity of resources. Considerable work has been done within the research community to establish certain measures of authenticity. In the P2P context, Daswani *et al.* [36] have proposed evaluating authenticity based on expert opinion, polls or first-serve basis. In our work, we have not identified authenticity as an important security issue in P2P networks, though we briefly touched on it in Section 6.2.1.1. Our work in Section 6.5.2 does provide an explicit authenticity mechanism, in the form of digital signatures on resources by the content provider. By preventing pseudospoofing attacks in anarchic P2P networks, our methods also go some way towards ensuring resource authenticity in such an environment. It would be interesting to explore further how resource authenticity could be enhanced using TCG in such networks.

6.6.2.5 Complexity

On average, a TPM takes between one and three seconds to perform a signature with a TPM-controlled key [119, 138]. Therefore, an excessive number of signature operations may result in a performance bottleneck at the TPM and cause the host platform to drop peer requests [138]. In our proposal for establishing secure channels, the DAA sign and verify operations are performed once by each peer in establishing a connection. After this initial handshake, peers migrate to use of a shared Diffie-Hellman key for further communications. The relative slowness of the initial sign/verify operations may have a negative impact upon some time-sensitive applications; however, in certain P2P networks, for example P2P instant message networks, peers typically restrict their interaction to a specified subset of the entire P2P community. This would allow the potential cost associated with participating in the network to be amortized over time.

6.7 Conclusion

In this chapter we have outlined the security issues faced in P2P networks and discussed the extent to which features of the TPM specifications can be used to enhance security. All but one of our proposals for enhancing security use only those TPM features which are in place and available on platforms today. We have applied our ideas to two distinct types of P2P network: one matched to today's file sharing networks, and the other more commercially-oriented. One perhaps counter-intuitive implication of our work is that TPM-enabled P2P networks can make P2P networks harder to effectively police. Our work also points the way forward to using Trusted Computing to enable secure P2P-commerce through the development of secured commercial P2P networks and of P2P networks free from pseudospoofing and pseudotheft.

Chapter 7

Protecting Mobile Agent Migration with Trusted Computing

Contents

Protecting Mobile Agent Migration with Trusted Computing

In this chapter we discuss the possible benefits (and limitations) of incorporating Trusted Computing technology into the mobile agent paradigm. In particular, we discuss how Trusted Computing technology can be used to enhance the security of mobile agent migration. We describe a number of options for securely migrating mobile agents between Trusted Computing platforms and compare and contrast the merits (and deficiencies) of each approach.

7.1 Introduction

Before we begin to define the security concerns pertaining to mobile agent migration and the role Trusted Computing can play in ameliorating them, it is important that we first define what we mean by a "mobile agent". Unfortunately, an examination of the literature yields no universally adopted definition. Instead an "agent" can be defined as a software construct that satisfies a number of properties. That is, an agent should be autonomous, persistent, adaptive, responsive, pro-active and social [117]. These properties imply a monocratic goal-driven entity that is capable of interacting with its environment (and other agents) in order to achieve its objectives. Consequently, the term "mobile agent" represents a superset of these autonomous software entities that possess one additional property, the capability to suspend its execution mid-process and transport itself (and its current data and state) to another platform where it can resume execution.

Given the broad set of properties a mobile agent may possess, the mobile agent paradigm offers numerous deployment scenarios that may exploit one or more of these properties. As a result, mobile agent systems have been proposed for use in areas as diverse as information management [62], contract negotiation [61], service brokering [44] and e-commerce applications [75]. However, precisely because of the flexibility that the mobile agent paradigm affords, both mobile agents, and the hosts upon which they execute, may be placed at greater risk of attack than traditional client-server architectures [74]. For example, from the host's perspective, the execution of a mobile agent may potentially lead to information leakage or denial of service attacks. Conversely, from the mobile agent's perspective, its very mobility places it at increased risk from attack by malicious hosts. As a host controls the environment in which an agent executes, it can engender a number of security sensitive violations including eavesdropping on a mobile agent's communications, altering a mobile agent's state, or preventing a mobile agent from performing its task.

In this chapter we examine the extent to which Trusted Computing can be used to enhance the security of mobile agents as they migrate between platforms. In doing so, we describe an agent framework that makes use of the full array of Trusted Computing technologies. In particular, we assume each host has the following functionality: measurement, storage and reporting capabilities as provided by a Trusted Platform Module (TPM) (see Section 2.4), strong process isolation as provided by processor/chipset extensions such as Intel's LaGrande or AMD's AMD-V and Operating System support for Trusted Computing as provided by Microsoft's NGSCB or EMSCB derivatives [121] (see Section 2.7). In the remainder of this chapter, we refer to such a host as Trusted Mobile Agent Platform (TMAP).

The remainder of this chapter is organised as follows. In Section 7.2, we briefly outline a number of threats to both mobile agent hosts and mobile agents. We also review previous work in the area of mobile agent and mobile host security as well in the application of Trusted Computing to agent systems. In Section 7.3, we discuss a number of options for securely migrating agents between TMAPs and discuss the relative merits and deficiencies of each approach. Finally, we conclude with Section 7.4.

7.2 Mobile Agent Security

The threats typically attributed to mobile agent systems tend to be context-driven but can broadly be categorised into threats to the host and threats to the agent [74, 26]. In this section, we outline a number of threats posed both by mobile agents and to mobile agents as well as outlining a number techniques proposed in the literature to mitigate one of more of these threats. These threats and countermeasures are compiled from lists obtained from [74, 26].

7.2.1 Threats To The Host

- **Unauthorised Access:** A mobile agent may attempt to exploit a vulnerability in a host system to gain access to resources that it is not permitted access to.
- **Denial of Service:** A mobile agent may attempt to appropriate the (limited) assets of a platform by unfairly consuming excessive system resources. Alternatively, a mobile agent may cause a host to crash by altering a system's configuration files or exploiting some vulnerability in the host platform.
- **Masquerade:** A mobile agent may pretend to be another (possibly more privileged) mobile agent to gain access to additional system resources or to shift the blame for malicious actions it has perpetrated to another mobile agent.
- **Eavesdropping:** A mobile agent may attempt to eavesdrop on the communications of other agents executing on a host platform.
- **Alteration:** A malicious (or incorrectly configured) mobile agent may alter a host's system or configuration files. The outcome of this alteration may lead to a denial of service or platform subversion.

7.2.2 Threats To The Mobile Agent

- **Unauthorised Access:** A malicious host may attempt to modify an agent's code, data or state. Modifications to an agent's code may affect how that code behaves on subsequent platforms that an agent visits, whilst modifications to an agent's data or state may affect the results obtained during execution on subsequently visited hosts.
- **Denial of Service:** A malicious host may deny resources to an agent, introduce unacceptable delays in processing, refuse to execute an agent or prematurely terminate an agent's execution.
- **Masquerade:** A malicious host may pretend to be another (possibly more trustworthy) host to gain access to an agent's sensitive data or state. Additionally, a host may alter its execution environment so that it appears more trustworthy to an agent.
- **Eavesdropping:** Since the host controls the environment in which the agent operates, the host may eavesdrop on all operations performed by an agent.
- **Alteration:** A malicious host may modify the code, data or state of a mobile agent. This may have a knock on effect for subsequent platforms on an agent's itinerary as a subverted mobile agent may attempt to subvert other platforms or agents. Maliciously modified agent code could result in the realisation of any of the threats described in Section 7.2.1.

7.2.3 Safeguards for Protecting the Host

The problem of a host assigning correct access permissions to a mobile agent is a non-trivial matter, especially given difficulties in establishing agent provenance. This is compounded by the fact that an agent could start out as trustworthy only to be compromised by a malicious platform prior to arriving at its destination. From a hosting environment's perspective, a mobile agent represents an arbitrary and potentially malicious piece of code that may compromise it. In order to limit the threats discussed in Sections 7.2.1, a number of mitigating proposals have been described in the literature. A sample of these are included below:

- Code signing: This is a mechanism that aims to provide evidence of code origin. A code producer digitally signs a piece of code using a private key for which the corresponding public component has been certified by a trusted third party. Under the assumption that the private key has not been compromised, and that the verifier has an authentic copy of the code producer's public key certificate, the verifier can authenticate a piece of code as coming from a particular code producer. Unfortunately, this approach says nothing as to how the code will behave once executed. Examples of code signing include signed Jars [140] in Java and Microsoft's Authenticode for ActiveX controls [63].
- Proof Carrying Code [90]: This mechanism necessitates the generation of a proof by the code author that formally demonstrates that agent code conforms to some defined security policy. The executing host can then check the correctness of the proof prior to allowing the agent to execute, avoiding expensive runtime checks. The main difficulty of this approach is that there is currently no efficient and automated way of generating proofs for arbitrary pieces of code [74].
- Path Histories [136]: The idea of this approach is to maintain a protected log of the hosts an agent has previously visited. The current host platform, after examining an agent's log, determines if an agent has previously executed on one or more hosts that the examiner deems untrustworthy. Based on this evaluation, the hosting platform decides an agent's execution privileges.
- State Appraisal [50]: This mechanism requires an *a priori* generation of a

signed state appraisal function that becomes part of an agent's code. The goal here is to detect subversion of an agent through modifications to its state information. The executing host uses the signed function to determine if an agent is in a correctly functioning state and to assign access privileges accordingly.

- Software-Based Fault Isolation [163]: This class of protection mechanisms refers to partitioning applications into software-enforced isolation domains with a reference monitor controlling access to system resources. Perhaps the best known example of such a fault isolation scheme is the Java sandbox for safe code interpretation [59]; other schemes exist for script-based languages such as Safe TCL [96] for the TCL language.

Many of the individual mechanisms presented here can be bolstered by Trusted Computing. In particular, path histories can be enhanced by Trusted Computing's attestation procedure. As we will see in Section 7.3, attestation and sealing can be used to provide a means of testing the "trustworthiness" of both the current platform in which the mobile agent is executing as well as the target platform to which the agent wishes to migrate. Additionally, software-based fault isolation techniques can be significantly improved by the inclusion of hardware-enforced isolation environments (see Section 2.7). A malicious agent attempting to attack a host platform can be constrained to an individual isolated compartment.

7.2.4 Safeguards for Protecting the Agent

Intuitively, the probability of a mobile agent being compromised by a malicious host increases in proportion with the length of an agent's itinerary. The more hosts an agent visits, the greater the likelihood that one of those hosts will attempt to subvert the agent. In order to prevent (or at least limit) an attack against a mobile agent, a number of proposals have been put forward in the literature that address one of more of the threats identified in Section 7.2.2:

- Code signing [140]: This mechanism allows a code producer to generate a signature over the non-mutable elements of his mobile agent's code, data and state. A verifier, in turn, can detect if a mobile agent's signed non-mutable code, data or state elements have been tampered with after the mobile agent has been signed.
- Partial Result Encapsulation [167]: This approach attempts to detect modifications to an agent by recording the results of an agent's actions at each visited platform. Unfortunately, as the agent is executing in an environment controlled by the host, the host can control what the agent records.
- Cooperating Agents and Mutual Itinerary Recording [116]: In this approach a function can be divided amongst a number of agents that visit a disjoint set of hosts. In moving between platforms, an agent communicates its itinerary over an authenticated channel to a cooperating peer. The peer agent, in maintaining a record of its friend's itinerary, can take appropriate action when inconsistencies are noticed. A limitation of this approach is the inability of the peer to determine whether it is the current platform that an agent is visiting or the next one on its itinerary that is responsible if the agent is "killed".
- Execution Tracing [158]: This proposal attempts to detect unauthorised modifications of an agent's state. When an agent executes in a host environment, a record is kept of the actions that the agent performs. This record is appended to the agent's past actions for every subsequent host that an agent visits. Unfortunately, this approach is also vulnerable to the type of attack presented for Partial Result Encapsulation.
- Code Obfuscation [34]: This technique modifies source code to obscure the agent's true function and prevent reverse engineering. However, obfuscated

code often depends on the particular characteristics of the host platform or compiler, making it a difficult technique to manage if either change. Furthermore, general impossibility results have been reported for obfuscation as a protection technique [23].

- Environmental Key Generation (EKG) [115]: In this approach, keying material is generated from certain types of environmental data. An agent carries secret code or information that can only be revealed to a host if certain pre-defined environmental conditions are met. Only once environmental triggers are satisfied will an agent be capable of running its secret function.

For the interested reader, additional information on both agent and host defence mechanisms can be found in [25, 74].

7.2.5 Trusted Computing and Mobile Agents

The proposed use of trusted hardware as a means of protecting mobile agents can be traced back to Wilhelm *et al.'s* work on adding trusted third parties (in the form of isolated hardware environments) to host systems [166]. As part of their proposal, Wilhelm *et al.* define a Trusted Processing Environment (TPE) to consist of a CPU, RAM, ROM and non-volatile storage, all of which are accessible via a secure virtual machine interface. It is within this TPE that mobile agents will execute, free from interference and observation by external processes. In addition to hardware requirements, Wilhelm *et al.* also stipulate that each TPE should have a manufacturer-certified public key, indicating that the hardware environment is trustworthy, with the corresponding private key being non-migratable from the TPE environment.

As part of their work, Wilhelm *et al.* define a protocol for secure mobile agent migration. Their protocol operates in two phases. In the first phase, each TPE registers a copy of its certified public key and a description of the policy that it is currently enforcing with a resource broker. In the second phase, the owner of a mobile agent queries a resource broker for a list of available TPEs. In response to this query, the broker returns to the owner a list of all TPEs including their associated public keys and security policies. The mobile agent owner examines this list and picks the TPE with the policy most closely aligned with his security requirements. Once selected, a mobile agent owner encrypts his mobile agent with the selected TPE's public key and migrates his mobile agent to that platform where it can execute unhindered.

Unfortunately, this approach to secure agent migration suffers from two problems. Firstly, it assumes that the broker is completely trusted to return correct information to the mobile agent owner. Secondly, there is no way to validate the veracity of the claims made by a TPE about its enforcement of a given policy. Instead all claims made by a TPE must be taken at face value. Without the ability to test a platform for adherence to a claimed policy prior to migration, any security claims made by a TPE are merely anecdotal.

The use of Trusted Computing in agent systems has been put forward in [32, 98, 101, 107]. In [98], a method of using trusted agents running on TPM-enabled platforms is described where the agent controls access to a user's profile data through the use of a TPM's sealing mechanism. That is, access to personal information should only be granted if the platform is in a specified state. In [32], a similar architecture to [98] is proposed to constrain access to sensitive resources on host platforms. In [101], a number of proposals are made to enhance user privacy protection mechanisms using Trusted Computing in ubiquitous computing environments. Each proposal stresses the need for sealed storage functionality and the benefits that accrue from the ability to recognise when a platform will behave as expected.

However, these approaches do not address issues of agent migration.

Recently, a Trusted Computing enhanced mobile agent platform called SMASH was proposed [107]. In SMASH, an agent arriving at a remote host goes through a process of mutual attestation with its hosting environment. If the agent's attestation passes the host's security policy, and the host's attestation passes the agent's security policy, then the mobile agent can be granted access to limited system resources. Further to this, if both the mobile agent and host platform are capable of mutually authenticating each other then further access can be granted to protected resources. Unfortunately, this approach suffers from a major problem. In the SMASH framework, a mobile agent only requests the host platform to attest to its environment *after* it has arrived at the platform. At this point it may be too late to request a report of a host's environment as the host has complete execution control of the agent, which may result in the realisation of any or all of the threats discussed in Section 7.2.2.

7.3 Protecting Mobile Agent Migration with Trusted Computing

In this section we look at how Trusted Computing functionality can be used to protect sensitive agent information (be it code, data or state) as it migrates between TMAPs. In doing so, we present four alternatives for migrating mobile agents and discus their relative merits and drawbacks.

7.3.1 Assumptions

Before we outline the various alternatives, we first discuss the assumptions that we make about the environment in which a mobile agent will operate and provide justification for these assumptions.

1. **TMAP Functionality**: Each hosting platform will provide mechanisms to measure and report current platform state, seal data to a particular instance of a platform state, and provide hardware enforced isolation between environments in which agents execute.

2. **Authenticated Boot**: Upon platform start-up, reset or the launch of a new secure compartment via the Dynamic Root of Trust for Measurement (DRTM), the software state of the platform is measured, and these measurements are accurately stored in the TPM's PCRs.

3. **AIK Credentials**: Every TMAP platform has enrolled at least one AIK with a Privacy-CA known to every other TMAP platform.

4. **PCR Usage**: The use of PCR registers to store measurements representative of a platform's software state are consistent across all platforms.

5. **Agent Owner Signing Key**: Each mobile agent owner has a non-migratable, TPM-controlled asymmetric key pair, where we write $K_{AO_{pub}}$ and $K_{AO_{priv}}$ for the public/private portions respectively. The public key of this key pair, $K_{AO_{pub}}$, has been certified by an SKAE CA (see Section 2.4.5.4).

6. **Itinerary Planning**: Prior to deployment, the owner of a mobile agent populates his agent with a list of platforms he would like his mobile agent to visit. Additionally, a mobile agent may optionally take recommendations from hosts or other agents during its travels and visit additional platforms not on its itinerary prior to returning to its owner.

7. **Trustworthy States**: A mobile agent owner has access to integrity metrics representing platform states which guarantee faithful execution and forwarding of his mobile agent. These integrity metrics will typically be provided in the form of validation credentials representing trustworthy TMAP states (see Section 2.4.6).

Assumptions 1 and 2 are general requirements if a platform is to be considered a Trusted Platform. These assumptions are realised through the inclusion of a TPM, processor, chipset and operating system support for Trusted Computing. Assumption 3 is necessary for establishing trust in the attestation claims made by a third party Trusted Platform (see Section 2.4.5). Assumption 4 is implicit in all Trusted Computing work. Certain PCR registers must only be used for a specific purpose so that there is commonality amongst all attestations reported by a disparate group of Trusted Platforms. For instance, PCR 0 is set aside for CRTM, BIOS, and Host Platform Extensions in the TCG's PC client specifications [148]. Assumption 5 is necessary so that an agent owner may sign his mobile agent. Like the code signing approach outlined in Sections 7.2.3 and 7.2.4, we stress that signing mobile agent code does not guarantee correct execution. Instead code signing provides a means of appropriating blame should the mobile agent prove malicious. Assumption 6 is a standard assumption in the mobile agent literature. As the agent originator is assumed off-line for all purposes but agent deployment, the agent owner needs to specify the platforms he would like his agent to visit. Assumption 7 is a new, but necessary, assumption for Trusted Computing augmented mobile agent environments. It is impractical to assume that the agent owner would have sufficient domain expertise to evaluate the trustworthiness of platform states. Instead, we require that the metrics representative of trustworthy platform states are independently evaluated by a trusted third party, and that the outcome of such evaluations be published in the form of validation credentials.

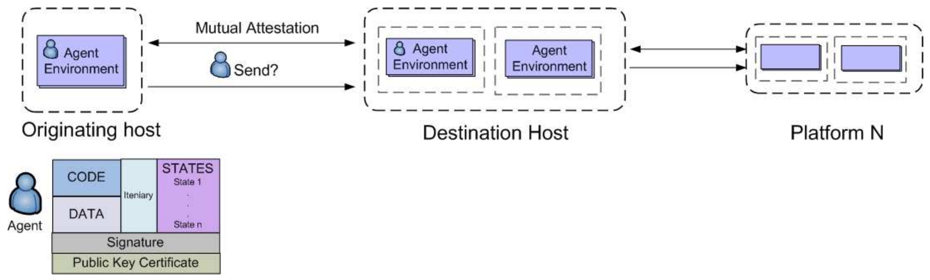

Figure 7.1: Mutual Attestation for Mobile Agent Migration

With these assumptions in place, we now discuss four alternatives for secure agent migration.

7.3.2 Mutual Attestation

Our first option for secure agent migration (see Figure 7.1) occurs in two stages.

Pre-Deployment: Prior to deployment, a mobile agent owner populates his agent with a list of software states that he deems acceptable for his mobile agent to execute within. For each item in this list, the mobile agent owner extracts trustworthy states from his validation credentials which he uses to specify one or more (PCR index, acceptable value) pairs for his mobile agent. The mobile agent's code, the set of acceptable software states, the mobile agent's itinerary and any static data elements of the mobile agent are signed by the mobile agent owner using his private key $K_{AO_{priv}}$. Finally, the mobile agent owner attaches his newly generated signature and his public key certificate for $K_{AO_{pub}}$ to his mobile agent (see Figure 7.1).

Deployment: The mobile agent's originating host and the destination host mutually attest to their current operational environments (see Figure 7.1). During this process, both platforms call their TPM_GetRandom command to generate a random number. This random number and a list of the indices of the PCRs to be reported are then exchanged. Both TMAPs attest to the current configuration of their respective platforms (more specifically, the values of the specified PCR indices) by calling the TPM_Quote command which outputs an AIK signed data structure consisting of the following elements: the indices of the PCRs being reported, the values of each of the specified PCRs, and the random challenge received from the other platform. These signed data structures, along with the respective TMAP's AIK credentials, are then exchanged.

If the originating host deems the destination host's state trustworthy (by comparing it to the signed list of trustworthy software states obtained from the mobile agent, verifying the destination's AIK signature and checking for the inclusion of its random number), then the mobile agent is migrated to the destination host. Upon receipt of the mobile agent, the destination host validates the public key certificate attached to the mobile agent, extracts the public key and verifies the mobile agent owner's signature. Additionally, the destination host queries the agent for a list of trustworthy software states and compares this list to the attested state returned by the originating platform. If a match occurs, then the destination host executes the agent in its own (attested) isolated compartment. If no match can be found, the destination host reports back to the mobile agent owner that something went awry on the originating host, providing the reported attestation as evidence. Once the mobile agent finishes executing, the above process is re-run with the current host acting as the originating host and next host on the agent's itinerary as the destination host.

Analysis: We will now examine how well this proposal prevents the threats identified in Sections 7.2.1 and 7.2.2.

- **Unauthorised Access:** From the host's perspective, the use of hardware-enforced compartmentalised memory limits the opportunity for a malicious agent to access resources outside of this compartment. Any additional privileges requested by the mobile agent will be subject to approval by an external Measured Virtual Machine Monitor (MVMM).

  From a mobile agent's perspective, allowing the originating host to examine the state of the destination host prior to migration ensures that, at the time of migration, the destination host is in a trustworthy platform state. Unfortunately, in isolation, this attestation only enables the state of the destination host to be verified at a particular point in time. There are no guarantees that the destination host will remain in its attested state after it has been verified but prior to the mobile agent arriving. This problem is commonly referred to as the Time-Of-Check-To-Time-Of-Use (TOCTOU) problem.

  Our use of mutual attestation mitigates this TOCTOU threat provided that destination host $N+1$ compares the attestation of destination host N to that of the signed trustworthy states obtained from the mobile agent. If destination host N's state matches a trustworthy state (obtained from the mobile agent) then host $N+1$ can be assured that host N's state remained stable whilst the mobile agent executed. However, this approach only works if host N engages in a mutual attestation process with destination host $N+1$ on the mobile agent's itinerary. Should a host N be compromised after passing the originating host's attestation challenge and receiving the mobile agent, the newly compromised host N may attempt to attack the mobile agent to gain access to its sensitive code, data or state and refuse to propagate the mobile agent further. We note, however, that if the trustworthy states obtained from the validation authority prior to the initial migration are sufficiently robust then the likelihood of this occurring should be negligible. For example, if the hosting compartment has been evaluated to a sufficiently high common criteria evaluation assurance level then the attack surface of the compartment should be limited enough to prevent post-attestation compromises.

- **Denial of Service :** From the host's perspective, a malicious mobile agent may attempt to modify the configuration files for its compartment resulting in that compartment failing future attestation challenges. This issue is mitigated somewhat by allowing a host to "take down" a corrupted compartment and replace it with a clean one. However, in doing so the host may lose useful configuration data obtained during the life-time of the now corrupt compartment. From a mobile agent's perspective, denial of service remains a problem. A host may simply suspend execution, degrade the quality of service available to a mobile agent or refuse to migrate the mobile agent to the next host should the compartment become corrupted. However, for the reasons outlined in our analysis of unauthorised access, the likelihood of this happening should be low.
- **Eavesdropping:** From the host's perspective, the use of compartmentalised memory limits an agent's opportunity to eavesdrop upon the host, because the mobile agent will be constrained to an individual execution environment. From the mobile agent's perspective, the host will always be capable of eavesdropping upon the agent. However, agents executing in other compartments will not be able to eavesdrop on other mobile agent's communications unless expressly permitted to do so by the MVMM.
- **Alteration:** From the host's perspective, a mobile agent may alter the compartment in which it executes. However, such modifications will be exclusively limited to this compartment and will be detected in subsequent attestation challenges. From the mobile agent's perspective, the host is capable of altering the mutable portions of the mobile agent's code, data or state that are not part of the mobile agent signature. Any attempt to modify the immutable portions of the agent's code data or state will result in subsequent mobile agent signature verification procedures failing. However, the likelihood of a

host modifying mutable portions of the mobile agent should be low for the reason cited in our analysis of unauthorised access.

- **Masquerade:** From the host's perspective, the mobile agent can be unambiguously identified by verifying the mobile agent's signature using the public key obtained from the mobile agent's public key certificate. From the mobile agent's perspective, the use of mutual attestation may be vulnerable to proxy relay attacks (also know as man-in-the-middle attacks) allowing one host to masquerade as another host. For example, a destination host may attempt to forward the attestation challenge from the originating host to a TMAP which the destination host knows to be in a good state. The outcome of this redirected attestation challenge is then passed back to the originating host via the destination host. An originating host seeing this attestation concludes that the destination is indeed in a trustworthy state and migrates the mobile agent. We note, however, that this attack can be easily addressed using the attestation mechanism presented in [139], which build directly upon our earlier work presented in Chapter 6 in establishing secure channels with known endpoints in P2P networks. That is, in addition to exchanging random numbers, the originating and destination hosts also exchange ephemeral Diffie-Hellman values g^{x_O}, g^{x_D} (where g is a generator of a suitable group negotiated in advance). Each host then computes a key $K = g^{x_O x_D}$, concatenates this with the exchanged TPM_GetRandom value, and hashes the result. The output of this hash is then fed into their respective TPM_Quote commands as the random challenge. Upon receipt of their respective attestations, both platforms check for the inclusion of their random challenges by hashing K with their random number. If this challenge is present in the received attestion, the originating platform derives a symmetric encryption key from K and encrypts the mobile agent with this key prior to migration. Thus, only the authenticated, attested destination platform will be able to decrypt the agent.

7.3.3 Sealing

Our second option is based upon the key exchange protocol defined in [54, 88]. In this option, as per the pre-deployment phase presented in Section 7.3.2, a mobile agent's static code, data, state, its itinerary and a list of trustworthy software states are all signed by the agent owner using his private key $K_{AO_{priv}}$ prior to deployment. This signature and the agent owner's public key certificate are attached to his mobile agent.

Prior to migrating a mobile agent, the originating host sends a list of candidate platform states (as extracted from the mobile agent's list of trustworthy states) to the destination host. If the destination host is capable of configuring itself to match any of the received candidate platform states, the destination host generates a non-migratable asymmetric key pair, where the private component of this key is sealed to one of the selected platform states (see Section 2.4.4.2).

To generate this key pair the destination host calls its TPM_CreateWrapKey command to create a non-migratable key with specified state constraints for private key usage as input. The destination host next calls TPM_CertifyKey using his private AIK key (for which the corresponding public part has been certified) to generate a signed data structure containing the public key from the newly generated key pair and a description of the state to which the corresponding private key is bound. This data structure, along with a random number (as per our mutual attestation approach) is returned to the originating host on which the mobile agent resides. Once this response has been received, the AIK credential of the destination host is validated, the signature of the destination host on the returned data structure is verified, and the state reflected in the PCRs to which the key is bound is compared to the values stored within the mobile agent. If a match is found, the originating host attests to its current state (with the random number returned from the destination host as input), encrypts the mobile agent with the destination hosts's certified public key and sends both the host attestation and the encrypted mobile agent. As per our mutual attestation based approach, upon receipt of the mobile agent, the destination host validates the public key certificate attached to the mobile agent, extracts the public key and verifies the mobile agent owner's signature. Additionally, the destination host queries the agent for a list of trustworthy software states and compares this list to the attested state returned by the originating platform. If a match occurs, then the destination host decrypts the mobile agent and executes it in its own (attested) isolated compartment. If any of the above checks fail, the originating host selects the next host in the mobile agent's itinerary and reruns the above process. This process continues on all subsequent platforms until the agent achieves its goal or the agent reaches the end of its itinerary.

Analysis: We will now examine how well this proposal prevents the threats identified in Sections 7.2.1 and 7.2.2.

- **Unauthorised Access:** From the host's perspective, this approach provides similar guarantees to our mutual attestation approach. However instead of the *explicit* destination host attestation provided in our mutual attestation approach, we replace the destination host's attestation with an *implicit* attestation in the form of sealing the mobile agent. From the mobile agent's perspective, the use of sealing in this context prevents the mobile agent from being decrypted and executed on the destination host if the destination host is not in a specified trustworthy state. This approach thus limits the TOCTOU problem discussed in our mutual attestation approach.
- **Denial of Service :** The threat of either host or mobile agent denial of service is identical to that in our mutual attestation based approach.
- **Eavesdropping:** The threat of either host or mobile agent eavesdropping is

identical to that in our mutual attestation based approach.

- **Alteration:** The threat of either host or mobile agent alteration is identical to that in our mutual attestation based approach.
- **Masquerade:** The threat of either host or mobile agent masquerade is similar to that in our mutual attestation based approach. However, instead of introducing an additional Diffie-Hellman channel to prevent proxy relay attacks, our use of sealing directly addresses this issue. The use of a non-migratable key allows only the identified destination platform to decrypt the mobile agent post-migration.

7.3.4 Environmental Key Generation

A sealed object in Trusted Computing is uniquely bound to a single TPM instance. It is not possible for a sealed object to be opened on a platform that did not originally seal that object. In our sealing-based proposal in Section 7.3.3, we outlined an approach that would allow a host to seal an object to a destination platform. However, such a sealing mechanism merely represents a one-hop extension to the TCG's native sealing approach. For every host an agent visits, our sealing approach would require the current host to encrypt an agent with a public key (for which the private key is bound to an agreed state) obtained from the next host in a mobile agent's itinerary. In certain environments it may be preferable to have a generic sealing mechanism that allows an object to be sealed to a class of platforms, rather than being sealed to an individual platform.

The solution we present here extends the concept of Environmental Key Generation (EKG), as discussed in Section 7.2.4, to incorporate Trusted Computing functionality. EKG is founded upon the notion of constructing keying material from classes of environmental data. A Trusted Computing augmented version of this approach uses a current platform's state, as indicated by one or more PCR register values, as the environmental data from which a key is created. This key can then be used to "seal" an agent's sensitive code, data, itinerary and state information. Prior to deployment, the mobile agent owner makes use of Algorithm 7.1 to produce an agent in which its sensitive code, data or state can only be accessed if certain environmental properties are met:

Algorithm 7.1 Pre-Deployment Environmental Key Generation

```
1: $N$:= TPM_GetRandom
2: $PCRs$:= Indices of PCRs
3: $K_{EKG}$:= $H$("concatenated values of $PCRs$ indicative of a trusted platform state")
4: $MA$:= Mobile Agent || $E_{K_{EKG}}$("sensitive portion of code/data/state information to be protected - specific to the mobile agent")
5: $T$:= $H$(N $\oplus$ "concatenated values of PCR registers indicative of a trusted platform state")
6: $Signature := Sign_{K_{AO_{priv}}}(MA \parallel N \parallel PCRs \parallel T)$
7: $MA := MA \parallel N \parallel PCRs \parallel T \parallel Signature$
```

Notation: N is a random number generated by the TPM. $PCRs$ is a list of PCR indices. K_{EKG} is an environmentally generated key generated by the mobile agent's owner. H is a hash function. $E_{K_{EKG}}$ denotes an encryption operation using key K_{EKG}. MA is the mobile agent, the sensitive portions of which are encrypted with K_{EKG}. T is the test case that will be used to determine if a TMAP is in the desired state (see Algorithm 7.3.4). *Signature* is a signature generated over $MA \parallel N \parallel PCRs \parallel T$ using the mobile agent owner's $K_{AO_{priv}}$ private key.

Upon arrival at a destination TMAP, the TMAP verifies the mobile agent's signature and, if verified, executes the mobile agent. The non-EKG protected portion of the mobile agent executes Algorithm 7.2.

The mobile agent requests the host TMAP to perform a TPM_Quote command. As input to this command, the mobile agent specifies the PCR indices it would like reported, $PCRs$, as well as a nonce, $H(N \parallel$ TPM_GetRandom) as a freshness check. Using its private AIK, the TMAP outputs a signed data structure, *attest*.

Algorithm 7.2 Decrypting an EKG Protected Agent

```
Require: $attest$ := TPM_Quote$_{AIK}$($PCRs$, $H(N \parallel$ TPM_GetRandom$)$)
Require: $Verify(attest)$ == valid
Require: TPM_PCR_Composite := Extract($attest$)
1: while TPM_PCR_Composite has more elements do
2:     $state := state \parallel$ TPM_PCR_Composite.pcrvalue
3: end while
4: if $H(N \oplus state) == T$ then
5:     $D_{H(state)}(M)$
6: else
7:     Migrate to next host
8: end if
```

The mobile agent next verifies the AIK signature of the TMAP on the *attest* data structure and checks for the inclusion of his nonce, N. If these checks pass, the mobile agent extracts the TPM_PCR_Composite data structure (which contains the indices of attested PCRs and their values) from *attest*.

In steps 1–3 of Algorithm 7.2 the mobile agent constructs a representation of the current TMAP's platform state by concatenating the values of all the returned PCR registers together into a single string, *state*. In step 4, if $N \oplus state$ matches the owner's test case T, then the TMAP decrypts ($D_{H(state)}(M)$) the sensitive portion of the mobile agent, else the mobile agent migrates to the next host on the mobile agent's itinerary. This process continues until either the mobile agent reaches the end of its itinerary or the mobile agent achieves its goal.

Analysis: We will now examine how well this proposal prevents the threats identified in Sections 7.2.1 and 7.2.2.

- **Unauthorised Access:** From the host's perspective, the threat of unauthorised access is identical to that in our mutual attestation based approach. From the mobile agent's perspective, the inability of a host to gain unauthorised access is predicated upon the inability of a host platform to "guess" the required platform configuration state necessary for decryption of the mobile agent's protected elements. If a host is capable of doing this, then the host can extract the agent's sensitive code data or state.

- **Denial of Service:** From the host's perspective, the threat of unauthorised access is identical to that in our mutual attestation based approach. From the mobile agent's perspective, the threat of denial of service is much greater than that in either our mutual attestation-based or sealing-based approaches. As there is no (implicit or explicit) attestation of either party, the host upon which the mobile agent is currently executing is free to constrain the mobile agent as it sees fit.

- **Eavesdropping:** From the host's perspective, the threat of unauthorised access is identical to that in our mutual attestation based approach. From the mobile agent's perspective, the risk of eavesdropping is much greater than that in either our mutual attestation-based approach or our sealing-based approaches due to the absence of any attestation checks which would allow the mobile agent to know how trustworthy a destination host is prior to migration.

- **Alteration:** From the host's perspective, the threat of alteration is identical to that in our mutual attestation-based approach. From the mobile agent's perspective, the risk of alteration (of unsigned data, code or state information) is much greater than either our mutual attestation based approach or our sealing based approach due to the absence of any attestation checks.

- **Masquerade:** From the host's perspective, the potential of a mobile agent masquerading as another agent is identical to that in our mutual attestation based approach. From the mobile agent's perspective, the risk of a host masquerading as another host is much greater than that in either our mutual attestation based approach or our sealing based approach due to the absence of any attestation checks.

On the surface this approach may seem to suffer from insurmountable deficiencies when compared to either our mutual attestation based approach or our sealing based approach. For reasons that will be outlined in Section 7.3.6, our EKG-based approach has benefits in certain environments where time-sensitive epidemic updates may be required. For instance, one could imagine a vehicular ad hoc network in which a car manufacturer wishes to rapidly upgrade the firmware of select car models whilst ensuring that the correct update is applied only to a specific subset of cars. In this environment, interactions with either the originating host's TPM or the destination host's TPM may cause protracted delays resulting in cars driving past each other and out of each other's communication rage. In this setting, an EKG-based approach may be highly desirable to allow the rapid exchange of mobile agents without having to incur processing delays resulting from interacting with the car's TPM.

7.3.5 Certified Migration

As we saw in Section 2.4.4.1, a Certified Migratable Key (CMK) is a key that are migrated from one TPM-compliant platform to another provided certain conditions can be satisfied. In particular, prior to a CMK being migrated, the approval of a Migration Authority (MA) or Migration Selection Authority (MSA) must be sought. Additionally, post-migration any attribute designation, key type, authorisation or state requirements specified at the time of CMK generation will be enforced by the TPM of the platform to which the key is migrated.

In our final option, prior to deployment, the mobile agent owner signs any immutable portions of the mobile agent using his private key $K_{AO_{priv}}$ and attaches his public key certificate for $K_{AO_{pub}}$ to his mobile agent. The mobile agent owner next generates a new CMK key pair using the TPM_CMK_CreateKey command. As part of this key generation process, the mobile agent owner specifies both a state constraint for private key usage and the identity of the MA/MSA that must provide approval prior to allowing a key to migrate further.

After the private key is generated, the mobile agent owner requests migration of his newly generated CMK private key to all the hosts on his mobile agent's itinerary. The MA/MSA, upon receiving this request, determines whether each platform on the list is considered trustworthy enough to obtain the mobile agent owner's CMK private key. Based on this determination, the MA/MSA signs a TPM_CMK_AUTH or TPM_RESTRICTEDKEYAUTH (depending on whether the approval authority is an MA or an MSA) and returns an array of these structures to the mobile agent owner. Once approval has been received, the mobile agent owner migrates his CMK private key to each approved TMAP on the agent itinerary. The agent owner next encrypts his mobile agent using his CMK public key in the knowledge that his mobile agent can only be decrypted by a destination TMAP to which the CMK private key has been migrated and only when those platforms are in a particular software state.

This approach provides similar mitigation to both host and mobile agent based unauthorised access, denial of service, eavesdropping, alteration and masquerade threats as our sealing-based approach. However, achieving Certified Migration within a mobile agent framework necessitates the inclusion of an additional infrastructural element. We require the presence of either an MSA or an MA within the network and for each agent owner to enrol one or more CMKs with this MA/MSA. Additionally, this approach prevents variability in the mobile agent's itinerary as the agent owner must ahead of time migrate keys to each platform he wishes his agent to visit prior to initially migrating the agent. This may not always be practical in all deployments and may limit the scalability of such a solution.

7.3.6 Performance

One issue not discussed in our analysis of the preceding schemes is the issue of denial of service resulting from delays arising from interactions with a host's TPM. The potential for multiple attestation requests or, in particular, requests involving the generation or use of asymmetric key pairs arriving in quick succession from remote hosts may result in a denial of service attack against a host. On average, a TPM takes between one and three seconds to perform a signature with a TPM-controlled key and even longer to generate an asymmetric key pair [119, 138]. Therefore, an excessive number of attestation or key generation requests may result in a performance bottleneck at the TPM and causes the host platform to drop requests [138]. Our mutual attestation and sealing approaches would appear to suffer most given these constraints. However, our certified migration approach and our environmental key generation approach fare considerably better. In particular, with both our EKG and CMK approaches, the destination host can schedule the execution of agents based on a fair scheduling algorithm for timed access to the host's TPM.

7.4 Conclusions

Anecdotal evidence suggests that the mere presence of Trusted Computing technology in a mobile agent setting would solve a multitude of security issues currently present in mobile agent systems [51]. Unfortunately, this is not always the case. In this chapter we outlined the security issues faced by both mobile agents and mobile agent hosts, and discussed various ways in which Trusted Computing can be used to enhance the security of mobile agent migration. We examined various ways in which Trusted Computing primitives can be deployed in order that an agent originator can gain assurance that each host platform visited by his agent will behave in an expected manner prior to sensitive agent information being accessed. We highlighted a number of ways in which we can achieve this functionality and noted the pros and cons of each approach.

Whilst outside the scope of this chapter, the introduction of light-weight attestation techniques as proposed in [56], [134] and [138] may go a long way to reducing the performance burden for our attestation and sealing approaches. However, much work still remains in the design of light-weight protocols for Trusted Computing. We believe this to be a fruitful area for future investigation.

Chapter 8

Summary and Conclusions

8.1 Summary

In this thesis we have examined how Trusted Computing technologies, as applied to selected application domains, can be used to impede the distribution, infection and execution of malicious applications. Specifically, we have looked at how fundamental concepts of Trusted Computing (namely, integrity measurement, storage and attestation; protected storage; authenticated boot; and hardware-enforced isolation) can be used to restrict the impact of malicious applications (malware) and reduce the attack surface of a platform.

In isolation, integrity measurement and storage functionality do not provide a means of defending against malware. They do, however, provide the foundation for a number of other services useful in combating the effects of malware. The accurate recording and storage of events within a platform enables a Trusted Platform to later reliably report (attest) information about its current state to interested parties. On request from a challenger, a Trusted Platform can, using a private attestation key, sign integrity metrics reflecting (all or part of) the platform's software environment. The challenger may use this information to determine whether it is safe to trust the platform from which the statement has originated and (all or part of) the software environment running on the platform. For instance, a platform, upon requesting access to a company's intranet, may be required to demonstrate through attestation that it has up-to-date anti-virus software with the latest signature definitions, that its spam filters are operating correctly and that it has installed the latest OS security patches. Failure to report, or reporting unacceptable results, may prevent a platform from accessing a company's intranet thus (potentially) reducing the likelyhood of the requesting platform acting as a vector of infection.

Protected storage functionality, as provided by a Trusted Platform, can effectively act as a barrier to the revelation of sensitive data on an infected platform. Using sealed storage, an end-user can protect their private data (e.g., credit card details) by making revelation of that data contingent on a platform being in a particular state. A user can seal their banking data to a state that requires a particular banking application to be running on their platform, and nothing more. This feature would therefore ensure that malicious software, such as a keystroke logger or trojan horse, could not gain access to security sensitive data, as the presence of such malware would change the platform state.

Finally, hardware-enforced software isolation enables the segregation of security-critical software and data so that they cannot be observed and/or modified in an unauthorised manner by software executing in parallel execution environments. The presence of isolated execution environments can ensure that any infection is contained within the execution environment which the malware has infected. Additionally, complementary hardware extensions, developed as part of, and in conjunction with, Intel's LaGrande and AMD's AMD-V initiatives, support the establishment of trusted channels between input and output devices and programs running within isolated execution environments. In this way, user I/O data can be secured in transit to protect it from malware which may have infiltrated the platform [60].

In this thesis we have shown how these various functions, both in isolation and when working in concert, can be used to protect sensitive user data. In Part I of this thesis, we examined the issue of protecting Internet-based Card Not Present (CNP) transactions with Trusted Computing technology. In doing so, we examined what is currently possible versus what will be possible in the medium-to-long term.

In Chapter 4 we examined what can be achieved with current deployments of Trusted Computing for protecting CNP transaction data. To address the problem of a user unknowingly revealing sensitive data to a phishing site we examined the use of the TPM's protected storage capabilities coupled with the use of physical presence signals. We proposed the use of SSL client-side authentication to establish a mutually authenticated SSL tunnel over which credit card data can be communicated. In our approach, an SSL private key (for which the public key has received certification) is non-migratable/certifiable migratable from the customer's TPM. Consequently, when a user visits a phishing site and is tricked into revealing credit-card data, a phisher will not be able to impersonate the user as the phisher will not have access to the private key used to complete client-side SSL authentication. Unfortunately, such an approach does not prevent malware resident on the TPM-host platform from capturing customer data and using the TPM-protected private key to establish illegitimate sessions and fraudulently impersonate a customer. To combat this threat, we employed Trusted Computing's physical presence functionality. The use of physical presence effectively prevents malware from generating multiple spurious transactions without direct user involvement. When combined with client-side certification, our approach offers the potential to significantly reduce the levels of fraud currently seen in on-line CNP payments. Indeed our approach can be extended to any application where demonstration of sole control of a cryptographic key is important.

In Chapter 5, we described a system that makes use of the full spectrum of Trusted Computing technologies to securely emulate point-of-sale Integrated Circuit Cards compliant with the Europay Mastercard and Visa (EMV) specifications. Emulation of EMV-compliant cards confers "tamper resistant" properties that are normally associated with physical EMV card use at point-of-sale terminals, making it possible to demonstrate card ownership and authentication in a virtual environment.

Our e-EMV architecture incorporates both PKI enrolment processes for customers and merchants, and the secure download of an e-EMV card and a merchant terminal application to their respective platforms. Together, the interoperation of these applications replicate the functionality of a standard EMV payment card transaction. Prior to transaction initiation, a customer launches his newly downloaded e-EMV application in an isolated execution environment, and attests to this fact to the merchant. This provides a guarantee that, at a particular point in time, the customer's e-EMV application is operating as intended and that the customer is a valid cardholder. Likewise, a customer may obtain a similar guarantee from a merchant. The customer may request attestion of a merchant terminal application to ensure that (i) the merchant has been properly enrolled and (ii) that the merchant's terminal application will behave as expected and will not divulge his sensitive card data. With both attestations, any divergence from intended operating state (due to unwanted memory resident applications) will be detected on verification of an attestation, allowing customer and merchant risk management routines to terminate a transaction.

In addition, during transaction initiation, EMV PIN authentication and authorisation is made intrinsic to transaction processing in our system, just as with EMV in PoS transactions. A trusted path between the keyboard and TPM, enabled by chipset extensions, can be used to securely transfer authorisation data in the form of a PIN or a passphrase from the customer's keyboard to the TPM. A correctly entered PIN/passphrase then assures the TPM of the physical presence of the cardholder and "unlocks" a TPM private key. The assurance of physical presence can be transferred to the merchant through the TPM's subsequent use of this key. This combined functionality provides a much stronger form of authentication and authorisation than is currently employed for CNP payments. Our work in this area builds upon the pre-existing EMV infrastructure to provide a secure and extensible architecture for CNP payments.

In Part II of this thesis, we examined Trusted Computing technology in the context of Peer-to-Peer systems and mobile agents. In Chapter 6, we demonstrated the application of Trusted Computing to securing Peer-to-Peer networks. In doing so, we noted that many security services within networks depend on the establishment of stable identities of network users. We showed how the TPM protocols for Direct Anonymous Attestation can be used to enforce the use of stable, platform-dependent pseudonyms and thus reduce pseudospoofing in P2P networks. Taking this a step further, we showed how runs of DAA protocol can be used to build entity authentication at the level of pseudonyms and can be securely linked to the establishment of secure channels with known endpoints. Our work here reflects a possible way forward in using Trusted Computing to enable secure P2P e-commerce, through the development of secured commercial P2P networks and of P2P networks free from pseudospoofing and pseudotheft.

In Chapter 7, we examined the long-term role Trusted Computing can play in mobile agent security. Mobile agents are autonomous programs that are capable of migrating from one host to another in order to perform some specific task or function. This mobility aspect engenders a number of threats to both mobile agents and the hosting systems on which they execute. The more hosts an agent visits, the greater the risk of agent compromise by a malicious host. This in turn leads to a corresponding increase in risk of compromise to subsequent host systems that an agent visits. To address these problems a wide range of proposals have been developed that tackle both host and agent security. We examined a number of alternative mechanisms for sealing mobile agents to pre-determined platform states to protect sensitive agent information, and discussed the relative merits and deficiencies of each approach.

8.2 Directions for Future Work

At the end of each chapter, we provide several avenues for future research. However, in addition to these individual pieces of work, a number of key challenges need to be addressed in order to accelerate the widespread adoption of Trusted Computing. Many of the challenges are not purely technical in nature, but rather involve a mixture of technical, policy and management aspects.

Perhaps the four most difficult problems presented in this thesis will be the design of a semantically meaningful form of attestation that is capable of dealing with the complexities of modern systems, the design of robust business models that take advantage of Trusted Computing functionality, the creation of a global PKI infrastructure and the design of Trusted Computing products such that the technology will be usable for mass-market consumption. Thus far, with the exception of ongoing research into the design of new attestation protocols, all of these concerns have largely been ignored in the literature. We believe these to be serious problems that must be overcome if Trusted Computing is to fulfil its maximum potential.

8.2.1 Attestation

Given current software development practices, frequent patching to OS components and applications can be expected to be the norm for the foreseeable future. The current method for performing integrity measurements on Trusted Computing platforms means that the order in which patches are applied can result in a combinatorial explosion of distinct configurations for a single application, with each configuration requiring a distinct reference value for attestation purposes. This makes the job of the entity evaluating an application exceedingly difficult, as they will need to be able to verify all possible combinations of patches to an application. Indeed, perhaps more problematic is the fact that the application of one or more patches to an application may render any security guarantees associated with the original application void, unless the evaluation process is rerun for every combination of patch. Semantic remote attestation and property based attestation take the first steps in addressing some of these concerns but much work still remains.

8.2.2 Business Models and Risk

We are beginning to see the emergence of Trusted Computing augmented applications in enterprise-level security infrastructures, with the pharmaceutical, military and commercial sectors taking advantage of one or more aspects of TPM functionality [77]. However, at present, consumer deployments of Trusted Computing remain relatively untested. A through investigation of the business risks associated with deploying a consumer-based Trusted Computing infrastructures would be necessary to fully realise the applications discussed in Chapters 4 – 7. Integrating Trusted Computing with the strategic objects of a consumer-facing industry would necessitate a through cost-benefit quantification, specific to each environment into which Trusted Computing functionality would be deployed. We believe this will be an interesting area for future research.

8.2.3 Public Key Infrastructure and Trusted Computing

Trusted Computing relies on an as yet largely unavailable and unspecified PKI in which multiple CAs (possibly existing in different organisational, procedural and/or jurisdictional domains) are expected to inter-operate. The challenges and pitfalls of PKI deployment are well-documented [64, 106], and high-profile system and protocol failures that have been blamed on inappropriate deployment of PKI abound, with SET [133] providing one of the most prominent recent examples. Consequently, the development of a Trusted Computing PKI and the supporting key life-cycle policies and practice statements pose many legal and policy-design challenges. We believe this will be a fruitful area for future research.

8.2.4 Usability

Currently, the use of a TPM requires a detailed understanding of how the underlying technology works. For example, the very act of taking ownership of a TPM prior to its use is a non-trivial task requiring a user to understand and edit BIOS settings. Similarly, problems may arise with respect to password use and management, where a user may be expected to remember large numbers of passwords associated with protected data or keys in a TPM. While the deployment of numerous passwords may be viewed as a sound security decision, management of such passwords so that access is not jeopardised may prove problematic.

Usability issues such as these are currently a reflection of the general immaturity of Trusted Computing technology and the associated marketplace. Whilst a huge effort has been put into the design and specification of technical aspects of Trusted Computing by the TCG, so far less work seems to have been done to address user-centric issues. The design of sufficiently simple, yet semantically powerful, user-interfaces that are capable of communicating rich security information to users remains an open research problem.

8.3 Conclusions

Malware is beginning to combine characteristics of viruses, worms and trojan horses with server and Internet vulnerabilities, fusing numerous methods for compromising end-user systems with multiple means of propagation to other networked machines [142]. Consequently, malware technology represents a serious and viable threat to both consumer and corporate data.

As a consequence of the perpetual increases in the volume, complexity and scope of malware attacks, the natural question arises, how can users trust the software environments with which they interact? Trusted Computing technology partially addresses this question by providing a means for end-users (and third-parties) to derive increased confidence in the platforms with which they interface, as well as providing standardised mechanisms to protect user data and information from software attack.

Trusted Computing is undoubtedly a powerful technology, with a huge range of possible applications. Some of these have been developed in this thesis. Nevertheless, there remain a number of significant obstacles to its widespread use, as we have discussed above. Addressing these challenges is therefore a high priority for future research. Many challenges to the successful large-scale use of Trusted Computing remain. Providing the full benefits of Trusted Computing to the widest possible audience represents a major challenge for future research.

Bibliography

[1] M. Al-Meaither and C. J. Mitchell. Extending EMV to Support Murabaha Transactions. In *Proceedings of the 7th Nordic Workshop on Secure IT Systems (NordSec 2003)*, pages 95–108. Department of Telematics, NTNU, Trondheim, Norway, October 2003.

[2] A. Alsaid and C. J. Mitchell. Preventing Phishing Attacks Using Trusted Computing Technology. In *Proceedings of the 6th International Network Conference (INC 2006)*, pages 221–228, July 2006.

[3] T. Alves and D. Felton. TrustZone: Integrated Hardware and Software Security – Enabling Trusted Computing in Embedded Systems. White paper, ARM, Available On-line, July 2004. `http://www.arm.com/pdfs/TZ_Whitepaper.pdf`.

[4] AMD. AMD64 Architecture Programmer's Manual: Volume 2: System Programming. Technical Report AMD Publication no. 24594 rev. 3.11, Advanced Micro Devices, May 2006. `http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/24593.pdf`.

[5] R. Anderson. Cryptography and Competition Policy: Issues with 'Trusted Computing'. In L. J. Camp and S. Lewis, editors, *Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing (PODC 2003)*, pages 3–10. Kluwer Academic Publishers, July 2003.

[6] R. Anderson. 'Trusted Computing' Frequently Asked Questions - Version 1.1. Available On-line, August 2003. http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html.

[7] Anti-Phishing Working Group. Phishing Activity Trends Report. Available On-line, April 2007. http://www.antiphishing.org/reports/apwg_report_april_2007.pdf.

[8] APACS. Card Fraud – The Facts 2008. Available On-line, April 2007. http://www.apacs.org.uk/resources_publications/documents/FraudtheFacts2008.pdf.

[9] APACS. Card Fraud Losses Continue to Fall. Available On-line, March 2007. http://www.apacs.org.uk/media_centre/press/07_14_03.html.

[10] B. Arbaugh. Improving the TCPA Specification. *IEEE Computer*, 35(8):77–79, August 2002.

[11] F. Armknecht, Y. Gasmi, A.-R. Sadeghi, P. Stewin, M. Unger, G. Ramunno, and D. Vernizzi. An Efficient Implementation of Trusted Channels Based on Openssl. In *Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing (STC 2008)*, pages 41–50. ACM Press, 2008.

[12] Visa International Service Association. 3-D Secure Protocol Specification: System Overview. Available On-line, April 2007. http://partnernetwork.visa.com/pf/3dsec/main.jsp.

[13] T. Aura. RFC 4346 – Cryptographically Generated Addresses (CGA). Available On-line, March 2005.

[14] B. Balacheff, D. Chan, L. Chen, S. Pearson, and G. Proudler. Securing Intelligent Adjuncts Using Trusted Computing Platform Technology. In J. Domingo-Ferrer, D. Chan, and A. Watson, editors, *Proceedings of the 4th Working Con-*

ference on Smart Card Research and Advanced Applications (CARDIS 2001), pages 177–195. Kluwer Academic Publishers, 2001.

[15] 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 Workshops (AINA 2007)*, pages 486–492. IEEE Computer Society, May 2007.

[16] S. Balfe, A. D. Lakhani, and K. G. Paterson. Securing Peer-to-Peer Networks using Trusted Computing. In Mitchell [88], chapter 10, pages 271–298.

[17] S. Balfe, A. D. Lakhani, and K. G. Paterson. Trusted Computing: Providing security for Peer-to-Peer Networks. In G. Caronni, N. Weiler, M. Waldvogel, and N. Shahmehri, editors, *Proceedings 5th International Conference on Peer-to-Peer Computing (P2P 2005)*, pages 117–124. IEEE Computer Society, August 2005.

[18] S. Balfe and K. G. Paterson. Augmenting Internet-based Card Not Present Transactions with Trusted Computing: An Analysis. Technical Report RHUL-MA-2006-9, Department of Mathematics, Royal Holloway, University of London, London, UK, 2006. `http://www.rhul.ac.uk/mathematics/techreports`.

[19] S. Balfe and K. G. Paterson. Augmenting Internet-based Card Not Present Transactions with Trusted Computing: An Analysis. Technical Report RHUL-MA-2006-9-v2, Department of Mathematics, Royal Holloway, University of London, London, UK, 2006. `http://www.rhul.ac.uk/mathematics/techreports`.

[20] S. Balfe and K. G. Paterson. e-EMV: Emulating EMV for Internet Payments using Trusted Computing Technology. Technical Report RHUL-MA-2006-10, Department of Mathematics, Royal Holloway, University of London, London, UK, 2006. `http://www.rhul.ac.uk/mathematics/techreports`.

[21] S. Balfe and K. G. Paterson. Augmenting Internet-based Card Not Present Transactions with Trusted Computing (Extended Abstract). In *Proceedings of the 12th International Conference of Financial Cryptography and Data Security (FC 2008)*, pages 171–175. Springer, January 2008.

[22] S. Balfe and K. G. Paterson. e-EMV: Emulating EMV for Internet Payments with Trusted Computing Technologies. In *Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing (STC 2008)*. ACM Press, October 2008.

[23] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (Im)possibility of Obfuscating Programs. In *Proceedings 21st Annual International Cryptology Conference (Crypto 2001)*, pages 1–18, August 2001.

[24] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauery, I. Pratt, and A. Warfield. XEN and the Art of Virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003)*, pages 164–177. ACM Press, October 2003.

[25] E. Bierman and E. Cloete. Classification of Malicious Host Threats in Mobile Agent Computing. In *Proceedings of the 2002 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement Through Technology (SAICSIT 2002)*, pages 141–148. South African Institute for Computer Scientists and Information Technologists, September 2002.

[26] N. Borselius. Multi-agent System Security for Mobile Communication). Technical Report RHUL-MA-2003-5, Department of Mathematics, Royal Holloway, University of London, London, UK, September 2003. `http://www.ma.rhul.ac.uk/static/techrep/2003/RHUL-MA-2003-5.pdf`.

[27] E. Brickell, J. Camenisch, and L. Chen. Direct Anonymous Attestation. In B. Pfitzmann and P. Liu, editors, *Proceedings of the 11th ACM Conference on*

Computer and Communications Security (CCS 2004), pages 132–145. ACM Press, October 2004.

[28] J. Camenisch. Direct Anonymous Attestation: Achieving Privacy in Remote Authentication. Zurich Information Security Colloquium 2004. IBM Zurich Information Security Center. `http://www.zisc.ethz.ch/events/infseccolloquium2004`.

[29] J. Camenisch and A. Lysyanskaya. A Signature Scheme with Efficient Protocols. In S. Cimato, C. Galdi, and G. Persiano, editors, *Proceedings of the 3rd International Conference on Security in Communication Networks (SCN 2002)*, volume 2576 of *LNCS*, pages 268–289. Springer, September 2003.

[30] D. L. Chaum. Untraceable Electronic Mail, Return addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[31] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval system. In *Proceedings of International Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, pages 46–66. Springer–Verlag, 2001.

[32] S. Crane. Privacy Preserving Trust Agents. Technical Report HPL-2004-197, Hewlett-Packard Laboratories, Bristol, UK, November 2004. `http://www.hpl.hp.com/techreports/2004/HPL-2004-197.pdf`.

[33] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A Reputation-based Approach for Choosing Reliable Resources in Peer-to-PeerNetworks. In V. Atluri, editor, *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, pages 207–216. ACM Press, November 2002.

[34] L. D'Anna, B. Matt, A. Reisse, T. Van Vleck, S. Schwab, and P. LeBlanc. Self-Protecting Mobile Agents Obfuscation Report. Technical Report Report

03-015, Network Associates Laboratories, June 2003. http://www.au.af.mil/au/awc/awcgate/darpa/obfreport.pdf.

[35] N. Daswani, H. Garcia-Molina, and B. Yang. Open Problems in Data-Sharing Peer-to-Peer Systems. In D. Calvanese, M. Lenzerini, and R. Motwani, editors, *Proceedings of 9th International Conference on Database Theory (ICDT 2003)*, volume 2572 of *LNCS*, pages 1–15. Springer–Verlag, January 2003.

[36] N. Daswani, P. Golle, S. Marti, H. Garcia-Molina, and D. Boneh. Evaluating Reputation Systems for Document Authenticity. Technical Report 2003-34, Computer Science Department, Stanford University, June 2003. http://dbpubs.stanford.edu:8090/pub/2003-34.

[37] A. Datta, M. Hauswirth, and K. Aberer. Beyond "Web of Trust": Enabling P2P E-Commerce. In R. Grinter, T. Rodden, P. Aoki, E. Cutrell, R. Jeffries, and G. M. Olsen, editors, *Proceedings of the 2003 IEEE Conference on Electronic Commerce (CEC 2003)*, pages 303–312. IEEE Computer Society, June 2003.

[38] S. Deering and C. Allen. RFC 2460 – Internet Protocol, Version 6 (IPv6) Specification. Available On-line, December 1998. http://www.ietf.org/rfc/rfc2460.txt.

[39] A.W. Dent and C.J. Mitchell. *User's Guide to Cryptography and Standards.* Artech House, Boston, Massachusetts, USA, 2005.

[40] L. Detweiler. The Snakes of Medusa and Cyberspace – Internet Identity Subversion. Available On-line, November 1993. http://www.interesting-people.org/archives/interesting-people/199311/msg00054.html.

[41] R. Dhamija, J. D. Tygar, and M. Hearst. Why Phishing Works. In R. Grinter, T. Rodden, P. Aoki, E. Cutrell, R. Jeffries, and G. M. Olsen, editors, *Proceed-*

ings of the 2006 Conference on Human Factors in Computing Systems (CHI 2006), pages 581–590. ACM Press, April 2006.

[42] T. Dierks and C. Allen. RFC 4346 – The TLS Protocol Version 1.1. Available On-line, April 2006. `http://www.ietf.org/rfc/rfc4346.txt`.

[43] J. R. Douceur. The Sybil Attack. In P. Druschel, M.F. Kaashoek, and A.I.T. Rowstron, editors, *Proceedings of the 1st International Workshop on International Workshop on Peer-to-Peer Systems (IPTPS 2002)*, volume 2429 of *LNCS*, pages 251–256. Springer–Verlag, March 2002.

[44] T. C. Du, E. Y. Li, and E. Wei. Mobile Agents for a Brokering Service in the Electronic Marketplace. *Decision Support Systems*, 39(3):371–383, 2005.

[45] EMVCo. *Book 3 – Application Specification*, 4.0 edition, December 2000. `http://www.emvco.com`.

[46] EMVCo. *Book 1 – Application independent ICC to Terminal Interface requirements*, 4.1 edition, May 2004. `http://www.emvco.com`.

[47] EMVCo. *Book 2 – Security and Key Management*, 4.1 edition, May 2004. `http://www.emvco.com`.

[48] EMVCo. *Book 3 – Application Specification*, 4.1 edition, May 2004. `http://www.emvco.com`.

[49] EMVCo. *Book 4 – Cardholder, Attendant, and Acquirer Interface Requirements*, 4.1 edition, June 2004. `http://www.emvco.com`.

[50] W. M. Farmer, J. D. Guttman, and V. Swarup. Security for Mobile Agents: Authentication and State Appraisal. In *Proceedings of the 4th European Symposium On Research In Computer Security (ESORICS 1996)*, pages 118–130. Springer–Verlag, September 1996.

[51] Unlimited Freedom. Interesting Uses of Trusted Computing, Part 2. Available On-line, March 2004. `http://invisiblog.com/1c801df4aee49232/#mobile`.

[52] Fuzen_op. The FU Rootkit. Available On-line. `http://www.rootkit.com/`.

[53] 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 The 2nd International Conference on Availability, Reliability and Security (ARES 2007)*, pages 120–127. IEEE Computer Society, April 2007.

[54] E. Gallery and A. Tomlinson. Protection of Downloadable Software on SDR Devices. In *Proceedings of the 4th Software Defined Radio Forum Technical Zonference (SDR 2005)*. Software Defined Radio Forum (SDRF), November 2005.

[55] E. Gallery and A. Tomlinson. Secure Delivery of Conditional Access Applications to Mobile Receivers. In Mitchell [88], chapter 7, pages 195–238.

[56] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A Virtual Machine-based Platform for Trusted Computing. *ACM SIGOPS Operating Systems Review*, 37(5):193–206, 2003.

[57] T. Garfinkel, M. Rosenblum, and D. Boneh. Flexible OS Support and Applications for Trusted Computing. In *Proceedings of the 9th USENIX Workshop on Hot Topics on Operating Systems (HotOS-IX)*, pages 145–150. USENIX Association, May 2003.

[58] Y. Gasmi, A.-R. Sadeghi, P. Stewin, M. Unger, and N. Asokan. Beyond Secure Channels. In S. Xu and M. Yung, editors, *Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing (STC 2007)*, pages 30–40. ACM Press, November 2007.

[59] L. Gong, G. Ellison, and M. Dageforde. *Inside Java 2 Platform Security: Architecture, API Design, and Implementation.* Addison-Wesley Longman Publishing, Inc., 2nd edition, 2003.

[60] D. Grawrock. *The Intel Safer Computer Initiative: Building Blocks for Trusted Computing*, chapter Protected Input and Output, pages 143–164. Intel Press, 2006.

[61] F. Griffel, M. T. Tu, M. Münke, M. Merz, W. Lamersdorf, and M. M. da Silva. Electronic Contract Negotiation as an Application Niche for Mobile Agents. In *Proceedings of the 1st International Conference on Enterprise Distributed Object Computing (EDOC 1997)*, pages 354–367. IEEE Computer Society, October 1997.

[62] D. Griffin, G. Pavlou, and P. Georgatsos. Providing Customisable Network Management Services Through Mobile Agents. In *Proceedings of the 7th International Conference on Intelligence and Services in Networks (IS&N 2000)*, pages 209–226. Springer–Verlag, February 2000.

[63] R. Grimes. Authenticode. Available On-line, 2008. `http://technet.microsoft.com/en-us/library/cc750035.aspx`.

[64] P. Gutman. PKI: It's Not Dead, Just Resting. *Computer*, 35(8):41–49, 2002.

[65] V. Haldar, D. Chandra, and M. Franz. Semantic Remote Attestation: A Virtual Machine Directed Approach to Trusted Computing. In *Proceedings of the 3rd Conference on Virtual Machine Research And Technology Symposium (VM 2004).* USENIX Association, May 2004.

[66] E. V. Herreweghen and U. Wille. Risks and Potentials of Using EMV for Internet Payments. In *In Proceedings of the 1st USENIX Workshop on Smartcard Technology*, pages 163–174. USENIX Association, May 1999.

[67] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. Available On-line, January 1999.

[68] Intel. LaGrande Technology Architectural Overview. Technical Report 252491-001, Intel Corporation, September 2003. `http://www.intel.com/technology/security/downloads/LT_Arch_Overview.pdf`.

[69] International Organisation for Standardization. *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture.* ISO/ITU, 1989.

[70] *ITU-T Recommendation X.509, Information technology — Open Systems Interconnection — The Directory: Public-key and Attribute Certificate Frameworks.* International Organization for Standardisation, Geneva, Switzerland, 2000. 4th edition.

[71] S. Iyer, A. Rowstron, and P. Druschel. Squirrel: A Decentralized Peer-to-Peer Web Cache. In *Proceedings of the 21st Annual Symposium on Principles Of Distributed Computing (PODC 2002)*, pages 213–222. ACM Press, July 2002.

[72] C. Jackson, D. Boneh, and J. Mitchell. Spyware Resistant Web Authentication Using Virtual Machines. `http://crypto.stanford.edu/antiphishing/spyblock.pdf`.

[73] C. Jackson, D. Boneh, and J. Mitchell. Transaction Generators: Root Kits for Web. In *Proceedings of 2nd USENIX Workshop on Hot Topics in Security (HotSec 2007)*, pages 1–4. USENIX Association, August 2007.

[74] W. Jansen and T. Karygiannis. Mobile Agents and Security. NIST Special Publication 800-19, National Institute of Standards and Technology (NIST), Computer Security Division, Gaithersburg, MD, USA, 1999. `http://src.nist.gov/publications/nistpubs/800-19/sp800-19.pdf`.

[75] R. Jha and S. Iyer. Performance Evaluation of Mobile Agents for E-commerce Applications. In *Proceedings of the 8th International Conference on High Performance Computing (HiPC 2001)*, pages 331–340. Springer-Verlag, December 2001.

[76] A. Jösang, R. Ismail, and C. Boyd. A Survey of Trust and Reputation Systems for Online Service Provision. *Decision Support Systems*, 43(2):618–644, 2007.

[77] R. L. Kay. Trusted Computing is Real and its Here. Available On-line, January 2007. `https://www.trustedcomputinggroup.org/news/Industry_Data/Endpoint_Technologies_Associates_TCG_report_Jan_29_2007.pdf`.

[78] V. Khu-Smith and C. J. Mitchell. Using EMV Cards to Protect E-commerce Transactions. In K. Bauknecht A. M. Tjoa and G. Quirchmayr, editors, *Proceedings of the 3rd International Conference on E-Commerce and Web Technologies (EC-WEB 2002)*, volume 2455 of *LNCS*, pages 388–399. Springer–Verlag, January 2002.

[79] M. Kinateder and S. Pearson. A Privacy-enhanced Peer-to-Peer Reputation System. In K. Bauknecht, A.M. Tjoa, and G. Quirchmayr, editors, *Proceedings of the 4th International Conference on Electronic Commerce and Web Technologies (EC-Web 2003)*, volume 2738 of *LNCS*, pages 206–216. Springer–Verlag, September 2003.

[80] S. T. King, P. M. Chen, Y-M. Wang, C. Verbowski, H. J. Wang, and J. R. Lorch. SubVirt: Implementing Malware with Virtual Machines. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P 2006)*, pages 314–327. IEEE Computer Society, May 2006.

[81] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. *SIGPLAN Notices*, 35(11):190–201, November 2000.

[82] S. Marti and H. Garcia-Molina. Identity Crisis: Anonymity vs. Reputation in P2P Systems. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P 2003)*, pages 134–141. IEEE Computer Society, September 2003.

[83] J. M. McCune, B. Parno, A. Perrig, M. K. Reiter, and A. Seshadri. Minimal TCB Code Execution. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P 2007)*, pages 267–272. IEEE Computer Society, May 2007.

[84] J. M. McCune, A. Perrig, and M. K. Reiter. Bump in the Ether: A Framework for Securing Sensitive User Input. In *Proceedings of the 2006 USENIX Annual Technical Conference (USENIX 2006)*, pages 185–198. USENIX Assocation, June 2006.

[85] P. Meadowcroft. Combating Card Fraud. Available On-line, January 2005. `http://www.scmagazine.com/uk/news/article/459478/combating+card+fraud/`.

[86] A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*, volume 6 of *Discrete Mathematics and its Applications.* CRC Press, Boca Raton, Florida, USA, 1997.

[87] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-Peer computing. Technical Report HPL-2002-57, Hewlett-Packard Laboratories, March 2002. `http://www.hpl.hp.com/techreports/2002/HPL-2002-57.html`.

[88] C. J. Mitchell, editor. *Trusted Computing.* IEE Professional Applications of Computing Series 6. The Institute of Electrical Engineers (IEE), April 2005.

[89] D. Molnar, R. Dingledine, and M. J. Freedman. Free Haven. In Oram [95], chapter 12.

[90] G. C. Necula. Proof-Carrying Code. In *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles Of Programming Languages (POPL 1997)*, pages 106–119. ACM Press, January 1997.

[91] C. Neuman, S. Hartman, and K. Raeburn. RFC 4120 – The Kerberos Network Authentication Service (V5). Available On-line, July 2005. `http://tools.ietf.org/html/rfc4120`.

[92] NIST. Specifications for the SECURE HASH STANDARD. Technical Report Federal Information Processing Standards Publication 180-2, The National Institute of Standards and Technology (NIST), August 2002. `http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf`.

[93] OMA. DRM architecture v2.0. Technical Specification OMA-DRM-ARCH-V2_0-2004071515-C, The Open Mobile Alliance (OMA), July 2004.

[94] D. O'Mahony, M. Peirce, and H. Tewari. *Electronic Payment Systems for E-Commerce*. Artech House, 2nd edition, 2001.

[95] A. Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, 2001.

[96] J. K. Ousterhout, J. Y. Levy, and B. B. Welch. The Safe-TCL Security Model. Technical Report TR-97-60, Sun Microsystems Laboratories, California, US, March 1997. `http://research.sun.com/techrep/1997/smli_tr-97-60.pdf`.

[97] PCI Security Standards Council. Payment Card Industry Data Security Standard – Version 1.1. Available On-line, September 2006. `https://www.pcisecuritystandards.org/tech/download_the_pci_dss.htm`.

[98] S. Pearson. Trusted Agents that Enhance User Privacy by Self-Profiling. Technical Report HPL-2002-196, Hewlett-Packard Laboratories, Bristol, UK, 15 July 2002. `http://hpl.hp.co.uk/techreports/2002/HPL-2002-196.pdf`.

[99] S. Pearson. Trusted Computing Platforms, the Next Security Solution. Technical Report HPL-2002-221, Hewlett-Packard Laboratories, November 2002. http://www.hpl.hp.com/techreports/2002/HPL-2002-221.pdf.

[100] S. Pearson, editor. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall, 2003.

[101] S. Pearson. How Trusted Computers can Enhance Privacy Preserving Mobile Applications. In *Proceedings of the 1st International IEEE Workshop on Trust, Security and Privacy for Ubiquitous Computing (WOWMOM 2005)*, pages 609–613. IEEE Computer Society, June 2005.

[102] M. Peinado, Y. Chen, P. England, and J. Manferdelli. NGSCB: A Trusted Open System. In H. Wang, J. Pieprzyk, and V. Varadharajan, editors, *Proceedings of 9th Australasian Conference on Information Security and Privacy, (ACISP 2004)*, volume 3108 of *LNCS*, pages 86–97. Springer–Verlag, July 2004.

[103] M. Peinado, P. England, and Y. Chen. An Overview of NGSCB. In Mitchell [88], chapter 7, pages 115–141.

[104] B. Pfitzmann, J. Riordan, C. Stüble, M. Waidner, and A. Weber. The PERSEUS System Architecture. Technical Report RZ 3335 (#93381), IBM Research Division, Zurich Laboratory, April 2001.

[105] D. Piper. RFC 2407 – The Internet IP Security Domain of Interpretation for ISAKMP. Available On-line, November 1998. http://www.ietf.org/rfc/rfc2407.txt.

[106] G. Price. PKI – An Insider's View (Extended Abstract). Technical Report RHUL-MA-2005-8, Department of Mathematics, Royal Holloway, University of London, London, UK, June 2005. http://www.ma.rhul.ac.uk/static/techrep/2005/RHUL-MA-2005-8.pdf.

[107] 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 2006)*, pages 67–74. ACM Press, May 2006.

[108] G. J. Proudler. Concepts of Trusted Computing. In Mitchell [88], chapter 2, pages 11–27.

[109] D. Qiu and R. Srikant. Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks. In *Proceedings of the 2004 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2004)*, pages 367–378. ACM Press, August 2004.

[110] C. Radu. *Implementing Electronic Card Payment Systems.* Artech House, November 2002.

[111] J. Reid, J. M. Gonzalez Nieto, and E. Dawson. Privacy and Trusted Computing. In *Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA 2003)*, pages 383–388. IEEE Computer Society, September 2003.

[112] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):66–92, 1998.

[113] M. Rennhard and B. Plattner. Practical Anonymity for the Masses with Mix-Networks. In *Proceedings of the 12th International Workshop on Enabling Technologies (WETICE 2003)*, pages 255–262. IEEE Computer Society, June 2003.

[114] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation Systems. In *Communications of ACM*, volume 43, pages 45–48. ACM Press, December 2000.

[115] J. Riordan and B. Schneier. Environmental Key Generation Towards Clueless Agents. In G. Vigna, editor, *Mobile Agents and Security*, volume 1419 of *LNCS*, pages 15–24. Springer–Verlag, 1998.

[116] V. Roth. Secure Recording of Itineraries through Co-operating Agents. In *Proceedings of the 12th European Conference on Object-Oriented Programming (ECOOP 1998)*, pages 297–298. Springer–Verlag, July 1998.

[117] K. Rothermel and M. Schwehm. Mobile Agents. In A. Kent and J.G. Williams, editors, *Encyclopedia for Computer Science and Technology*, volume 40, pages 155–176. M. Dekker Inc., 1999.

[118] S. Schoen, Electronic Frontier Foundation. Trusted Computing: Promise and Risk. Available On-line, October 2003. `http://www.eff.org/Infrastructure/trusted\_computing/20031001_tc.pdf`.

[119] A.-R. Sadeghi, M. Selhorst, C. Stüble, C. Wachsmann, and M. Winandy. TCG Inside?: A Note on TPM Specification Compliance. In *Proceedings of the 1st ACM Workshop on Scalable Trusted Computing (STC 2006)*, pages 47–56. ACM Press, November 2006.

[120] A.-R. Sadeghi and C. Stüble. Property-based Attestation for Computing Platforms: Caring About Properties, Not Mechanisms. In C.F. Hempelmann and V. Raskin, editors, *Proceedings of the 2004 Workshop on New Security Paradigms (NSPW 2004)*, pages 67–77. ACM Press, 2004.

[121] A.-R. Sadeghi, C. Stüble, and N. Pohlmann. European Multilateral Secure Computing Base: Open Trusted Computing for You and Me. Available On-line, 2004. `http://www.prosec.rub.de/Publications/SaStPo2004Web.pdf`.

[122] R. Sandhu and X. Zhang. Peer-to-Peer Access Control Architecture Using Trusted Computing Technology. In *Proceedings of the 10th ACM Symposium*

on Access Control Models And Technologies (SACMAT 2005), pages 147–158. ACM Press, June 2005.

[123] L. F. G. Sarmenta, M. van Dijk, C. W. O'Donnell, J. Rhodes, and S. Devadas. Virtual Monotonic Counters and Count-Limited Objects Using a TPM Without a Trusted OS. In *Proceedings of the 1st ACM Workshop on Scalable Trusted Computing (STC 2006)*, pages 47–56. ACM Press, November 2006.

[124] S. E. Schechter, R. A. Greenstadt, and M. D. Smith. Trusted Computing, Peer-To-Peer Distribution and the Economics of Pirated Entertainment. In *Proceedings of 2nd Workshop on Economics and Information Security*. May 2003.

[125] J. Schiller. RFC 4307 – Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2). Available On-line, December 2005. `http://www.rfc-editor.org/rfc/rfc4307.txt`.

[126] S. Schoen. Comments on LT Policy on Owner/User Choice and Control 0.8. Available On-line, December 2003. `http://www.eff.org/Infrastructure/trusted_computing/eff_comments_lt_policy.pdf`.

[127] S. Schoen. Give TCPA an Owner Override. Available On-line, December 2003. `http://www.linuxjournal.com/article/7055`.

[128] S. Schoen. Comments on TCG Design, Implementation and Usage Principles 0.95. Available On-line, October 2004. `http://www.eff.org/Infrastructure/trusted_computing/20041004\_eff\_comments\_tcg_principles.pdf`.

[129] S. Schoen. Compatibility, Competition, and Control in Trusted Computing Environments. *Information Security Technical Report*, 10(2):105–119, 2005.

[130] U.S. Securities and Exchange Commission. Form 10-K – The TJX Companies, INC. Available On-line, January 2007. `http://www.sec.gov/Archives/edgar/data/109198/000095013507001906/b64407tje10vk.htm`.

[131] IBM Global Services. IBM Global Business Security Index Report, February 2005. `http://www-935.ibm.com/services/us/index.wss/offering/bcrs/a1008776`.

[132] A. Seshadri, M. Luk, N. Qu, and A. Perrig. SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes. In T. Bressoud and F. Kaashoek, editors, *Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP 2007)*, pages 335–350. ACM Press, October 2007.

[133] SETCo. SET Secure Electronic Transaction 1.0 Specification – The Formal Protocol Definition. Available On-line, May 1997. `http://www.cl.cam.ac.uk/research/security/resources/SET/`.

[134] 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 (S&P 2005)*, pages 154–168. IEEE Computer Society, May 2005.

[135] A. Spalka, A. B. Cremers, and H. Langweg. Protecting the Creation of Digital Signatures with Trusted Computing Platform Technology against Attacks by Trojan Horse Programs. In M. Dupuy and P. Paradinas, editors, *Proceedings of the 16th Annual Working Conference on Information Security (IFIP/Sec 2001)*, volume 193 of *IFIP Conference Proceedings*, pages 403–419. Kluwer Academic Publishers, 11–13 June 2001.

[136] C. Spyrou, G. Samaras, E. Pitoura, and P. Evripidou. Mobile Agents for Wireless Computing: The Convergence of Wireless Computational Models

with Mobile-agent Technologies. *Mobile Networks and Applications*, 9(5):517–528, October 2004.

[137] R. Stallman. *Free Software, Free Society: Selected Essays of Richard M. Stallman*, chapter 17, pages 115–119. GNU Press, 2002.

[138] F. Stumpf, A. Fuchs, S. Katzenbeisser, and C. Eckert. Improving the Scalability of Platform Attestation. In *Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing (STC 2008)*. ACM Press, October 2008.

[139] F. Stumpf, O. Tafreschi, P. Röder, and C. Eckert. A Robust Integrity Reporting Protocol for Remote Attestation. In *Proceedings of the 2nd Workshop on Advances in Trusted Computing (WATC 2006)*, November 2006.

[140] Sun Microsystems. The Java Tutorials: Signing and Verifying JAR Files. Available On-line, 2008. `http://java.sun.com/docs/books/tutorial/deployment/jar/signindex.html`.

[141] Symantec. Infostealer.Bankash.G. Available On-line, Febuary 2006. `http://www.symantec.com/security_response/writeup.jsp?docid=2006-010317-5218-99`.

[142] Symantec. Symantec Internet Security Threat Report Volume XI. Available On-line, March 2007. `http://www.symantec.com/enterprise/theme.jsp?themeid=threatreport`.

[143] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous Connections and Onion Routing. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy (S&P 1997)*, page 44. IEEE Computer Society, May 1997.

[144] TCG. Interoperability Specification for Backup and Migration Services. TCG specification Version 1.0, The Trusted Computing Group (TCG), May 2005.

[145] TCG. Subject Key Attestation Evidence Extension. TCG specification version 1.0 revision 7, The Trusted Computing Group (TCG), June 2005.

[146] TCG. TCG Generic Server Specification. TCG specification Version 1.0, The Trusted Computing Group (TCG), July 2005.

[147] TCG. TCG Infrastructure Working Group Reference Architecture for Interoperability (Part I). TCG specification version 1.0 revision 1, The Trusted Computing Group (TCG), June 2005.

[148] TCG. TCG PC Client Specific Implementation Specification For conventional BIOS. TCG specification Version 1.2 Final, The Trusted Computing Group (TCG), July 2005.

[149] TCG. TCG Software Stack (TSS) Specification. TCG Specification Version 1.2 Level 1, The Trusted Computing Group (TCG), January 2006.

[150] TCG. TPM Main, Part 2: TPM Data Structures. TCG Specification Version 1.2 Revision 103, The Trusted Computing Group (TCG), July 2006.

[151] TCG. TCG Specification Architecture Overview. TCG specification Version 1.4, The Trusted Computing Group (TCG), August 2007.

[152] TCG. TNC Architecture for Interoperability. TCG Specification Version 1.2 Revision 4, The Trusted Computing Group (TCG), September 2007.

[153] TCG. TPM Main, Part 1: Design Principles. TCG Specification Version 1.2 Revision 103, The Trusted Computing Group (TCG), July 2007.

[154] TCG. TPM Main, Part 3: Commands. TCG Specification Version 1.2 Revision 103, The Trusted Computing Group (TCG), July 2007.

[155] TCG MPWG. TCG Mobile Trusted Module Specification. TCG Specification Version 1.0 Revision 1, The Trusted Computing Group (TCG), September 2007.

[156] The Sunday Times. Don't Use Cards At Petrol Stations. Available On-line, Febuary 18 2007. `http://business.timesonline.co.uk/`.

[157] US Department of Homeland Security, SRI International Identity Theft Technology Council and the Anti-Phishing Working Group. The Crimeware Landscape: Malware, Phishing, Identity Theft and Beyond. Available On-line, October 2006. `http://www.antiphishing.org/reports/APWG_CrimewareReport.pdf`.

[158] G. Vigna. Cryptographic Traces for Mobile Agents. In G. Vigna, editor, *Mobile Agents and Security*, volume 1419 of *LNCS*, pages 137–153. Springer–Verlag, 1998.

[159] Visa. Cardholder Information Security – Program Bulletin 102307 – Visa Announces New Payment Application Security Mandates. Available On-line, October 2007. `http://usa.visa.com/merchants/risk_management/cisp_payment_applications.html`.

[160] Visa. Cardholder Information Security Program – List of Validated Payment Applications. Available On-line, October 2007. `http://usa.visa.com/merchants/risk_management/cisp_payment_applications.html`.

[161] VMWare. VMWare Server: Free Virtualization for Windows and Linux Servers. Available On-line. `http://www.vmware.com/pdf/server_datasheet.pdf`.

[162] F. von Lohmann. Meditations on Trusted Computing. Available On-line, 2003. `http://www.eff.org/Infrastructure/trusted_computing/20031001_meditations.php`.

[163] R. Wahbe, S. Lucco, T. E. Anderson, and S. L. Graham. Efficient Software-based Fault Isolation. In *Proceedings of the 14th ACM symposium on Operating systems principles (SOSP 1993)*, pages 203–216, December 1993.

[164] D. S. Wallach. A Survey of Peer-to-Peer Security Issues. In M. Okada, B. C. Pierce, A. Scedrov, H. Tokuda, and A. Yonezawa, editors, *Software Security*

– *Theories and Systems, International Symposium, (ISSS 2002)*, volume 2609 of *LNCS*, pages 42–57. Springer–Verlag, November 2002.

[165] T. Weigold, T. Kramp, R. Hermann, F. Horing, P. Buhler, and M. Baentsch. The Zurich Trusted Information Channel — An Efficient Defence Against Man-in-the-Middle and Malicious Software Attacks. In *Proceedings of TRUST 2008*, volume 4968 of *LNCS*, pages 75–91. Springer–Verlag, 2008.

[166] U.G. Wilhelm, S. Staamann, and L. Butty. Introducing Trusted Third Parties to the Mobile Agent Paradigm. In J. Vitek and C. Jensen, editors, *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, volume 1603 of *LNCS*, pages 469–489. Springer–Verlag, 1999.

[167] B. S. Yee. A Sanctuary for Mobile Agents. In J. Vitek and C. D. Jensen, editors, *Secure Internet programming: Security Issues for Mobile and Distributed Objects*, pages 261–273. Springer–Verlag, 1999.

[168] M. Yung. Trusted Computing Platforms: The Good, the Bad, and the Ugly. In R.N. Wright, editor, *Proceedings of the 7th International Conference of Financial Cryptography (FC 2003)*, volume 2742 of *LNCS*, pages 250–254. Springer–Verlag,Springer, January 2003.

[169] K. Zetter. CardSystems' Data Left Unsecured. Available On-line, July 2005. `http://www.wired.com/news/technology/0,1282,67980,00.html`.

[170] P. Zimmermann. *PGP Source Code and Internals.* MIT Press, Cambridge, MA, USA, 1995.