Trusted Computing: Security and Applications

Eimear Gallery and Chris J. Mitchell

Information Security Group, Royal Holloway, University of London

Egham, Surrey TW20 0EX, UK

1st May 2008

Abstract

The main objective of this paper is to highlight some of the major security and application issues confronting trusted computing technology. This technology, now present in a large proportion of new PCs and incorporating a wide range of cryptographic functionality, has the potential to have a major practical impact, but has not been widely discussed. This paper is an attempt to encourage greater debate about this technology and its possible implications. Following a brief introduction to the history of trusted computing, we provide a summary of its main features. This leads naturally to a consideration of the issues which may impede its use, including potential problems with its cryptographic components. Possible applications of the technology are then discussed.

Keywords: trusted computing, computer security, trust

1 Introduction

Trusted computing (TC) is a security technology that has become ubiquitous almost by stealth. A significant proportion of all new PCs now incorporate special-purpose hardware implementing trusted computing functionality. This hardware provides every PC with a secure environment capable of storing secret information, generating cryptographic keys, and implementing cryptographic functions such as encryption and digital signatures. This alone means that trusted computing has the potential to significantly enhance PC security. Even more importantly, trusted computing also incorporates features which enable the state of a PC to be measured and recorded, and for data to be protected so that it is only available if the PC is in a trusted state.

In this paper we provide a brief introduction to trusted computing technology. We outline the main functions it provides, and consider how it achieves its goals. We then go on to review possible security issues with trusted computing. Unsurprisingly, given its high ambitions and considerable complexity, trusted computing technology has already attracted a number of criticisms. Finally, we review some of the ways in which the technology can be used.

The paper has the following main parts. Following this introductory section, in section 2 we provide a brief history of the development of trusted computing. This is followed, in section 3, by a review of the main components of trusted computing technology. Section 4 considers the core trusted computing functionality in greater detail. Security issues arising with trusted computing are considered in section 5, and possible applications of the technology are reviewed in section 6. The paper concludes in section 7 with a brief discussion of the possible future of the technology.

2 Trusted Computing: A Brief History

The term trusted computing, as used here, is surprisingly recent, although some of the key ideas have been around for much longer. Two of the earliest papers to explore trusted computing were published as recently as 2000 [6, 20]. As discussed by Pearson [60], the Trusted Computing Platform Alliance (TCPA), an industry alliance created to develop and standardise Trusted Platform technology, was formed in October 1999. The TCPA released the first specifications in early 2001, defining a fundamental component of a trusted platform, namely a *Trusted Platform Module* (TPM). A TPM is typically implemented as a chip mounted on a PC motherboard, and provides a foundation for all trusted functionality in the PC (in combination with the BIOS). The TCPA specifications are described in some detail in a book published in 2002 [59] (and summarised in [60]).

The work of the TCPA was inherited by its successor body, the Trusted Computing Group (TCG), which is continuing to develop these specifications. An analysis of privacy issues relating to trusted computing, as defined by the TCG, has been provided by Reid, Gonzaléz Nieto and Dawson [67].

The TCPA and TCG have not been the only source of developments relating to trusted computing. Probably the best known of these other developments is the set of proposals from Microsoft, initially under the name Palladium, and subsequently under the title Next Generation Secure Computing Base (NGSCB). NGSCB is a secure operating system architecture, which requires the underlying platform to have certain trusted features. These include the features that are provided by a TCG TPM, together with certain other processor and chipset enhancements. The architecture of NGSCB would appear to have changed significantly since the first announcements of Palladium. Useful background on the evolution of NGSCB is provided in [31, 32, 62].

The Terra system architecture [37], the Perseus framework [64, 71], the Open Trusted Computing architecture [44], and the European Multilaterally Secure Computing Base (EMSCB) [72], all have some similarities to the most recently published version of NGSCB. At the heart of each architecture is an isolation layer that has been designed to support the compartmentalised execution of software.

Terra is based on the notion of a Trusted Virtual Machine Monitor (TVMM) that partitions a computing platform into multiple, isolated, virtual machines. The Perseus framework and the Open Trusted Computing architecture use either a virtual machine monitor such as Xen [14] or a microkernel to provide isolated execution environments. EMSCB incorporates an L4 microkernel-based isolation layer. Each architecture also assumes the use of a trust-enhanced hardware platform which includes a TPM. The presence of chipset and processor enhancements are also pivotal to all architectures in order to enable an implementation of a high-assurance isolation layer.

Hardware manufacturers such as Intel and AMD have specified the required processor enhancements and chipset extensions under the names of LaGrande [39] and Presidio respectively.

While the execute-only memory (XOM) architecture, and the architecture for tamper evident and tamper resistant processing (AEGIS), are not strictly examples of trusted computing platforms, they also provide strong process isolation through the development of hardened processors. The XOM architecture of Lie et al. [46, 47] attempts to fulfil two fundamental objectives: preventing unauthorised execution of software and preventing software consumers from examining protected executable code. This is achieved through the provision of on-chip protection of caches and registers, protection of cache and register values during context switching and on interrupts, and confidentiality and integrity protection of application data when transferred to external memory.

The architecture for a single chip AEGIS processor bears a strong resemblance to the XOM architecture described above. "AEGIS provides users with tamper evident authenticated environments in which any physical or software tampering by an adversary is guaranteed to be detected, and private and authenticated tamper resistant environments, where, additionally, the adversary is unable to obtain any information about software and data by tampering with, or otherwise observing, system operation" [79].

The concept of a secure boot process, another key issue related to the notion of trusted computing, has been widely discussed in the literature, most notably by Tygar and Yee [90], Clark [22], Arbaugh, Farber and Smith [5] and Itoi et al. [42].

Finally we note that trusted computing has been the subject of a considerable amount of criticism. Suggestions have been made that trusted computing is both a potential threat to user privacy and a threat to the ability of the owner of a PC to use it however he or she wishes. It is outside the scope of this paper to describe all the issues raised; we simply note that some of the most outspoken criticism is due to Anderson [3] and Arbaugh [4].

3 Trusted Computing Components

Trusted computing, as discussed here, relates directly to the types of system proposed by the TCG. That is, for our purposes a trusted system is one that can be relied on to behave in a particular manner for a specific purpose. Since its release, trusted computing has become synonymous with three fundamental concepts:

- an authenticated boot process, which enables a platform's state to be reliably measured and recorded;
- platform attestation, which allows a platform's state to be reliably reported; and
- protected storage functionality, which enables data to be stored on a trusted platform so that it is both confidentiality and integrity protected, and so that the owner of the data can control the software state

that the platform must possess in order for that data to be accessible.

However, more recently, the definition of what constitutes trusted computing functionality has been revised and extended to incorporate the concepts of:

- a secure boot process, which enables a platform's state to be reliably measured, verified and recorded; and
- software isolation, which supports the unhindered execution of software safe from interference by other software running on the same platform.

In the remainder of this section we describe the basic components which must be integrated into a trusted platform in order to support these five fundamental functions, i.e. authenticated boot, platform attestation, protected storage, secure boot and software isolation. In section 4 we then examine the functionality offered by a trusted platform in greater detail.

3.1 The Root of Trust for Measurement

An integrity measurement is defined by Peinado, England and Chen [63] as the cryptographic digest or hash of a platform component (i.e. 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 defined as "a condensed value of integrity measurements" [59].

The Root of Trust for Measurement (RTM) is a computing engine capable of measuring at least one platform component, and hence providing an integrity measurement. The RTM is typically implemented as the normal platform processor controlled by a particular instruction set (the so-called 'Core Root of Trust for Measurement' (CRTM)). On a PC, the CRTM may be contained within the BIOS or the BIOS Boot Block (BBB), and is executed by the platform when it is acting as the RTM. It is required by the TCG that the CRTM is protected against software attack; the CRTM must be immutable, as defined by the TCG, meaning that its replacement or modification must be under the control of the host platform manufacturer alone [81]. It is also preferable that the CRTM be physically tamper-evident [59].

3.2 The Root of Trust for Storage

The Root of Trust for Storage (RTS) is a collection of capabilities which must be trusted if storage of data in a platform is to be trusted [59]. The RTS is capable of maintaining an accurate summary of integrity measurements made by the RTM, i.e. condensing integrity measurements and storing the resulting integrity metrics. The RTS also provides integrity and confidentiality protection to data.

3.3 The Root of Trust for Reporting

In conjunction with the RTM and RTS, an additional root of trust is necessary for the implementation of platform attestation, namely the Root of

Trust for Reporting (RTR). The RTR is a collection of capabilities that must be trusted if reports of integrity metrics are to be trusted (platform attestation) [59].

The RTS and the RTR constitute the minimum functionality that should be provided by a TPM [85, 86, 87]. A TPM is generally implemented as a chip which must be physically bound to a platform. In order to support RTS and RTR functionality, a TPM incorporates a number of functional components, including:

- a number of special purpose registers for recording platform state, known as Platform Configuration Registers (PCRs);
- a means of reporting this state to remote entities;
- secure volatile and non-volatile memory;
- random number generation;
- a SHA-1 hashing engine; and
- asymmetric key generation, encryption and digital signature capabilities.

The TPM must be completely protected against software attack, i.e. the RTS and RTR (in the TPM) must be immutable, which implies that the replacement or modification of RTS and RTR code must be under the control of the TPM manufacturer alone. The TPM is required to provide a limited degree of protection against physical attack (i.e. tamper-evidence) [59].

3.4 Root of Trust for Verification

The TCG main specifications do not define the components necessary to enable a secure boot process. However, the recently released specifications for a TCG mobile trusted module [84] describe a fourth root of trust which must be incorporated into a platform if it is to offer this functionality, namely a Root of Trust for Verification (RTV). The RTV is defined as a computing engine capable of verifying at least one platform component's integrity measurement against its expected value (known as a Reference Integrity Measurement (RIM)).

3.5 Isolation Technology

A number of approaches have been proposed to provide software isolation, the most prominent of which we now describe. An operating system (OS)-hosted Virtual Machine Monitor (VMM), such as VMWare workstation, can be used to enable software isolation. In this case, all guest OSs executing in virtual machines use the host OS device drivers. While this implies that every guest can use drivers developed for the host machine, it also means that the isolation layer essentially incorporates the VMM and the host OS, making assurance problematic [2, 63].

In a standalone virtual machine monitor, such as Terra [37], all devices are virtualised or emulated by the VMM. This means that the VMM must contain a virtual device driver for every supported device. As the set of devices used in consumer systems is often large, and as many virtual device drivers are complex, the size of the VMM quickly grows at the cost of assurance. A standalone VMM exposes the original hardware interface to its guests. While this implies that legacy OSs can be supported, it also means that the VMM size is increased because of the complexity involved in virtualising the x86 CPU instruction set [63].

Isolation layers using para-virtualisation techniques, such as Xen [14], have been designed for efficiency, and try to alleviate the complexity introduced when devices are virtualised. Two common approaches used in order to para-virtualise I/O are as follows [2]. In the first case, an I/O-type-specific API for each device is integrated into the VMM, in conjunction with the device drivers [2]. This approach requires a guest OS to incorporate para-virtualised drivers which enable communication with the VMM APIs rather than the hardware device interfaces. While this gives performance gains over full virtualisation, the guest OS must be modified to communicate with the I/O-type-specific APIs. Alternatively, a service OS, which incorporates the VMM APIs and the device drivers, executes in parallel to guest OSs, which are modified to incorporate para-virtualised drivers [2]. To enable this approach, devices are exported to the service OS. While this approach means that device drivers do not have to be implemented within the isolation layer, the isolation layer may become open to attack from a guest in control of a direct memory access device which is, by default, given unrestricted access to the full physical address space of the machine.

The NGSCB isolation layer [62, 63] follows the second approach above, and was designed to take advantage of CPU and chipset extensions incorporated in a new generation of processor hardware; such hardware is being provided, for example, by Intel's LaGrande initiative [39]. The isolation kernel has been designed to execute in a CPU mode more privileged than the existing ring 0, effectively in ring -1, which is being introduced in new versions of the x86 processors. This enables the isolation layer to operate in ring -1 and all guest OSs to execute in ring 0. Thus, complexity problems which arise when virtualising the x86 instruction set are avoided [63]. The original hardware interface is exposed to one guest OS [63]. However, rather than necessitating the virtualisation of all devices, as a VMM does, devices are exported to guest OSs which contain drivers for the devices they choose to support. Guest operating systems may then efficiently operate directly on the chosen device.

This does, however, leave the problem of uncontrolled DMA devices, which by default have access to all physical memory. In order to prevent DMA devices circumventing virtual memory-based protections provided by the isolation layer, it is necessary for the chipset manufacturers to provide certain instruction set extensions. These enable a DMA policy to be set by the isolation layer, which specifies, given the state of the system, if a particular subject (DMA device) has access (read or write) to a specified resource (physical address), [63]. The DMA policy is then read and enforced by hardware, for example the memory controller or bus bridges.

Hardware extensions required in order to support the secure implementation of the NGSCB isolation layer have been provided as part of Intel's LaGrande [39] and AMD's Presidio initiatives. Both enable the efficient and secure implementation of the NGSCB isolation layer through CPU and chipset extensions. Both also support the establishment of trusted channels between the input and output devices and programs running within an isolated environment.

4 The Trusted Platform Subsystem Functionality

In this section we explore the core mechanisms synonymous with trusted computing in further detail.

4.1 The Authenticated Boot Process

An authenticated boot process enables the state of a platform to be reliably measured and recorded. In order to describe an authenticated boot process we first need to introduce some fundamental TPM concepts. A PCR is a 20-byte integrity-protected register present in a TPM; a TPM must contain a minimum of 16 such registers. When a component is 'measured', a 20-byte SHA-1 hash of the component is computed. The output hash value (i.e. the measurement of the component) is then stored in one of the TPM PCRs. In order to ensure that an unlimited number of measurements can be stored in the limited number of PCRs in a TPM, multiple measurements can be stored in a single PCR. This is achieved by concatenating a new measurement with the existing contents of a PCR, hashing the resulting string, and then storing the output hash code in the PCR.

A record of all measured components is stored in the *Stored Measurement Log (SML)*, which is maintained externally to the TPM. The information in the SML is necessary to interpret the PCR values, but does not need to be integrity protected.

A simplified authenticated boot process in a PC might proceed as follows, where we assume that the CRTM is part of the BIOS Boot Block (BBB). The CRTM first measures itself and the rest of the BIOS (i.e. the POST BIOS). The computed measurements are then passed to the RTS, which condenses them and records the resulting integrity metric in the first of the 16 PCRs (PCR-0) within the TPM. Control is then passed to the POST BIOS, which measures the host platform configuration, the option ROM code and configuration, and the OS loader. The computed measurements are passed to the RTS, which condenses them and stores the resulting integrity metrics in PCRs 1-5. Control is then passed to the OS loader which measures the OS. At each stage a record of all measurements computed is stored in the SML.

This process of measuring, condensing, storing, and handing off, continues until, at the end of the boot process, the platform's state has been measured and stored. The exact measurement process is dependent on the platform; for example, the TCG specifications detail authenticated boot processes for a platform which has a 32-bit PC architecture BIOS [81], and for an Extensible Firmware Interface (EFI) platform [83].

4.2 TPM Protected Storage

The TPM provides secure ('protected') storage functionality, designed to enable an unbounded number of secrets/data to be confidentiality and integrity protected on a TP.

Each TPM contains a *Storage Root Key* (SRK), a 2048-bit key pair for an asymmetric encryption scheme. The private key from this key pair is permanently stored inside the TPM. This key pair is the root of the *TPM protected object hierarchy.* A TPM protected object in this hierarchy may be classified as either a *TPM protected key object*, i.e. an asymmetric key pair whose private key is encrypted using a key at a higher layer in the hierarchy, or a *TPM protected data object*, i.e. data or a secret key for a symmetric algorithm, which has been encrypted using a key at a higher layer of the hierarchy.

Asymmetric encryption is used to confidentiality-protect key and data objects. Encrypted storage also provides implicit integrity protection of TPM-protected objects. Data can be associated with a string of 20 bytes of authorisation data before it is encrypted. When data decryption is requested, the authorisation data must be submitted to the TPM. The submitted authorisation data is then compared to the authorisation data in the decrypted string, and the decrypted data object is only released if the values match. If the encrypted object has been tampered with, the authorisation data will most likely have been corrupted (because of the method of encryption employed) and access will not be granted even to an entity which has submitted the correct authorisation data. However, functionality to control how data is used on its release, or to protect data from deletion, is not provided.

The TPM protected storage functionality incorporates a key generation capability. This capability enables the generation of key pairs whose private keys can only be used on the TPM on which they were generated. An additional constraint may also be applied which prevents private key use unless the TPM host platform is in a specified state. Moreover, key pairs can be generated with the property that the private keys are never exported from the TPM in unencrypted form.

The TPM enables the encryption of keys or data outside the TPM in such a way that they can only be decrypted on a particular TPM. It also enables the encryption of keys or data so that they can only be decrypted when a particular TPM host platform is in a specified state.

Finally, sealing functionality is provided, i.e. the encryption by the TPM of data so that it can only be decrypted on that particular TPM, and only when the host platform is in a specified state. The data to be sealed is associated with two sets of integrity metrics, one which represents the state of the platform when the data was sealed (*digest at creation*), and one which represents the state of the platform required for the data to be unsealed (*digest at release*). The sealed data will only be released by the TPM if the host platform is in the state specified in the *digest at release*. Once the data has been released, the *digest at creation* can be checked in order to ensure that the data was not sealed by rogue software.

4.3 Platform Attestation

Platform attestation enables a TPM to reliably report information about its identity and the current state of the TPM host platform. This is achieved using asymmetric cryptography, as we describe below. The procedure also uses a set of key pairs and associated credentials (certificates); this somewhat complex process is necessary in order to allow TP anonymity. We describe the key pairs and the credentials before describing the attestation process itself.

4.3.1 Platform Keys and Credentials

Each TPM is associated with a unique asymmetric encryption key pair called an *endorsement key pair*, which is generated at the time of manufacture. The TP incorporating the TPM is further equipped with a set of *credentials*, i.e. data structures (certificates) signed by a variety of third parties. It is to be expected that these credentials will all be in place at the time the platform is provided to an end user.

We next briefly enumerate the three key types of credential.

- A entity known as the *trusted platform management entity* (which is likely to be the TPM manufacturer) attests to the fact that the TPM is genuine by digitally signing an *endorsement credential.* This certificate binds the public endorsement key to a TPM description.
- *Conformance credentials* are certificates that attest that a particular

part of a trusted platform, e.g. a type of TPM, an associated component such as a CRTM, the connection of a CRTM to a motherboard, and/or the connection of a TPM to a motherboard, conform to the TCG specifications. Such a certificate might be signed by a third party testing laboratory.

- *A platform entity* (typically the platform manufacturer) offers assurance in the form of a *platform credential* that a particular platform is an instantiation of a TP. In order to create a platform credential, a platform entity must examine the endorsement credential of the TPM, the conformance credentials relevant to the TP, and the platform to be certified.

Since a TPM can be uniquely identified by the public key from its endorsement key pair, this key pair is not routinely used by a platform, helping to ensure that the activities of a TP cannot be tracked. Instead, an arbitrary number of pseudonyms in the form of *Attestation Identity Key* (AIK) key pairs can be generated by a TPM and associated with its host TP. This can be achieved using a special type of third party known as a *Privacy-Certification Authority* (P-CA). A P-CA associates AIK public keys with TPs by signing certificates known as *AIK credentials*.

When a platform requests an AIK credential from a P-CA, it must supply the three types of TP credential listed above, as issued at the time of manufacture. The P-CA verifies the TP credentials, thereby obtaining assurance that the TP is genuine, and then creates (signs) an AIK credential binding the AIK public key to a generic description of the TP; note that this generic description should capture enough information for a verifier of the credential to have assurance in the trustworthiness of the platform, but not enough information to uniquely identify it. High-level descriptions of a TPM endorsement credential, a platform credential, an AIK credential and their relationship are shown in figure 1.

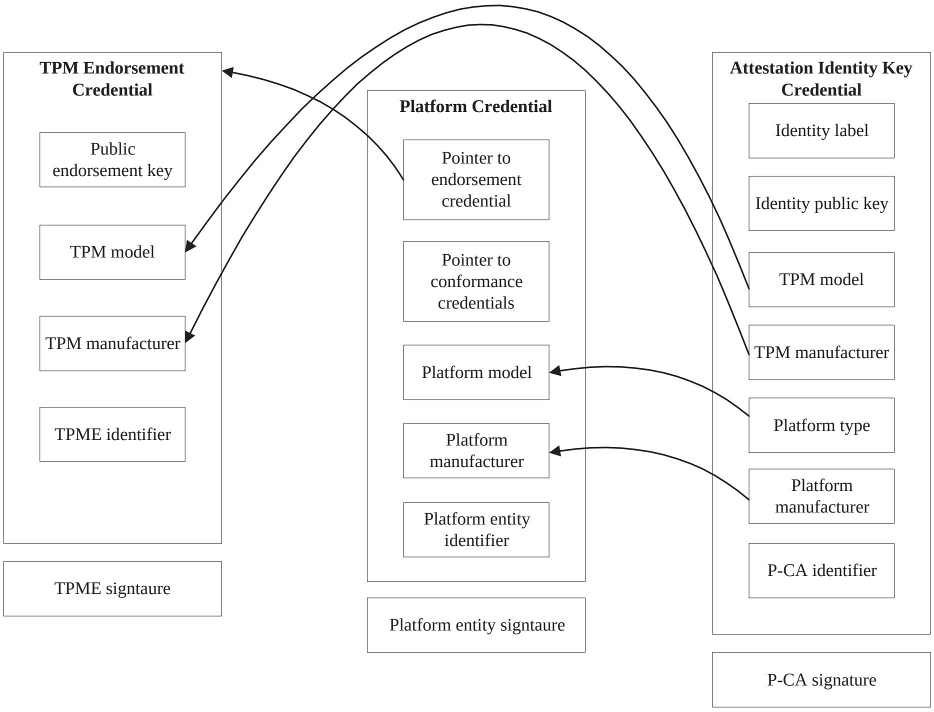

Figure 1: TP credentials [88]

The AIK private key is then used by the TPM during platform attestation. Note, the fact that a platform can generate arbitrary numbers of AIKs (and obtain associated credentials) enables a platform to obtain and use unlinkable pseudonyms, i.e. so that attestations to different third parties (or even to the same party) can be made unlinkable.

4.3.2 Platform Attestation

As stated above, platform attestation is a process by which a platform makes a verifiable claim about its current state, as captured by the current contents of its PCRs. The process starts with the *challenger*, i.e. the party wishing to have assurance about the current platform state, sending a nonce to the platform. The platform then uses one of its AIK private keys to sign a combination of this nonce and integrity metrics reflecting the current state of the platform.

This signed string is returned to the challenger, along with the record of the platform components which are reflected in the integrity metrics (i.e. (a portion of) the SML), together with the appropriate AIK credential. The challenger then uses this information to determine whether it is:

- safe to trust the TP from which the statement has originated, by verifying the TPM's signature and the AIK credential; and
- safe to trust (all or part of) the software environment running on the platform; this is achieved by validating the integrity metrics received from the TP using 'trustworthy' software integrity measurements attested to by trusted third parties such as software vendors.

4.4 Privacy-CAs and Direct Anonymous Attestation

As discussed by Brickell, Camenisch and Chen [17], version 1.2 of the TCG specifications incorporate the Direct Anonymous Attestation (DAA) protocol. This protocol is designed to address an anonymity issue in the previous version of the specifications. This arises from the use of AIK pairs by a TPM. Whilst a TPM can have an unlimited number of AIKs, acting as pseudonyms for the host platform, an entity which interacts with the platform needs to be given evidence that the public part of an AIK corresponds to a genuine trusted platform. As stated in the previous section, in systems conforming to the version 1.1 specifications this is achieved by using a special trusted third party called a Privacy-CA, which is responsible for generating certificates for the public AIKs. The problem with such an approach is that the Privacy-CA, when checking the credentials of a trusted platform, learns the fixed public endorsement key of a TPM, and hence has the means to link the public AIK back to a unique platform. That is, the Privacy-CA could collude with platform verifiers to break platform anonymity.

Of course, one solution to this would be for the owner of a trusted platform to choose a Privacy-CA which he/she trusts — indeed, for a corporate environment this should not be difficult to arrange. However, it was felt that this placed an unnecessary burden on the user, and for this reason the DAA protocol was introduced. This protocol enables a platform to simultaneously prove ownership of a DAA credential (provided by a DAA Issuer) and sign a message (e.g. containing the public part of an AIK). This proof of ownership does not reveal the credential, and so, even if the third party which issued the credential (the 'DAA Issuer') colludes with a verifier, it is still impossible to identify the platform.

However, even though DAA is a rather sophisticated protocol, which has provable security properties (see, for example, [15, 16]), possible problems remain (see also section 5.2 below). As discussed by Camenisch [18], the Privacy-CA scheme has the advantage that it makes it easier for compromised TPMs to be revoked; see also section 5.4. Camenisch also describes a means of improving the operation of DAA, which enables the identification of compromised TPMs without reducing platform privacy.

4.5 Isolated Execution Environments

An isolated execution environment, independent of how it is implemented, should provide the following services to hosted software [63]:

- protection of the software from external interference;
- observation of the computations and data of a program running within an isolated environment only via controlled inter-process communication;
- secure communication between programs running in independent execution environments; and
- a trusted channel between an input/output device and a program running in an isolated environment.

4.6 The Secure Boot Process

A secure boot process extends the integrity measurement and storage functionality described in section 4.1. During a secure boot, a platform's state is reliably captured, compared against measurements indicative of a trustworthy platform state, and then stored. If a discrepancy is discovered between the computed measurements and the expected measurements, then the platform halts the boot process.

5 Security Issues

The TCG specifications for the TPM are large and complex, and as a result it seems inevitable that some security vulnerabilities will be present. In fact, given this complexity, it is rather surprising that more issues have not been identified — in any event, detailed security analysis of the TCG specifications will remain an area of considerable importance for some time to come.

In this section we review those issues that have been raised to date. In each case we briefly review the main issue, and provide a brief assessment of its seriousness.

5.1 Use of Cryptographic Primitives

The cryptographic primitives used in the TCG specifications are to a significant extent 'hard coded' — for example, the hash-function used to compute integrity metrics, as stored in the PCRs of a TPM, is SHA-1. This hard coding is a problem for two main reasons.

Firstly, there is a known issue with SHA-1, namely the recent discovery of possible weaknesses [91] which indicates that collisions can be found significantly more easily than would ideally be the case. As a result, NIST has recently launched a competition to develop a new family of hash-functions [54], under the title SHA-3.

Secondly, apart from SHA-1, the use of cryptography in the TCG specifications is not in accordance with current best practice (or even the best practice at the time they were written). For example, RSA signatures and RSA encryption are not performed in provably secure ways. This contrasts with recent efforts by ISO and the IEEE to standardise best practice for asymmetric cryptography — see, for example, [26]. As a result, it would not be altogether surprising if cryptography-based attacks emerge over the next few years.

In retrospect it would almost certainly have been better to take a more generic approach to the use of cryptography, with the use of 'algorithm identifiers' to identify the algorithms employed. Moreover, it would also have been better to adopt the best practice for the use of asymmetric cryptography. It is understood that the TCG is developing a revised specification for the TPM (presumably to become version 1.3), and it is anticipated that some moves in this direction will be incorporated in these new documents.

As a side remark, it is interesting that Microsoft has chosen to compound this adoption of non-standard (and unproven) cryptographic techniques within the BitLocker drive encryption system, which forms part of its Vista operating system. This is of relevance here since BitLocker can exploit functionality provided by a trusted platform to provide hardware support for drive encryption. The Bitlocker scheme uses something called the 'Elephant Diffuser' in conjunction with AES encryption in CBC mode [33] to provide a degree of integrity protection for encrypted data. This scheme falls somewhat short of the level of protection provided by more widely known techniques for authenticated encryption (see, for example, [41]) — although its ambitions are more limited, and the BitLocker application requirements rule out standard techniques since the encrypted data string must be no longer than the plaintext data.

5.2 An Anonymity Attack

Rudolph [68] has recently pointed out a potential problem with the way that the DAA protocol is used in the TCG specifications. This problem means that it may be possible for a malicious DAA Issuer to enable DAA Verifiers to identify the trusted platform with which they are interacting, thereby breaching the anonymity properties of the protocol. This problem arises from the way that public keys are used by DAA Issuers.

Each DAA Issuer has its own long-term key pair, the public key of which is contained in a certificate generated by a CA. However, this key pair is not used directly in generating credentials for individual trusted platforms. Instead the DAA Issuer generates another key pair for this purpose, and it uses the long-term private key to sign the public key of this pair. This enables the DAA Issuer to update its key pair from time to time, without needing to obtain a new certificate from the CA.

Rudolph [68] points out that a malicious DAA Issuer could generate a large number of different key pairs, and use a different key pair for each credential that it generates. This would enable the DAA Issuer to collude with platform verifiers to identify individual trusted platforms.

However, as discussed in [45], it is not clear whether the Rudolph attack would work in practice. In particular, if a DAA Issuer did create a large number of different key pairs, then this would become obvious to the system users. It would be very simple to put in place monitoring procedures which could readily detect such behaviour by a DAA Issuer.

However, more subtle attacks along the lines proposed by Rudolph would be much more difficult to detect. In particular, if a DAA Issuer is only interested in tracking the behaviour of a small number of trusted platforms, then this could be achieved using only a correspondingly small number of distinct key pairs. Such tracking of a small subset of platforms would then be much harder to detect. Possible countermeasures to such an attack are discussed in [45], although this is clearly an area which merits further research.

5.3 Corrupt Administrator Attacks

Another possible privacy vulnerability in the implementation of the DAA protocol has been pointed out by Smyth, Ryan and Chen [75]. Giving a complete description of the attack requires going into the details of the DAA protocol, which is beyond the scope of this paper. Nevertheless, we attempt here to summarise the main issue giving rise to the vulnerability.

The DAA protocol is designed so that, if the same platform performs the DAA-signing protocol twice with the same DAA Verifier, then the Verifier can link these two transactions. The purpose of this linking is to enable rogue platforms, e.g. platforms using secrets obtained from a compromised TPM, to be identified. This linking is achieved by binding the name of the Verifier to a parameter used in the DAA signing process.

The problem identified in [75] arises if a corrupt DAA Issuer and the DAA Verifier use the same identifier. If this occurs, then these two entities can link the DAA credential (created by the DAA Issuer) with the DAA signature created by the trusted platform — that is, the Verifier can, with the help of a corrupt DAA Issuer, identify a trusted platform, breaking the anonymity property which DAA was create to provide.

As described in [75], this problem is simple to fix by making a small change to the way the DAA protocol is used in the TCG specifications. Like the attack described in section 5.2, it can be seen as a way in which a provably secure protocol has undesirable properties when used in particular circumstances. This suggests that there may be yet further unexpected vulnerabilities in DAA and/or other security features of the TPM, despite the soundness of the underlying cryptographic protocols and primitives.

5.4 Practical Issues

We also highlight a number of problems relating to the establishment of the infrastructure necessary to support the full TCG vision, as discussed in [9]. Topics addressed include issues with setting up and maintaining the Public Key Infrastructure (PKI) required to support the full set of trusted computing functionality, the practical use and verification of attestation evidence, and backwards compatibility, usability and compliance issues.

The deployment and use of trusted computing services is dependent on a fully functioning Trusted Computing PKI (TC-PKI), which is currently unavailable. While the challenges of PKI deployment are well-documented [38, 65], implementing a TC-PKI introduces additional considerations. As described in section 4.3.1, not only does a TC-PKI involve a large number of CAs, but there are also a series of implicit dependencies between these CAs. For example, a platform CA depends on the due diligence of an endorsement CA and one or more conformance CAs. Furthermore, these dependencies are currently only informally defined and there is therefore no indication as to where any liability will lie. While certificate polices and certification practice statements are traditionally used to define and limit the liability of CAs to relying parties, they are difficult and costly to create.

Issues are further complicated because of the reliance of every CA within the TC-PKI on the endorsement CA and, indeed, on the validity of the endorsement credential. This implies that, if a TP's endorsement key is compromised and the endorsement credential is revoked, all dependent credentials must in turn be revoked. This would include the TP's platform credential and all attestation identity credentials. Contacting all relevant CAs regarding the revocation decision may prove costly and time-consuming.

Further issues arise with respect to TPM compromise and revocation. A TPM is only required to be tamper-evident rather than tamper resistant. An attack enabling a TPM's PCRs to be reset without a platform reboot, thereby destroying the transitive chain of trust upon which a remote verifier relies to assess a platform, has recently been documented [78]. The demonstration of a PCR reset using a relatively unsophisticated hardware attack underlines the need for a verifier to consider the quality of a trusted platform when assessing its state; this also suggests that it may not be long before a TPM is completely compromised, and all its keys and credentials extracted. To help address this threat, a TPM's interactions with P-CAs and DAA Issuers could be monitored, and an excessive number of certification requests from a particular TPM could be used to detect a compromised platform. A number of issues arise with this approach.

- CAs might specify different thresholds for determining what is meant by "excessive", potentially leading to a high number of false positives for CAs with low thresholds.
- Once a compromised TPM has been detected, there is a need to globally propagate this information to prevent the compromised TPM host platform from being (mis)used elsewhere. This requires the establishment of a global revocation infrastructure.

- A CA must consider potential legal issues that might result from the wrongful issuance of revocation statements damaging a platform's ability to interact with other parts of the infrastructure.

- To alleviate the risk of a malicious P-CA issuing falsified revocation statements, a means by which the credibility of CAs in issuing such statements can be assessed must be provided.

Problems relating to the binary representation of platform state information have also been widely documented [9, 70]. The exact parameters to be considered when performing integrity measurements on platform components have yet to be standardised.

Because of the limited number of PCRs in a TPM, each PCR can be used to record multiple measurements. However, as the number of platform components increases, so does the complexity of third party verification of attestation statements. It also becomes difficult for a challenger to verify a single component running on a platform.

Problems relating to platform component updates and patching are also likely to arise. The order in which patches are applied can result in a "combinatorial explosion" of distinct configurations for a single application, each configuration requiring a distinct reference value for attestation purposes. Frequent patching may also lead to problems with sealed data. If an update or patch is applied to a software component to which a key or data is sealed, this key or data must be unsealed and resealed to the updated software component measurements. Failure to reseal to the updated component measurements will result in the key or data being inaccessible after the patch has been applied.

The issue of user observable verification must also be considered. McCune et al. [51] describe a scenario in which a user's platform has become infected with malware. Despite the fact that this infection can be detected by an external entity during an attestation process, the external entity has no way of reliably informing the end user that the user platform has 'failed' the attestation process. Malware might simply modify the user's display, resulting in the user believing their platform to be in an acceptable state, and, because of this, going on to disclose sensitive information to the malware.

As a consequence of the piecemeal roll-out of TC technologies, current trusted platforms do not come equipped with CRTMs, isolation technologies, processors or chipset extensions. Instead, current trusted platforms include only a TPM meeting the relevant TCG specifications, and, with the exception of Infineon TPMs, do not even include endorsement credentials. To the best of our knowledge, all currently available platforms lack both conformance credentials and platform credentials. This situation has the potential to create an awkward backward compatibility issue as and when fully-deployed TC-PKIs become available. In particular, the absence of these credentials will make it difficult, if not impossible, for a platform to later acquire AIK credentials without operating at reduced assurance levels.

The absence of CRTMs, isolation technologies, processors and chipset extensions in current TPM-enabled platforms makes the use of much of the TC functionality described in section 4 essentially unreliable. Techniques such as sealing and attestation are unworkable if the host platform's state cannot be reliably measured. In order to enable these features on an already deployed platform, measurement functionality (in the form of a CRTM and modified operating system) would need to be integrated into the platform after deployment. This would require the installation of a new OS and the BIOS to be flashed, tasks that would prove difficult for the average user. Furthermore, hardware-based isolation, enabled through the processor and chipset extensions, cannot be retrofitted to platforms already in the field. As a result, it seems unlikely that first generation trusted platforms can ever be adequately upgraded to provide all the services associated with a trusted platform, although the problems might be more tractable in a corporate environment.

Problems are also foreseen with respect to usability and conformance. Enabling a TPM prior to its use is a non-trivial task, which requires a user to understand and edit BIOS settings. Once enabled, a user is further confronted with setting a TPM owner password, selecting key types fit for purpose, and enrolling certain keys within a PKI. Further problems may arise with respect to password use and management, as unique passwords may be associated with the TPM owner as well as with data and keys protected by a TPM. While the deployment of multiple passwords may be viewed as a sound security decision, management of such passwords so that access is not jeopardised may prove problematic.

Finally, as many of the additional technological building blocks required to instantiate a trusted platform are not standardised, and the TCG does not dictate implementation specifics to its adopters, a number of currently available TPMs do not comply with the TPM specifications [69]. This latter issue may, however, be mitigated by the development (currently ongoing) of compliance and interoperability test suites for TPMs.

6 Applications

6.1 Existing Applications

Perhaps the most widely discussed existing application of trusted computing technology is that provided within certain versions of Windows Vista. Most notably, if present, a TPM can be used to enhance the security provided by the BitLocker drive encryption feature of Vista[1].

BitLocker is a full disk encryption feature, i.e. it enables the encryption of the entire system volume. By default it uses the AES encryption algorithm [55] in CBC mode [40] with a 128-bit key, combined with the Elephant diffuser [33] for additional security (see also section 5.1). The main objective of BitLocker is to protect against the compromise of data stored on computers that are lost or stolen, as well as more secure data deletion when BitLocker-protected computers are decommissioned. On computers that have a version 1.2 TPM, BitLocker uses the TPM to help ensure that stored data is accessible only if the computer's boot components are unaltered and the encrypted disk is

[1] BitLocker is only included in the Enterprise and Ultimate editions of Vista.

located in the original computer.

BitLocker has three modes of operation: *Transparent operation*, *User authentication*, and *USB key*. The first two modes require a version 1.2 TPM to be present, and the PC to possess a compatible BIOS. The three modes differ primarily in how the key necessary to decrypt the system volume is provided to the operating system during the boot process.

- Transparent operation mode uses the TPM to provide a transparent user experience. The key used for disk encryption is sealed (encrypted) by the TPM chip, and is only released to the operating system if the early boot files appear to be unmodified. The pre-operating system components of BitLocker check these files by implementing a Root of Trust for Measurement.

- In User authentication mode, the user must provide authentication information to the pre-boot environment in order to be able to boot the operating system. Two authentication modes are supported; either a pre-boot PIN must be entered by the user, or a USB device must be inserted that contains the required startup key.

- Finally, in USB key mode (which does not require a TPM), the user must insert a USB device containing a startup key into the computer in order to be able to boot the protected operating system. This mode requires that the BIOS on the protected machine supports use of USB devices in the pre-operating system environment.

Apart from this, a variety of vendor-specific software is provided by PC manufacturers with TC-enabled PCs. For example, HP provide a feature called *HP ProtectTools Embedded Security* on some of their PCs. This functionality uses the TPM to 'enhance native Microsoft operating system file and folder encryption and lays the foundation for authentication of TPM-enabled PCs to the corporate network'. Similarly, IBM provides trusted computing based enhanced security features for its PCs.

This relative shortage of available applications contrasts with the rapidly growing academic literature on possible uses of the technology. Proposals include uses to enhance the security of a variety of client, network and mobile applications; we examine a sample of these immediately below.

6.2 Client Applications

Client applications performing functions such as digital signature generation and verification, defence against crimeware, completion of private electronic transactions, and Internet-based card-not-present transactions, could benefit from the deployment of TC technologies. We briefly examine possible approaches of this type.

The use of TC functionality has been proposed to enhance the security of the digital signature process [7, 77]. Spalka, Cremers and Langweg [77] suggest the use of a secure boot process to provide assurance that security critical signature software is executing as expected on a platform. Balacheff et al. [7] describe how platform attestation could be used to verify the state of a trusted display controller. Given a successful verification, an end user can have confidence that what they see displayed on the screen is what they will digitally sign.

Balfe et al. [10] examine how TC could be used to defend against the ever-growing threat posed by crimeware. For example, a platform, on requesting access to a company's intranet, could 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. Using sealed storage, an end user can protect private data (e.g. credit card numbers) by making access to that data contingent on a platform being in a particular state. For example, a user could seal credit card data to a state that requires a particular banking application to be running on the platform, and nothing else. The presence of crimeware would change the platform state and prevent access. Secure boot functionality could be used to detect the malicious or accidental modification or removal of security-critical software at boot time. For example, the Subvirt rootkit, which modifies a system's boot sequence, could be detected by such functionality. 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 could be used to ensure that any infection is contained within the crimeware-infected execution environment.

The OpenTC project has developed a method using trusted computed functionality by which electronic transactions, such as online banking, can be securely performed with well-known and trusted entities. In this case a system is defined as 'secure' when "(a) a user can validate a given virtual machine that is used for commercial transactions and can convince others of its integrity, and (b) that the user's secrets are securely stored throughout the life-cycle of the virtual machine" [44].

Two trusted computing-based schemes have been proposed to help prevent phishing attacks, network redirection to fake web servers, exploitation of client software vulnerabilities, and/or modification of a client configuration. In both solutions an isolated execution environment is used to compartmentalise a browser on which the private electronic transaction (PET) application runs, so that it is isolated from all other software executing on the platform. Platform software, up to and including the isolation layer (i.e. the Trusted Computing Base (TCB) of the client platform which is assumed to be correct) and the compartment running the browser for the PET application, are measured during an authenticated boot.

In the first solution, which requires modifying both the client and server methods for completing web transactions, the system is configured so that the trusted compartment in which the browser is running looks different from any untrusted compartments running on the platform. In addition, the appearance of the browser changes when connected to the correct website (i.e. following a successful TLS connection). Clients are required to attest to their state, and all outbound traffic is routed through a dedicated compartment (regarded as part of the TCB and therefore correct) which only forwards traffic to a specified list of trusted websites.

The other solution only involves modifying the client method for completing web transactions. In this case the trusted compartment is securely booted, and a user's credentials are sealed to the trusted compartment so that they can only be released when the compartment is connected to the correct web site. Identification of the web site is triggered by a successful TLS server authentication.

Balfe and Paterson [12] examine how the staged roll-out of trusted computing technology, beginning with ubiquitous client-side TPMs and culminating in trusted computing with processor, chipset and OS support [13], can be used to enhance the security of Internet-based 'cardholder not present' transactions. In [13], a system that makes use of the full spectrum of TC technologies to securely emulate point-of-sale Integrated Circuit Cards compliant with the Europay Mastercard and Visa (EMV) specifications [27, 28, 29, 30] is described. Emulation of EMV-compliant cards confers tamper-resistance properties that are normally associated with physical EMV card use at point-of-sale terminals, making it possible to demonstrate card ownership and cardholder authentication.

Further client applications are described in [37, 44, 59, 93].

6.3 Network Applications

The use of trusted computing functionality has also been proposed to harden network access control, to support secure single sign-on solutions [57], to secure peer-to-peer networks [11, 43, 73, 74], to improve the security and privacy of a biometric user authentication process [21], and to support identity management [52, 53]. A number of authors have also considered trusted computing's applicability to the agent paradigm [25, 58, 61, 66] and grid security [48, 49].

Attestation features form an important focus of the TCG's Trusted Network Connect (TNC) specifications [89]. TNC offers a way of verifying an endpoint's integrity to ensure that it complies with a particular predefined policy before it is granted network access. The process of checking end-point integrity for compliance with policy occurs in three distinct phases: assessment, isolation and remediation. The assessment phase involves a platform attesting to its current state. A server examines this attestation, compares the platform's integrity metrics to its network access policies, and as a result allows access, denies access or places the platform in an quarantined network (isolation). In this latter case, a platform would typically be able to obtain the requisite integrity-related updates that will allow it to satisfy the server's access policy and be granted access.

Trusted computing functionality could also be used to support secure single sign-on solutions [57]. Single sign-on enables a user to authenticate only once to an Authentication Service Provider (ASP), and then subsequently request services from a variety of Service Providers (SPs) without necessarily re-authenticating. Information about the user's authentication status is handled between the ASP and the desired SP transparently to the user. As described in [57], the end-users computing platform could itself play the role of the

ASP. In this case, authenticated boot, protected storage, and attestation functionality could be used to enable an SP to check the authenticity of the user's TP and the integrity of its software state before trusting any authentication assertions.

Trusted computing has also been suggested for use in securing peer-to-peer networks [11, 43, 73, 74]. Balfe, Lakhani and Paterson describe a pseudonymous authentication scheme for peers based on DAA, and extend this scheme to build secure channels between peers for future communications.

Chen, Pearson and Vamvakas [21] examine how trusted computing functionality could be used to improve the security and privacy of a biometric user authentication process in a distributed environment. A user can establish trust in a biometric system by verifying the state attested to by the system, and the user can hence trust that the system (incorporating both reader and platform) will not disclose his or her sensitive biometric information to an unauthorised entity.

Mont et al. [52, 53] describe a system supporting identity management. This system obfuscates user information before it is sent to external entities, and associates policies with the obfuscated data detailing the associated constraints on disclosure. The system enforces tracing and auditing of disclosures to increase a receiver's accountability to a trusted third party. Trusted computing functionality enables the owner of the private data, the recipient of the private data, and any third party users to verify the integrity of all parties with which they must interact, prior to the disclosure of any sensitive information.

A number of authors have considered trusting computing's applicability to the agent paradigm [25, 58, 61, 66]. The use of trusted hardware to protect mobile agents can be traced back to Wilhelm et al.'s work [92] on adding trusted third parties (in the form of an isolated hardware environments) to host systems. In many ways this work was rather prophetic, in that the requirements for trusted hardware are very much mirrored by trusted computing. Wilhelm et al. define a Trusted Processing Environment (TPE) to consist of a CPU, RAM, ROM, and non-volatile storage, all of which execute within a virtual machine. An agent executes within this environment, and hence the host OS will not be able to observe its execution. When it was proposed, Wilhelm's system would have been proprietary and expensive; however, this is no longer the case, as TPMs and LaGrande/Pacifica enabled hardware will soon be ubiquitous in the marketplace.

The use of trusted computing in agent systems has also been proposed in [25, 58, 61, 66]. Both [25] and [58] deal with the use of non-mobile agents to help preserve user privacy. In [61] a number of proposals are made to enhance privacy protection for mobile applications. Each uses sealed storage functionality and exploits the benefits of the ability to recognise when a platform will behave as expected. Recently [66], a trusted computing enhanced mobile agent platform called SMASH was proposed. In this system, trusted computing is deployed to form a middleware-based instantiation of some aspects of Wilhelm et al.'s scheme [92]. In [8], the TPM's sealing mechanism is used in various ways to protect security critical functions within a mobile agent. The use of sealing in this context provides a mechanism to enable a mobile agent to securely migrate from one platform to another, whilst retaining a guarantee that the next visited platform will behave as expected to meet the agent's objectives.

The application of trusted computing to Grid Computing has been widely discussed (see, for example, [23, 24, 50, 94]). Much of this work aims to prevent or detect resource provider misbehavior. Mao et al. [19] propose Daonity, a system to establish a relocatable key, which enables controlled group sharing of encrypted content. Löhr et al. [48] propose a scheme in which resource providers publish attestation tokens containing public keys from non-migratable TPM key pairs and the platform states to which private key use is bound. Each token is signed to prove that it was produced by an authentic TPM.

6.4 Mobile Applications

Whilst trusted computing technology is already becoming commonplace in new PCs, at least as far as the inclusion of TPMs is concerned, the situation is not so advanced for other types of platform. In particular, whilst many potential applications for the technology can be identified for mobile devices (e.g. PDAs, smart phones, etc.), the inclusion of TPMs in such platforms has yet to occur.

Indeed, for a variety of reasons, including cost and complexity, it would appear that trusted computing technology may be implemented in rather different ways in mobile devices. In particular, it seems that such devices may not include an identifiable separate TPM, but instead the functionality of the TPM could be implemented using a combination of trusted hardware functionality built into a mobile platform and software. How this might be achieved will probably vary widely from manufacturer to manufacturer.

The TCG has always had the mission of providing specifications for any type of device that connects to a network. However, the initial standardisation work centred around the specification of the TPM and a standard set of APIs which provide an abstraction of the TPM to software developers/vendors. More recently, the baseline TCG specification set has been expanded by platform-specific working groups to include specifications describing specific platform implementations for PC clients, servers, peripherals and storage systems.

One such working group is the TCG Mobile Phone Working Group (MPWG), the main challenge for which is to determine the 'roots of trust', see section 3, required within a trusted mobile phone. In order to identify the capabilities required of a trusted mobile phone, a number of use cases have been identified by the MPWG, whose secure implementation may be aided by the application of trusted platform functionality. Among these are SIMLock, device authentication, mobile ticketing, mobile payment, and robust Digital Rights Management (DRM) [80]. As stated by the MPWG [80], the use cases lay a foundation for the ways in which:

- the MPWG derives requirements that address situations described in the use cases;

- the MPWG specifies an architecture based on the TCG architecture that meet these requirements; and
- the MPWG specifies the functions and interfaces that meet the requirements in the specified architecture.

In 2006, the MPWG published the TCG Mobile Trusted Module (MTM) Specification [84]. It is assumed that a mobile platform will typically contain multiple MTMs to support multiple mobile device stakeholders. It is envisaged that each MTM will provide a subset of the TPM v1.2 functionality. Some MTMs may also contain additional functionality to ensure that parts of the device boot into a preset state (i.e. secure boot functionality) [82]. More specifically, two types of MTM have been defined.

A *Mobile Local-owner Trusted Module (MLTM)* supports uses (or a subset of uses) similar to those of existing v1.2 TPMs (controlled by an entity with physical access to the platform). Some TPM v1.2 functionality may not be supported because of the restrictions inherent in today's phone technologies [82]. The use cases described by the TCG in [80] have been analysed, along the lines of the analyses given in [34], in order to determine the subset of functionality required within an MTM to support them.

A *Mobile Remote-owner Trusted Module (MLTM)* also supports a subset of uses similar to those of existing v1.2 TPMs. It moreover enables a remote entity (such as the device manufacturer or network operator) to predetermine the state into which some parts of the phone must boot [82].

We focus here on three specific use-cases, namely robust implementations of OMA DRM v2, secure SIMLock, and secure software download.

6.4.1 OMA DRM v2

The OMA DRM version 2 specifications describe a DRM system for a mobile environment [56]. It is supported by a 'trust model' which has been defined by the Content Management Licensing Administrator for Digital Rights Management (CMLA DRM). Such a trust model enables a rights issuer to obtain assurances about the robustness of an OMA DRM v2 implementation on a mobile device [56].

The following trusted computing functionality could be used to meet the robustness rules defined by the CMLA. While TC functionality cannot guarantee the integrity of the OMA DRM v2 agent while it is being stored, a secure boot process can help detect malicious or accidental modifications or removal. Stored security-critical data associated with the OMA DRM v2 agent requiring integrity protection can also be verified as part of a secure boot process.

Alternatively, sealed storage functionality could be used to detect the malicious or accidental modification or removal of the mobile device OMA DRM v2 application while in storage, and indeed, to store data and/or keys which need to be confidentiality and/or integrity-protected. It can also ensure that sensitive data is only accessible by authorised entities when the mobile device is in a predefined state, for example when a legitimate OMA DRM v2 application is executing in an isolated execution environment. Isolation technologies, in turn, enables security-critical software and data to be isolated in a secure execution environment, so that it cannot be observed and/or modified by software executing in parallel execution environments.

A good quality random number generator is provided by a TPM, enabling the generation of non-repeating unpredictable nonces for use in the OMA DRM protocols. The TPM can also be used to provide accurate time source synchronisation, as described in [85].

6.4.2 Secure SIMLock

Mobile device personalisation, or SIMLocking (see [1]), enables a device to be constrained to operate only with (U)SIMs associated with a network, network subset, service provider, corporate customer, or, indeed, with a unique (U)SIM; this is achieved using a pre-defined series of *personalisation categories*. Each category has an independent personalisation indicator, used to show whether a particular personalisation category is active (on) or deactivated (off). An independent personalisation code or code group is defined for each category; this indicates how the device is personalised for this category (e.g. which networks the device will work with).

When a (U)SIM is inserted into a device, or when a device is powered on, it checks which personalisation indicators are set to ‘on’. The personalisation agent then reads the (U)SIM, and extracts the required code(s)/code group(s). The code(s)/code group(s) are then verified against the list of values stored on the mobile device. The mobile device then responds accordingly, displaying a message of success or failure to the device user.

Unauthorised modification or removal of the device personalisation agent cannot be prevented using trusted computing technology. However, while the software is stored, secure boot functionality can be used so that, at start-up, a measurement of the device personalisation agent software is verified against an expected value. This enables any unauthorised modification and/or removal to be detected. Any security-critical data requiring integrity protection, such as network, network subset, corporate and service provider codes or code groups and indicators, can also be covered by the secure boot process. Isolation technologies can be used to ensure the integrity of the personalisation agent, and that any security-critical data is protected while in use on the device.

Alternatively, personalisation codes/code groups, personalisation indicators and control keys could simply be sealed to an isolated execution environment which hosts a device personalisation agent. In this way, security-critical data can be both integrity and confidentiality-protected while stored. If the personalisation agent, and/or the supporting environment to which the data is sealed, is modified, then the security-critical data will be inaccessible. While sealing ensures that data is released into a predefined execution environment, isolation technologies are necessary to ensure that both the device personalisation agent and the security related data remain confidentiality and integrity-protected while in use on the platform.

6.4.3 Secure Software Download

A Software Defined Radio (SDR) is a communications device "whose operational modes and parameters can be changed or augmented, post manufacturing via software" [76]. This implies that the device can be reconfigured to communicate using multiple frequency bands and protocols, or upgraded in a low cost and efficient manner. Trusted computing functionality can be deployed to protect the downloaded SDR software from the host and, indeed, the mobile host from the downloaded software.

TC mechanisms could be used to confidentiality-protect the reconfiguration software in transit between the software provider and the end host, while stored or executing on the end host, and to ensure that only the intended recipient device can access the software. [35, 36] describe a secure software download protocol using trusted computing functionality, or, more specifically, sealed storage, platform attestation, and isolation techniques.

Alternatively, if a more traditional mechanism such as SSL/TLS is used to provide secure download of the reconfiguration software, TC functionality can be used to 'harden' the SSL/TLS implementation. In this case, prior to the completion of any SSL/TLS protocol, the TPM is used to generate the client-side (SDR device) key pair for SSL authentication, which is bound to a set of integrity metrics so that the private key can only be used by the TPM on which it was generated and when the TPM host platform is in the required state. This hardened implementation of SSL/TLS gives the software provider some assurance that the SDR device's SSL/TLS private key is stored securely and cannot been stolen. Evidence of the device's ability to provide an isolated execution environment for the downloaded software can also be demonstrated.

A capability exchange can be completed by the network and the SDR prior to software download to ensure that the appropriate software entities and parameter sets are selected for a particular SDR device. The use of platform attestation could be used to ensure that the reports sent by the device are accurate.

While the integrity of stored security-critical host software cannot be ensured using TC functionality, a secure boot procedure can be used to help detect its malicious or accidental modification or removal. TC functionality also enables the isolation of security-critical software in a secure execution environment so that it cannot be observed or modified by software executing in a parallel insecure execution environment.

Also, if the downloaded software is isolated in its own execution environment, then any malicious behavior can be controlled and its effects limited. If sealed storage is used by the end user to protect private data (e.g., credit card numbers), then the impact of malicious software may be lessened, as it cannot gain access to security sensitive data which has been protected. On reconnection to a commercial network, a trusted SDR device could be required to attest to its state so that a decision can be made as to whether or not the device should be authorised to access the network.

7 The Future

We conclude this paper by briefly considering the future of trusted computing technology. Perhaps the most fundamental question regards whether or not the technology will succeed in its objectives. That is, will it really be possible to use trusted computing to determine the state of a remote PC, and to seal data to the state of a PC with confidence that this sealing will work effectively. For this to be possible requires a number of obstacles to be overcome, notably:

- the hardware must become ubiquitous;
- the infrastructure necessary to support use of trusted computing must be established (as discussed in section 5.4); and
- virtualisation technology must become widely available on desktop and notebook PCs, using techniques which enable the virtualisation layer to be verified using trusted computing.

With regard to the first point, the prospects are good, since the technology has already been deployed in desktop and notebook PCs. Moreover, work is under way within the MPWG to support the provision of trusted computing functionality on mobile phones, suggesting that the technology will become even more widespread.

Overcoming the second obstacle is more problematic, although within a corporate setting the obstacles may be much less significant. Indeed, it seems likely that the technology will succeed first in a corporate setting rather than for home use.

The third issue is almost certainly the most difficult of all. Despite much activity over the past few years, including white papers and announcements at conferences and shows, Microsoft has provided only very limited support for trusted computing in Windows Vista. The NGSCB vision appears to be a long way from being realised. Microsoft's future plans in this direction are far from clear.

However, much greater support for trusted computing technology is emerging from the open source community, and from collaborative research projects such as OpenTC and EMSCB. Open source trusted virtualisation layers are being developed by both the Xen and L4 communities. Thus it may well be that open source users will be able to enjoy the benefits of trusted computing based security long before Windows users — we must wait and see!

Acknowledgements

The development of this article was sponsored by the Open Trusted Computing project of the European Commission Framework 6 Programme. We would like to thank Stéphane Lo Presti and other partners in the OpenTC project for valuable guidance and advice over the last couple of years.

References

[1] 3rd Generation Partnership Project (3GPP), Global System for Mobile Communications (GSM) — Technical Specification Group Services and System Aspects, Sophia Antipolis, France, *Personalisation of mobile equipment (ME), Mobile functionality specification (release 5)*, 2002.

[2] D. Abraham, J. Jackson, S. Muthrasanallur, G. Neiger, G. Regnier, R. Sankaran, I Schionas, R. Uhlig, B Vembu, and J. Wiegert, *Intel virtualization technology for directed i/o*, Intel Technology Journal **10** (2006), no. 3, 179–192.

[3] R. Anderson, *Cryptography and competition policy — Issues with 'trusted computing'*, Proceedings of PODC '03, July 13-16, 2003, Boston, Massachsetts, USA, ACM, 2003, pp. 3–10.

[4] B. Arbaugh, *Improving the TCPA specification*, IEEE Computer **35** (2002), no. 8, 77–79.

[5] W. A. Arbaugh, D. J. Farber, and J. M. Smith, *A secure and reliable bootstrap architecture*, Proceedings of the 1997 IEEE Symposium on Security and Privacy (S&P 1997) (Oakland, California, USA), IEEE Computer Society Press, Los Alamitos, California, May 1997, pp. 65–71.

[6] B. Balacheff, L. Chen, S. Pearson, G. Proudler, and D. Chan, *Computing platform security in cyberspace*, Information Security Technical Report **5** (2000), no. 1, 54–63.

[7] B. Balacheff, L. Chen, D. Plaquin, and G. Proudler, *A trusted process to digitally sign a document*, Proceedings of the 2001 New Security Paradigms Workshop (V. Raskin and C. F. Hempelmann, eds.), ACM Press, 2001, pp. 79–86.

[8] S. Balfe and E. Gallery, *Mobile agents and the deus ex machina*, Proceedings of the 2007 IEEE International Symposium on Ubisafe Computing (UBISAFE 2007), vol. 2, IEEE Computer Society, 2007, pp. 486–492.

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

[10] ———, *Crimeware and trusted computing*, Crimeware: Understanding new attacks and defenses (M. Jakobsson and Z. Ramzan, eds.), Addison-Wesley, Upper Saddle River, NJ, 2008, pp. 457–472.

[11] S. Balfe, A. D. Lakhani, and K. G. Paterson, *Securing peer-to-peer networks using trusted computing*, Trusted Computing (C. J. Mitchell, ed.), IEE Press, London, 2005, pp. 271–298.

[12] S. Balfe and K. G. Paterson, *Augmenting internet-based card not present transactions with trusted computing: An analysis*, Tech. Report RHUL-MA-2006-9, Department of Mathematics, Royal Holloway, University of London, 2005.

[13] ______, *e-EMV: Emulating EMV for internet payments using trusted computing technology*, Tech. Report RHUL-MA-2006-10, Department of Mathematics, Royal Holloway, University of London, 2005.

[14] 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*, Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003), Bolton Landing, New York, USA, 19–22 October 2003, ACM Press, New York, USA, 2003, pp. 164–177.

[15] E. Brickell, J. Camenisch, and L. Chen, *Direct anonymous attestation*, Tech. Report HPL-2004-93, Hewlett-Packard Laboratories, June 2004, Available at `http://www.hpl.hp.com/techreports/`.

[16] ______, *Direct anonymous attestation*, Proceedings of CCS '04 (B. Pfitzmann and P. Liu, eds.), ACM Press, 2004, pp. 132–145.

[17] ______, *The DAA scheme in context*, Trusted Computing (C. J. Mitchell, ed.), IEE Press, London, 2005, pp. 143–174.

[18] J. Camenisch, *Better privacy for trusted computing platforms (extended abstract)*, Computer Security — ESORICS 2004, 9th European Symposium on Research in Computer Security, Sophia Antipolis, France, September 13-15, 2004, Proceedings (P. Samarati, D. Gollmann, and R. Molva, eds.), Lecture Notes in Computer Science, vol. 3193, Springer-Verlag, Berlin, 2004, pp. 73–88.

[19] H. Chen, J. Chen, W. Mao, and F. Yan, *Daonity — Grid security from two levels of virtualisation*, Information Security Technical Report **12** (2007), no. 3, 123–138.

[20] L. Chen, S. Pearson, G. Proudler, D. Chan, and B. Balacheff, *How can you trust a computing platform?*, Proceedings of Information Security Solutions Europe (ISSE 2000), 2000.

[21] L. Chen, S. Pearson, and A. Vamvakas, *On enhancing biometric authentication with data protection*, Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies (R. J. Howlett and L. C. Jain, eds.), vol. 1, IEEE, 2000, pp. 249–252.

[22] P. C. Clark and L. J. Hoffman, *BITS: a smartcard protected operating system*, Communications of the ACM **37** (1994), 66–94.

[23] A. Cooper and A. Martin, *Towards a secure, tamper-proof grid platform*, Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid, Singapore, May 2006, IEEE Press, 2006, pp. 373–380.

[24] ______, *Towards an open, trusted digital rights management platform*, Proceedings of the ACM workshop on Digital rights management (DRM '06), Alexandria, Virginia, USA, October 30, 2006, ACM Press, 2006, pp. 79–88.

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

[26] A. W. Dent and C. J. Mitchell, *User's guide to cryptography and standards*, Artech House, Boston, MA, 2005.

[27] EMVCo, *Book 1 — Application independent ICC to terminal interface requirements*, May 2004, Version 4.1.

[28] EMVCo, *Book 2 - Security and Key Management*, May 2004, Version 4.1.

[29] EMVCo, *Book 3 - Application Specification*, May 2004, Version 4.1.

[30] EMVCo, *Book 4 - Cardholder, Attendant, and Acquirer Interface Requirements*, June 2004, Version 4.1.

[31] P. England, B. Lampson, J. Manferdelli, M. Peinado, and B. Willman, *A trusted open platform*, IEEE Computer **36** (2003), no. 7, 55–62.

[32] P. England and M. Peinado, *Authenticated operation of open computing devices*, Information Security and Privacy, 7th Australasian Conference, ACISP 2002, Melbourne, Australia, July 3-5, 2002, Proceedings (L. Batten and J. Seberry, eds.), Lecture Notes in Computer Science, vol. 2384, Springer-Verlag, Berlin, 2002, pp. 346–361.

[33] N. Ferguson, *AES-CBC + Elephant diffuser: A disk encryption algorithm for Windows Vista*, Microsoft Corporation, August 2006.

[34] E. Gallery, *Authorisation issues for mobile code in mobile systems*, Tech. Report RHUL-MA-2007-3, Department of Mathematics, Royal Holloway, University of London, 2007.

[35] E. Gallery and A. Tomlinson, *Protection of downloadable software on SDR devices*, Proceedings of the 4th Software Defined Radio Forum Technical Conference (SDR 2005) (Orange County, California, USA), Software Defined Radio Forum (SDRF), November 2005.

[36] ______, *Secure delivery of conditional access applications to mobile receivers*, Trusted Computing (C. J. Mitchell, ed.), The Institute of Electrical Engineers (IEE), London, UK, 2005, pp. 195–238.

[37] T. Garfinkel, M. Rosenblum, and D. Boneh, *Flexible OS support and applications for trusted computing*, Proceedings of HotOS IX: The 9th Workshop on Hot Topics in Operating Systems, Lihue, Hawaii, USA, May 18–21, 2003, USENIX Association, 2003, pp. 145–150.

[38] P. Gutmann, *PKI: It's not dead, just resting*, IEEE Computer **35** (2002), no. 8, 41–49.

[39] Intel, *LaGrande technology architectural overview*, Tech. Report 252491-001, Intel Corporation, September 2003.

[40] International Organization for Standardization, Genève, Switzerland, *ISO/IEC 10116: 2006, Information technology — Security techniques — Modes of operation for an n-bit block cipher*, 3rd ed., 1997.

[41] International Organization for Standardization, Genève, Switzerland, *ISO/IEC FCD 19772, Information technology — Security techniques — Authenticated encryption mechanisms*, October 2007.

[42] N. Itoi, W. A. Arbaugh, S. J. Pollack, and D. M. Reeves, *Personal secure booting*, Information Security and Privacy, 6th Australasian Conference, ACISP 2001, Sydney, Australia, July 11-13 2001, Proceedings (V. Varadharajan and Y. Mu, eds.), Lecture Notes in Computer Science, vol. 2119, Springer-Verlag, Berlin, 2001, pp. 130–144.

[43] M. Kinateder and S. Pearson, *A privacy-enhanced peer-to-peer reputation system*, E-Commerce and Web Technologies, 4th International Conference, EC-Web, Prague, Czech Republic, September 2-5, 2003, Proceedings (K. Bauknecht, A. Min Tjoa, and G. Quirchmayr, eds.), Lecture Notes in Computer Science, vol. 2738, Springer-Verlag, Berlin, 2003, pp. 206–216.

[44] D. Kuhlmann, R. Landfermann, H. Ramasamy, M. Schunter, G. Ramunno, and D. Vernizzi, *An open trusted computing architecture — secure virtual machines enabling user-defined policy enforcement*, www.opentc.net, June 2006.

[45] A. Leung, L. Chen, and C. J. Mitchell, *On a possible privacy flaw in Direct Anonymous Attestation (DAA)*, Tech. Report RHUL-MA-2007-10, Mathematics Department, Royal Holloway, University of London, December 2007.

[46] D. Lie, *Architectural support for copy and tamper resistant software*, Ph.D. thesis, Department of Electrical Engineering, Stanford University, Stanford, California, USA, December 2003.

[47] D. Lie, C. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. Mitchell, and M. Horowitz, *Architectural support for copy and tamper resistant software*, Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX) (Cambridge, Massachusetts, USA), ACM Press, New York, USA, 12–15 November 2000, pp. 169–177.

[48] H. Löhr, H. V. Ramasamy, A.-R. Sadeghi, S. Schulz, M. Schunter, and C. Stüble, *Enhancing Grid security using trusted virtualization*, Proceedings of the 4th International Conference on Autonomic and Trusted Computing (ATC 2007), Lecture Notes in Computer Science, vol. 4610, Springer-Verlag, Berlin, 2007, pp. 372–384.

[49] W. Mao, F. Yan, and C. Chen, *Daonity: Grid security with behaviour conformity from trusted computing*, Proceedings of the 1st ACM workshop on Scalable Trusted Computing (STC 2006), ACM Press, New York, NY, USA, 2006, pp. 43–46.

[50] A. Martin and P.-W. Yau, *Grid security: Next steps*, Information Security Technical Report **12** (2007), no. 3, 113–122.

[51] J. McCune, A. Perrig, A. Seshadri, and Leendert van Doorn, *Turtles all the way down: Research challenges in user-based attestation*, Pro-

ceedings of 2nd USENIX Workshop on Hot Topics in Security (HotSec 2007), August 2007.

[52] M. C. Mont, S. Pearson, and P. Bramhall, *Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services*, 14th International Workshop on Database and Expert Systems Applications (DEXA '03), September 1-5, 2003, Prague, Czech Republic, IEEE Computer Society, 2003, pp. 377–382.

[53] ______, *Towards accountable management of privacy and identity information*, Computer Security — ESORICS 2003, 8th European Symposium on Research in Computer Security, Gjøvik, Norway, October 13-15, 2003, Proceedings (E. Snekkenes and D. Gollmann, eds.), Lecture Notes in Computer Science, vol. 2808, Springer-Verlag, Berlin, 2003, pp. 146–161.

[54] National Institute of Standards and Technology, Gaithersburg, MD, USA, *Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family*, November 2007, Docket No. 070911510-7512-01.

[55] National Institute of Standards and Technology (NIST), Gaithersburg, MD, *Federal information processing standards publication 197 (fips pub 197): Specification for the advanced encryption standard (aes)*, November 2001.

[56] The Open Mobile Alliance (OMA), *DRM architecture v2.0*, July 2004.

[57] A. Pashalidis and C. J. Mitchell, *Single sign-on using trusted platforms*, Information Security, 6th International Conference, ISC 2003, Bristol, UK, October 1-3, 2003, Proceedings (C. Boyd and W. Mao, eds.), Lecture Notes in Computer Science, vol. 2851, Springer-Verlag, Berlin, 2003, pp. 54–68.

[58] S. Pearson, *Trusted agents that enhance user privacy by self-profiling*, Tech. Report HPL-2002-196, Hewlett-Packard Laboratories, Bristol, UK, July 2002.

[59] S. Pearson (ed.), *Trusted computing platforms: TCPA technology in context*, Prentice Hall PTR, 2002.

[60] ———, *Trusted computing platforms, the next security solution*, Tech. Report HPL-2002-221, Hewlett-Packard Laboratories, November 2002, Available at `http://www.hpl.hp.com/techreports/`.

[61] ———, *How trusted computers can enhance for privacy preserving mobile applications*, Proceedings of the 1st International IEEE WoWMoM Workshop on Trust, Security and Privacy for Ubiquitous Computing (WOWMOM ’05), IEEE Computer Society, Washington, DC, USA, 2005, pp. 609–613.

[62] M. Peinado, Y. Chen, P. England, and J. Manferdelli, *NGSCB: A trusted open system*, Information Security and Privacy, 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15 2004, Proceedings

(H. Wang, J. Pieprzyk, and V. Varadharajan, eds.), Lecture Notes in Computer Science, vol. 3108, Springer-Verlag, Berlin, 2004, pp. 86–97.

[63] M. Peinado, P. England, and Y. Chen, *An overview of NGSCB*, Trusted Computing (C. J. Mitchell, ed.), IEE Professional Applications of Computing Series 6, The Institute of Electrical Engineers (IEE), London, 2005, pp. 115–141.

[64] B. Pfitzmann, J. Riordan, C. Stuble, M. Waidner, and A. Weber, *The PERSEUS system architecture*, Tech. Report RZ 3335 (#93381), IBM Research Division, Zurich Laboratory, April 2001.

[65] G. Price, *PKI — An insider's view (extended abstract)*, Technical Report RHUL-MA-2005-8, Department of Mathematics, Royal Holloway, University of London, June 2005.

[66] A. Pridgen and C. Julien, *A secure modular mobile agent system*, Proceedings of the 2006 international workshop on Software engineering for large-scale multi-agent systems (SELMAS '06), ACM Press, New York, NY, USA, 2006, pp. 67–74.

[67] J. Reid, J. M. Gonzalez Nieto, and E. Dawson, *Privacy and trusted computing*, 14th International Workshop on Database and Expert Systems Applications (DEXA '03), September 1-5, 2003, Prague, Czech Republic, IEEE Computer Society, 2003, pp. 383–388.

[68] C. Rudolph, *Covert identity information in Direct Anonymous Attestation (DAA)*, New Approaches for Security, Privacy and Trust in Com-

plex Environments: Proceedings of the IFIP TC-11 22nd International Information Security Conference (SEC 2007), 14-16 May 2007, Sandton, South Africa (H. Venter, M. Eloff, L. Lebuschagne, J. Eloff, and R. von Solms, eds.), IFIP International Federation for Information Processing, vol. 232, Springer, Boston, 2007, pp. 443–448.

[69] A.-R. Sadeghi, M. Selhorst, C. Stüble, C. Wachsmann, and M. Winandy, *TCG inside?: a note on TPM specification compliance*, Proceedings of the 1st ACM workshop on Scalable trusted computing (STC 2006) (Alexandria, Virginia, USA), ACM, New York, NY, USA, 2006, pp. 47–56.

[70] A.-R. Sadeghi and C. Stüble, *Property-based attestation for computing platforms: Caring about properties, not mechanisms*, Proceedings of the 2004 Workshop on New Security Paradigms (NSPW 2004) (C. F. Hempelmann, ed.), ACM, New York, NY, USA, 2004, pp. 67–77.

[71] A.-R. Sadeghi and C. Stüble, *Taming "trusted platforms" by operating system design*, Information Security Applications, 4th International Workshop, WISA 2003, Jeju Island, Korea, August 25-27, 2003, Revised Papers (K. Chae and M. Yung, eds.), Lecture Notes in Computer Science, vol. 2908, Springer-Verlag, Berlin, 2004, pp. 286–302.

[72] A.-R. Sadeghi, C. Stüble, and N. Pohlmann, *European multilateral secure computing base — open trusted computing for you and me*, White paper, 2004, www.emscb.de.

[73] R. Sandhu and X. Zhang, *Peer-to-peer access control architecture using trusted computing technology*, Proceedings of the 10th ACM symposium on Access control models and technologies (SACMAT 2005) (E. Ferrari and G.-J. Ahn, eds.), ACM, New York, NY, USA, 2005, pp. 147–158.

[74] S. E. Schechter, R. A. Greenstadt, and M. D. Smith, *Trusted computing, peer-to-peer distribution, and the economics of pirated entertainment*, Proceedings of The Second Annual Workshop on Economics and Information Security, 2003, College Park, Maryland, May 29–30, 2003.

[75] B. Smyth, M. Ryan, and L. Chen, *Direct anonymous attestation (DAA): Ensuring privacy with corrupt administrators*, Security and Privacy in Ad-hoc and Sensor Networks: 4th European Workshop, ESAS 2007, Cambridge, UK, July 2–3, 2007, Proceedings (F. Stajano, C. Meadows, S. Capkun, and T. Moore, eds.), Lecture Notes in Computer Science, vol. 4572, Springer-Verlag, Berlin, 2007, pp. 218–231.

[76] Software Defined Radio Forum (SDRF), *Overview and definition of software download for RF reconfiguration*, August 2002.

[77] 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*, Trusted Information: The New Decade Challenge, IFIP TC11 Sixteenth Annual Working Conference on Information Security (IFIP/Sec'01), June 11-13, 2001, Paris, France (M. Dupuy and P. Paradinas, eds.), IFIP Conference Proceedings, vol. 193, Kluwer Academic Publishers, Boston, 2001, pp. 403–419.

[78] E. Sparks, *A security assessment of trusted platform modules*, Tech. Report TR-2007-597, Department of Computer Science, Dartmouth College, June 2007.

[79] E. Suh, D. Clarke, B. Gassend, M. van Dyke, and S. Devadas, *The AEGIS processor architecture for tamper-evident and tamper-resistant processing*, 17th Annual ACM International Conference on Supercomputing (ICS '03) (San Francisco, California, USA), ACM Press, New York, USA, 23–26 June 2003, pp. 160–171.

[80] Trusted Computing Group, Mobile Phone Working Group, Portland, Oregon, USA, *Use case scenarios*, September 2005, Version 2.7.

[81] Trusted Computing Group (TCG), Portland, Oregon, USA, *TCG PC client specific implementation specification for conventional BIOS*, July 2005, Version 1.2 Final.

[82] Trusted Computing Group (TCG), Beaverton, Oregon, USA, *Mobile trusted module specification overview document*, 2006.

[83] Trusted Computing Group (TCG), Portland, Oregon, USA, *TCG EFI platform — for TPM family 1.1 or 1.2*, June 2006, Version 1.2 Final.

[84] Trusted Computing Group (TCG), Portland, Oregon, USA, *The TCG mobile trusted module specification*, September 2006.

[85] Trusted Computing Group (TCG), Portland, Oregon, USA, *TPM main, Part 1: Design principles*, March 2006, Version 1.2 Revision 94.

[86] Trusted Computing Group (TCG), Portland, Oregon, USA, *TPM main, Part 2: TPM data structures*, March 2006, Version 1.2 Revision 94.

[87] Trusted Computing Group (TCG), Portland, Oregon, USA, *TPM Main, Part 3: Commands*, March 2006, Version 1.2 Revision 94.

[88] Trusted Computing Group (TCG), Portland, Oregon, USA, *TCG credential profiles*, May 2007, Version 1.1 Revision 1.014 For TPM Family 1,2; Level 2.

[89] Trusted Computing Group (TCG), Portland, Oregon, USA, *TNC architecture for interoperability*, September 2007.

[90] J. Tygar and B. Yee, *Dyad: A system for using physically secure coprocessors*, Technical Report CMU-CS-91-140R, Carnigie Mellon University, Pittsburgh, Pennsylvania, USA, May 1991.

[91] X. Wang, Y. L. Yin, and X. Yu, *Finding collisions in the full SHA-1*, Advances in Cryptology — CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14–18, 2005, Proceedings (V. Shoup, ed.), Lecture Notes in Computer Science, vol. 3621, Springer, 2005, pp. 17–36.

[92] U. G. Wilhelm, S. Staamann, and L. Butty, *Introducing trusted third parties to the mobile agent paradigm*, Secure Internet Programming: Security Issues for Mobile and Distributed Objects (J. Vitek and C. Jensen, eds.), Lecture Notes In Computer Science, vol. 1603, Springer-Verlag, Berlin, 1999, pp. 469–489.

[93] Z. Yan and Z. Cofta, *A method for trust sustainability among trusted computing platforms*, Trust and Privacy in Digital Business: First International Conference, TrustBus 2004, Zaragoza, Spain, August 30 – September 1, 2004, Proceedings (S. Katsikas, J. Lopez, and G. Pernul, eds.), Lecture Notes in Computer Science, vol. 3184, Springer-Verlag, Berlin, 2004, pp. 11–19.

[94] P.-W. Yau and A. Tomlinson, *Using trusted computing in commercial grids*, Proceedings of the 15th International Workshop on Conceptual Structures (ICCS 2007), Sheffield, UK, July 22–27, 2007 (B. Akhgar, ed.), Springer-Verlag, 2007, pp. 31–36.

8 Biographical sketches

Eimear Gallery received her M.Sc. (2002) and Ph.D. (2006) degrees in Information Security from Royal Holloway, University of London. While completing her Ph.D. at Royal Holloway she worked as a consultant in trusted computing for Vodafone, participating in the TCG MPWG on their behalf. Since joining Royal Holloway, she has launched a course in trusted computing as part of the M.Sc. in Information Security. She is currently working as a post-doctoral researcher in the Open Trusted Computing (OpenTC) collaborative project.

Chris Mitchell received his B.Sc. (1975) and Ph.D. (1979) degrees in Mathematics from Westfield College, University of London. He has been a Professor of Computer Science at Royal Holloway, University of London since 1990. He previously worked at Racal Comsec (1979-85) and Hewlett-Packard Laboratories (1985-90), both in the UK. His main research interests are in the applications of cryptography and discrete mathematics.