Trusted Mobile Platforms

E. Gallery and C. J. Mitchell*

Royal Holloway, University of London, Egham,
Surrey, TW20 0EX, United Kingdom.
{e.m.gallery,c.mitchell}@rhul.ac.uk

Abstract. This article addresses two main topics. Firstly, we review the operation of trusted computing technology, which now appears likely to be implemented in future mobile devices (including mobile phones, PDAs, etc.). Secondly, we consider the possible applications of this technology in mobile devices, and how these applications can be supported using trusted computing technology. We focus in particular on three mobile applications, namely OMA DRM, SIMLock, and software download.

1 Introduction

Trusted Computing (TC) technology, which is already present in many recently manufactured PCs, has the potential to revolutionise many aspects of the secure management of IT, particularly in a corporate environment. In recent years, attention has been directed at how this technology might be deployed more broadly, including in a mobile and ubiquitous computing environment.

In this article we aim to do two main things. Firstly, we review the operation of trusted computing technology; we not only describe the main functional components of this technology, but also summarise the main motivations for its introduction. Secondly, we consider possible applications of this technology in mobile devices, since it appears likely that the technology will be implemented in a wide range of future such devices (including mobile phones, PDAs, etc.). In particular we consider how three possible applications, i.e. Open Mobile Alliance Digital Rights Management (OMA DRM), SIMLock, and software download, can be supported using trusted computing technology.

The remainder of the article is divided into two parts, as follows. The first main part commences in section 2, where we describe what trusted computing is intended to achieve and why it has been developed. This is followed in section 3 by a brief history of the development of trusted computing technology. Section 4 summarises the main technical concepts underlying trusted computing. This leads naturally to an overview of the trusted platform subsystem in section 5, followed in section 6 by a more detailed description of trusted computing functionality. The second main part of the article, which is concerned with the application of trusted computing technology to mobile devices, starts

* The development of this article was sponsored by the Open Trusted Computing project of the European Commission Framework 6 Programme.

with a review of the trusted mobile platform in section 7. This is followed by an analysis of three mobile use cases of trusted computing, namely OMA DRM v2, SIMLock and software download, given in sections 8, 9 and 10, respectively. The paper concludes in section 11.

2 Computer Security and Trusted Computing

2.1 Trust

The word *trust* means many different things to different people and in different contexts. Like the word *security*, it has become so over-used that it is almost meaningless unless a definition is provided. This is certainly the case for the term 'trusted computing', and so one thing that we try to do here is define what trust means in this particular context. This theme is returned to in many subsequent parts of this article.

So what does trust mean for our purposes? Well, perhaps the simplest definition would be that *trusted computing* refers to a computer system for which an entity has some level of assurance that (part or all) of the computer system is behaving as expected (or, to quote [1], a platform is trusted if it 'behaves in an expected manner for an intended purpose'). The entity may be various things, including the human user of the PC or a program running on a remote machine. The degree of coverage of this assurance, i.e. whether it covers all aspects of the system or just some part, and the nature of the entity to which assurance is provided, vary depending on the system and the environment within which it is used. Of course, just because the behaviour of a system is as expected, does not necessarily imply that a trusted platform is a secure platform. For example, if an entity can determine that a platform is infected with a virus, whose effects are known, the platform can be trusted by that entity to behave in an expected but malicious manner [2].

It has also been said that 'trusted platforms were so-called because they provide a technological implementation and interpretation of the factors that permit us, in everyday life, to trust others' [3], i.e.

- either first hand experience of consistent behaviour, or trust in someone who vouches for consistent behaviour;
- unambiguous identification; and
- unhindered operation.

In order to implement a platform of this nature, a trusted component, which is usually in the form of built-in hardware, is integrated into a computing platform [4]. This trusted component is then used to create a foundation of trust for software processes running on the platform [4]. Bodies such as the Trusted Computing Group (TCG) (discussed in section 3) standardise specific functionality to be incorporated into end systems which are known as 'trusted platforms'. Depending on how the specified functionality is implemented, such a platform is then able to provide a degree of assurance about some aspect of its operation.

2.2 Computer Security

Computer security is a long-established subject, with a considerable literature dating back to the 1960s. There are many books on the subject of secure computing (see, for example, Gollmann [5] and Pfleeger [6]). Trusted computing, with the meaning applied here, is a much more recent phenomenon, and is essentially one specialised part of the larger subject of computer security. One reason that the notion has emerged is because of the changing nature of computer systems, and their increasing ubiquity.

Historically, computer security has provided a theory to understand and reason about fundamentally important security functionality within operating systems. This functionality covers issues such as access control to resources within the context of multi-user operation. Most of computer security is thus concerned with security aspects of software, and pays relatively little attention to hardware security issues.

The reason for this is clear. Until the advent of the PC, computers were relatively large and expensive devices, typically with a number of users. The hardware was often kept in a physically secure location, to which access was only granted to authorised staff. Hence the main security issue was to design the file system software such that one user could not access data and resources to which he or she was not entitled, including other users' data. The security of the hardware was not something directly addressed — it was essentially a prerequisite for the correct operation of the software.

This resulted in a large and well-developed theory of computer security, covering such topics as multi-level system security and a host of models for access control systems. This theory remains very important; however despite overlapping terminology, this is not the main subject of this article. Terminological confusion is a particular problem, not just because of the overuse of the word *trust*, but because the term *Trusted Computing Base (TCB)* has become widely used to mean something somewhat different to the recent use of *trusted computing*.

As defined in the Orange Book [7] (see also Gollmann [5]) the TCB is the totality of protection mechanisms within a computer system, including hardware, firmware and software. Whilst this is by no means unrelated to trusted computing, as discussed here, the meaning is definitely not the same.

Of course, it is true that physically secure subsystems have always had a place in the spectrum of secure computing, but they have mainly been used in specialist applications. For example, many secure subsystems have been designed and used in applications requiring physical security for stored keying material — examples of such systems include the IBM 4758 [8] (see also Chapter 15 of [9] for a discussion of interfaces to such subsystems).

2.3 Computer Security and PCs

The traditional assumptions regarding the physical security of important computer systems are clearly completely inappropriate for the vast majority of PCs

in use today. Most such PCs have only a single user, and no physical security is provided for the PC hardware. Short term access to the PC can easily result in unauthorised copying of information and/or modifications to software and data; this is easy to achieve regardless of what software protections exist, simply by temporarily removing a hard disk and attaching it to a different system. That is, regardless of how 'secure' the operating system is in the traditional sense, the lack of guarantees about physical security means that the correctness of software or information stored on the PC cannot be trusted; neither can the confidentiality of any such information. The situation is made worse by the fact that modern PC operating systems and application software are enormously complex, and removing all software vulnerabilities is an almost impossible task. Hence for a combination of reasons today's systems are very vulnerable to a range of attacks.

Trusted computing as we mean it here is an idea which has arisen from the need to address these problems. Trusted computing refers to the addition of hardware functionality to a computer system to enable entities with which the computer interacts to have some level of trust in what the system is doing. Pearson [10] defines a related notion, namely that of a *trusted platform*, as follows.

> A trusted platform is a computing platform that has a trusted component, probably in the form of built-in hardware, which it uses to create a foundation of trust for software processes.

The exact nature of a trusted platform is an issue that is explored below. An interesting discussion of trust and trusted computing can be found in [11]. A useful high level introduction to trusted computing has also been given by Felten [12].

2.4 Goals of Trusted Computing

The main goals of trusted computing are to add some (modest) set of hardware enhancements to a computer system to enable (a) the state of the system to be checked, both locally and remotely, and (b) data to be protected so that it will be available only when the system is in a specified state. Achieving these apparently limited goals requires adding a significant amount of functionality to a PC, although most of the necessary functionality can be incorporated into a special-purpose chip, the *Trusted Platform Module (TPM)*.

Achieving the first of these goals is perhaps the most fundamental, and much of the functionality of the TPM is necessary simply to meet this goal. For example, it is clear that, unless there is some way of providing assurance about the correct functioning of the operating system, then there is no way of providing any assurance about the correct operation of applications. This means that it is necessary to monitor the process of booting the operating system, in such a way that the integrity of the system after the boot process has been completed can be verified. This is by no means a simple requirement, since booting a platform such as a PC is a highly complex process.

Essentially, it requires the first piece of software that executes to be fixed (unchangeable), and also for each piece of software that runs subsequently to

be checked (measured) by its predecessor. This process of measurement involves applying a cryptographic hash function to the software, storing the result, and later reporting the result in a reliable way. As we will see below, this means that the 'roots of trust' for measurement, storage and reporting are essential to meet the first goal.

This notion of monitoring the booting of an operating system also highlights the limitations of hardware-based trusted computing. That is, since modern operating systems are large and complex, it is clear that determining whether a measured version of the operational state of a PC can be trusted or not is essentially a hopeless task. This is because there will be a very large number of different possible 'valid' states for such a system, each of which will generate a measurement. As a result, trying to decide whether a measurement represents a valid or invalid software state becomes infeasible.

Thus, whilst the TPM can measure the initial stages of the booting of a PC, this process cannot be extended indefinitely to the entire system. As a result, at some point the software must be trusted to 'look after itself'. That is, at least for complex multi-purpose systems such as PCs, the use of hardware-based measurements of software state must be combined with some other means of providing ongoing protection for a system.

This is achieved by introducing the notion of a isolation layer (discussed in more detail below). That is, the trusted computing hardware can be used to provide assurance that a particular isolation layer has been booted; after that the isolation layer must itself guarantee the integrity of the system. An isolation layer will typically be booted immediately prior to starting up one or more 'guest' operating systems. The isolation layer must be trusted to provide ongoing security for the operating systems which it hosts. In particular, it must be trusted to isolate the different operating systems (and applications running on operating systems) so that data cannot pass between them in unauthorised ways, and so that even if malicious code is introduced into one environment it cannot damage other environments.

Of course, this analysis does not necessarily apply to all trusted platforms. For single use, simple platforms, e.g. as might be the case for embedded or mobile systems, ongoing hardware-based measurement and verification of the complete software environment may be possible, because the number of possible valid states may not be very large.

Finally, note that the functionality provided by the TPM can be used for a host of purposes, many quite distinct from the fundamental goals discussed above. Indeed, the addition of a TPM to a computer system is somewhat akin to equipping every PC with a physically secure 'hardware security module'. The presence of such hardware is likely to give rise to a host of new applications, which we cannot begin to envisage today.

3 A Brief History of Trusted Computing

The concept of trusted computing, as described throughout this article, was initially defined by the Trusted Computing Platform Alliance (TCPA). The TCPA, an industry working group which focussed on the development and standardisation of trusted computing technology, was formed in January 1999 by Compaq, HP, IBM, Intel and Microsoft. Some of the earliest papers introducing this paradigm were published by HP in 2000 [13, 14]. In early 2001, following the expansion of the group, the TCPA published the first specification for a TPM, a fundamental component of a trusted platform. Following this, a PC-specific specification, detailing the additional changes required in order to produce a TCPA-compliant trusted PC, was published. The TCPA TPM and PC specifications are described in [4].

In April 2003 the TCPA was superseded by the TCG. The TCG have continued to develop and expand the TPM specifications, the current version of which is v1.2 [15–17]. The TPM specifications are supported by a standard set of TPM APIs which provide an abstraction of the TPM to software developers/vendors [18]. The TCG has defined how a TPM may be utilised on a variety of platform types such as a PC client, server, hard copy device, and storage device. A trusted mobile platform is also being specified (see also section 7). In conjunction with this, work is ongoing on specifications designed to aid the seamless adoption, integration and inter-operability of trusted computing platforms.

Microsoft's proposals for a trusted computing architecture were initially released under the name Palladium, and subsequently under the title Next Generation Secure Computing Base (NGSCB). The fundamental component of the most recently described version of the Microsoft architecture is an isolation layer designed to support the execution of isolated runtime environments for sensitive applications. This architecture assumes the presence of TPM functionality, as defined by the TCG, in conjunction with processor enhancements and chipset extensions which enable the implementation of the high-assurance isolation layer. For further information see [19–22].

The Terra system architecture [23], the Perseus framework [24, 25], the Open Trusted Computing architecture [26] and the European Multilaterally Secure Computing Base (EMSCB) [27] have some similarities to the current version of NGSCB. At the heart of each architecture is an isolation layer, which has been designed to support the isolated 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 have been designed to use either a virtual machine monitor such as XEN [28] or a microkernel in order to provide isolated execution environments. EMSCB incorporates an L4 microkernel-based isolation layer. Each architecture also assumes a hardware platform which includes a TPM. The presence of chipset and processor enhancements are also acknowledged within all architectures as pivotal in order to ensure a high-assurance isolation layer implementation.

Hardware manufacturers such as Intel and AMD have specified the required processor enhancements and chipset extensions under the names of LaGrande [29] and Presidio respectively.

As this technology has evolved and matured, a growing body of work has emerged on the potential applications of trusted computing. Both Balacheff et al. [30] and Spalka, Cremers and Langweg [31] describe how trusted computing functionality may be utilised in order to enhance the security of the digital signature process. Schechter et al. [32], Kinateder and Pearson [33] and Balfe, Lakhani and Paterson [34] discuss the application of trusted computing to peer-to-peer networks. The deployment of trusted computing functionality has also been proposed in order to enable secure software download [35], support secure single sign-on solutions [36], improve the security and privacy of a biometric user authentication process [37] and to facilitate identity management [38, 39]. A number of authors have also considered trusting computing's applicability to the agent paradigm [40–43] and online gaming [44]. Further application scenarios are described in [4, 23, 26, 45].

While the benefits of trusted computing functionality have become apparent, this new technology has also been criticised. Anderson [46] expresses the view that trusted computing may be used to support censorship, stifle competition between software vendors, and hinder the deployment and use of open source software. The issues of software 'lock in' and interoperability, the contentious issue of TC-enabled DRM, and, more generally, remote control of the software on a platform, are also highlighted by members of the Electronic Frontier Foundation, namely Scheon [47] and von Lohmann [48]. Privacy concerns relating to trusted platforms have also been widely discussed [49], and will be revisited in section 6. A high level account of these criticisms is provided by Arbaugh [50].

In parallel to the development of the trusted computing technologies described above, closely related concepts such as secure boot have been widely discussed, and a number of alternative architectures have been developed with the goal of providing more secure and trustworthy computing platforms. The concept of a secure boot has been repeatedly discussed in the literature, most notably by Tygar and Yee [51], Clark [52], Arbaugh, Farber and Smith [53] and Itoi et al. [54]. While the eXecute-Only Memory (XOM) architecture [55, 56] and the architecture for tamper evident and tamper resistant processing (AEGIS) [57] are not strictly examples of trusted computing platforms, like trusted computing they provide strong process isolation through the development of hardened processors.

4 Trusted Computing Concepts

Trusted computing, as defined by the TCG, is synonymous with four fundamental concepts: integrity measurement, authenticated boot, sealing and platform attestation. A platform incorporating these concepts constitutes what we refer to here as a *Trusted Platform (TP)*. Note that, since the original description was

published, the definition of what constitutes trusted computing functionality has been revised and extended to incorporate the concept of software isolation.

4.1 Integrity Measurement

An integrity measurement is defined in [22] as the cryptographic digest or hash of a platform component (i.e. a piece of software executing on the platform). For example, the integrity measurement of a program could be calculated by computing the cryptographic digest or hash of its 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' [4]. Integrity metrics indicate the history of the platform.

4.2 Authenticated Boot

An authenticated boot is the process by which a platform's configuration or state is reliably captured and stored. During this process, the integrity of a predefined set of platform components is measured, as defined in section 4.1. These measurements are condensed to form a set of integrity metrics which are then stored in a tamper-resistant log. A record of the platform components which have been measured is also stored on the platform. Condensing enables an unbounded number of platform component measurements to be stored. If each measurement was stored separately, an unbounded amount of memory would be required to store them [4].

4.3 Sealing

Sealing is the process by which sensitive data can be associated with a set of integrity metrics representing a particular platform configuration, and encrypted. The protected data can only be decrypted and released for use when the current state of platform matches the integrity metrics sealed with the data.

4.4 Attestation

Attestation is the process by which a platform can reliably report evidence of its identity and its current state (i.e. the integrity metrics which have been stored to the tamper resistant log, and the record of the platform components which have been measured, as described in section 4.2).

4.5 Software Isolation

Isolation enables the unhindered execution of software through the provision of assured memory space separation between processes [58].

5 The Trusted Platform Subsystem

As stated in section 2.1, in order to provide the services described in sections 4.1 to 4.4, a 'trusted component' must be integrated into a computing platform. This 'trusted component' is made up of three roots of trust — the Root of Trust for Measurement (RTM), the Root of Trust for Storage (RTS) and the Root of Trust for Reporting (RTR). In order to provide software isolation, as described in section 4.5, an isolation layer can be deployed on the platform. In conjunction with this, the platform may also incorporate processor enhancements and chipset extensions which have been designed to enable a secure and high assurance isolation layer implementation.

5.1 The RTM

The RTM is a computing engine capable of measuring at least one platform component, and hence providing an integrity measurement, as described in section 4.1. 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 [59]. It is also preferable that the CRTM be physically tamper-evident [4].

5.2 The RTS and RTR

The RTS is a collection of capabilities which must be trusted if storage of data inside a platform is to be trusted [4]. 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, as described in section 4.2. The RTS also provides integrity and confidentiality protection to data and enables sealing.

In conjunction with the RTM and RTS, an additional root of trust is necessary for the implementation of platform attestation, namely the RTR. The RTR is a collection of capabilities that must be trusted if reports of integrity metrics are to be trusted (platform attestation) [4].

The RTR and the RTS constitute the minimum functionality that should be provided by a TPM [15–17]. 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 such as: input/output; non-volatile and volatile memory; a minimum of 16 *Platform Configuration Registers (PCRs)*, which are used by the RTS to store the platform's integrity metrics; a random number generator; a HMAC engine; a SHA-1 engine; key generation capabilities; an asymmetric encryption and digital signature

engine; and an execution engine, as shown in figure 1. The TPM must be protected completely against software attack, i.e. the RTS and RTR (i.e. 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 (tamper-evidence) [4].

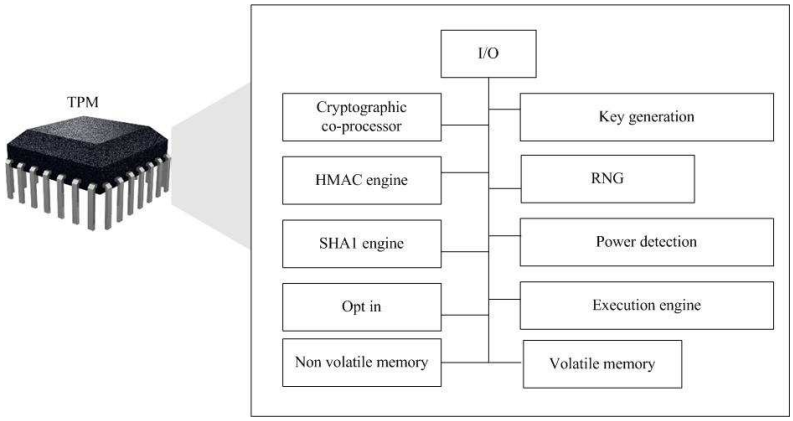

Fig. 1. The TPM chip

5.3 Software Isolation Technology

A number of approaches have been proposed in order to facilitate software isolation. Many of these approaches, however, have associated difficulties with respect to assurance, device support, legacy OS compatibility and performance.

OS-Hosted VMM: In the case of an OS-hosted virtual machine monitor, such as VMWare workstation, all guest OSs executing in VMs utilise the host OS device drivers. While this implies that the every guest can utilise drivers developed for the host machine, it also means that the isolation layer essentially incorporates the VMM and the host OS, making assurance problematic [22, 60].

Standalone VMM: In a standalone virtual machine monitor, such as Terra [23], 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 utilised 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 [22].

Para-Virtualisation: Isolation layers using para-virtualisation techniques, such as XEN [28], 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 [60]. In the first case, an I/O-type-specific API for each device is integrated into the VMM, in conjunction with the device drivers [60]. 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, may execute in parallel to guest OSs, which are modified to incorporate para-virtualised drivers [60]. 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.

An Isolation Layer with Hardware Support: The isolation layer described as part of the NGSCB [21, 22] 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 [29]. 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 [22]. The original hardware interface is exposed to one guest OS [22]. 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 indicates, given the state of the system, if a particular subject (DMA device) has access (read or write) to a specified resource (physical address), [22]. The DMA policy is then read and enforced by hardware, for example the memory controller or bus bridges.

Hardware extensions required in order to facilitate the implementation of the NGSCB isolation layer have been provided as part of Intel's LaGrande [29] and

AMD's Presidio initiatives. Both enable the efficient and secure implementation of an isolation layer, as described by Microsoft, through the implementation of 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.

6 The Trusted Platform Subsystem Functionality

6.1 The Authenticated Boot Process

An authenticated boot process enables the state of a platform to be measured and recorded. In order to describe an authenticated boot process we 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 might proceed as follows, where we assume that the CRTM is part of the BBB. The CRTM 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 Operating System (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 to the SML. This process is illustrated in figure 2.

This process of measuring, condensing, storing, and handing-off, continues until the platform's configuration 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 [59] and for an Extensible Firmware Interface (EFI) platform [61].

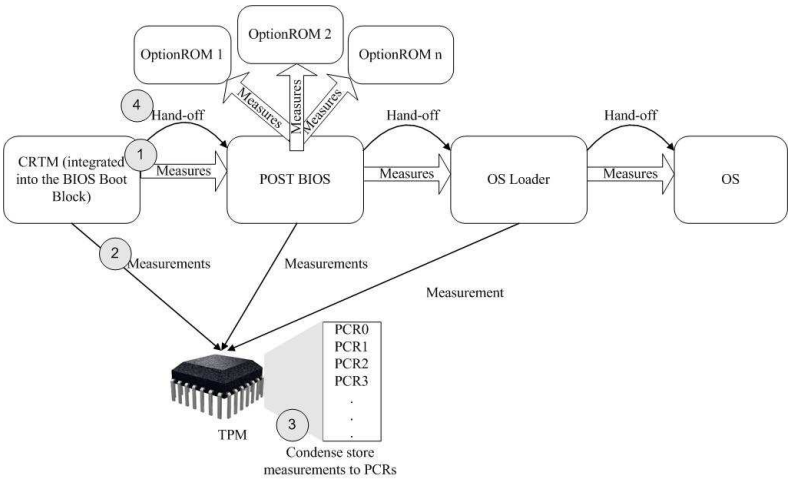

Fig. 2. The authenticated boot process

6.2 TPM Protected Storage

The TPM provides secure ('protected') storage functionality, which incorporates a sealing capability. This functionality was designed so that an unbounded number of secrets/data could be confidentiality and integrity protected on a TP.

Each TPM contains a 2048-bit asymmetric key pair known as a *Storage Root Key (SRK)*. 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, indeed, a symmetric key), which has been encrypted using a key at a higher layer of the hierarchy. A simplified TPM protected object hierarchy is illustrated in figure 3.

Asymmetric encryption is used to confidentiality-protect key and data objects. Protected 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.

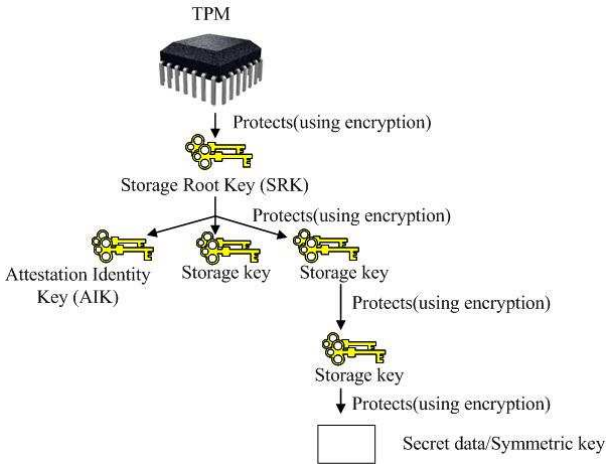

Fig. 3. The TPM protected object hierarchy

The TPM protected storage functionality incorporates an asymmetric key generation capability. This capability enables the generation of key pairs for which the 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 from the pairs 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 ability to associate data with a particular platform configuration, and then encrypt the data so that it is bound to this configuration. The configuration is recorded as a pair of sets of integrity metrics, which represent the state of the platform when the data was sealed (*digest at creation*), and the state of the platform required for the data to be unsealed (*digest at release*). The sealed data can only be decrypted by the TPM on which it was encrypted, and will only be released by the TPM of the host platform is in the state specified in the *digest at release*. Once the data has been released the *digest at creation* must be checked in order to ensure that the data was not sealed by rogue software.

6.3 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. However, to achieve this, it

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.

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. signed 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, when considered together, a particular type of TPM, associated components such as a CRTM, the connection of a CRTM to a motherboard, and 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 (see figure 3) can be generated by a TPM and associated with a 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. A highlevel description of a TPM endorsement credential, a platform credential, an AIK credential and their relationship is shown in figure 4.

The AIK private key is then used by the TPM during platform attestation. Note that the fact that a platform can generate arbitrary numbers of AIKs (and

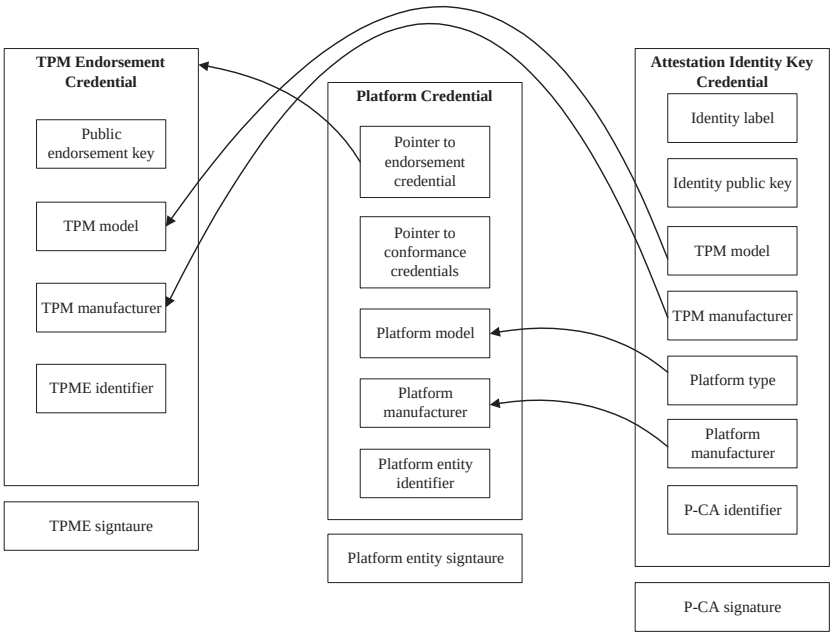

Fig. 4. TP credentials [62]

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.

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 this nonce together with 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 ((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;
- 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.

Anonymity Issues: The above approach has attracted a certain amount of criticism, since it puts the P-CA in a powerful position. That is, because the P-CA generates the AIK credentials, and it also sees all the platform credentials when it does so, a P-CA can link together all the pseudonyms of a particular platform, and hence breach anonymity for that platform.

As a result, v1.2 of the TCG specifications incorporates a new system that allows for trusted platform anonymity/pseudonymity without replying on a third party to keep links between pseudonyms secret. That is, it incorporates a means for an entity to obtain AIK credentials without revealing its 'identity' to the third party generating these credentials. This new technique, known as *Direct Anonymous Attestation* (DAA), is due to Brickell, Camenisch and Chen [63–65].

DAA essentially divided the process of obtaining AIK credentials into two phases. In the first phase, a platform obtains a DAA certificate from a third party, to which it shows all its credentials. This DAA certificate can then be used to obtain AIK credentials, in such a way that the issuer of the credentials does not actually get to see the DAA certificate, but just receives evidence that the platform possesses such an object. In this way, even the entity which generates the DAA certificate cannot link together two or more AIK public keys belonging to the same platform.

Whilst we do not describe the process here (it is, in fact, highly complex) it is important to note that it also possesses a number of other interesting properties. For example, the scheme can be used in such a way that the degree of unlinkability is 'tunable'; this allows the possibility of blacklisting (revoking) credentials for platforms which have been compromised.

6.4 Isolated Execution Environments

An isolated execution environment, independent of how it is implemented, should provide the following services to hosted software [22]:

- 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.

7 Trusted Mobile Platforms

7.1 The Development of Trusted Mobile Platforms

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 would appear 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 functionality that must be provided by such a device is in the process of being standardised. This is the role of the TCG Mobile Phone Working Group (MPWG), discussed immediately below.

7.2 The MPWG Activity

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 MPWG, the main challenge for which is to determine the 'roots of trust', see section 5, required within a trusted mobile phone. In order to identify the capabilities required of a trusted mobile phone, a number of use cases, whose secure implementation may be aided by the application of trusted platform functionality, have been identified by the MPWG. Among these use cases are SIMLock, device authentication, mobile ticketing, mobile payment and robust DRM implementation [66]. As stated by the MPWG [66], 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.

The MPWG has recently published the TCG Mobile Trusted Module (MTM) Specification [67]. 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) [68]. 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 [68]. The use cases described by the TCG in [66] have been analysed, along the lines of the analyses given in [69], in order to determine the subset of functionality required within a MTM to enable their secure implementation.

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 [68].

7.3 Applications

The applications for trusted mobile phones discussed in the current TCG MPWG use case document cover:

- the protection of downloaded content and software;
- the protection of user data and identity information, and device identity information; and
- enabling mobile payment and mobile ticketing.

In this article we focus on three specific use-cases, namely OMA DRM and software download, which involve the protection of downloaded data, and SIM-Lock, which requires the protection of device identity information. These use cases have been chosen because of their commercial and scientific interest.

In the following sections these three use cases are presented. Also given is an analysis of the trusted computing functionality required of a mobile platform in order to support a secure and robust implementation of each use case.

8 A Robust Implementation of OMA DRM v2

8.1 Use Case Description

DRM: Current 3G systems are already capable of delivering a wide range of digital content to subscribers' mobile telephones, including music, video clips, ring tones, screen savers or java games. As network access becomes ever more ubiquitous and media objects become more easily accessible, providers are exposed to increased risks of illegal consumption and use of their content. DRM facilitates the safe distribution of various forms of digital content in a wide range of computing environments, and gives assurance to the content providers that their media objects cannot be illegally accessed.

A Digital Rights Management system is an umbrella term for mechanisms used to manage the life cycle of digital content of any sort. A DRM agent, i.e. the DRM functionality of a device responsible for enforcing permissions and constraints associated with protected content, must be trusted with respect to its correct behaviour and secure implementation [70]. Stipulation of a trust model, within which robustness rules are defined, is one method of specifying how secure

a device implementation of a DRM agent must be, and what actions should be taken against a manufacturer that builds devices that are insufficiently robust [71].

The OMA: The OMA was founded in June 2002. One of the original objectives of the OMA was to define a DRM specification set for use in a mobile environment. OMA DRM v1 was published as a candidate specification in October 2002, and was approved as an OMA enabler specification in 2004 [72], after full interoperability testing had been completed.

Following this, OMA DRM v2 was published as a candidate specification in July 2004 [73]. OMA DRM v2 builds upon the version 1 specifications to provide higher security and a more extensive feature set [71]. Devices other than mobile phones are also supported by OMA DRM v2. The OMA DRM version 2 specification set defines [70]:

- the format and the protection mechanism for protected content;
- the format and the protection mechanism for rights objects;
- the security model for the management of encryption keys; and
- how protected content and rights objects may be transferred to devices using a range of transport mechanisms.

OMA DRM Functional Architecture: The model under consideration is taken from [70] and is summarised in figure 5. A user requests a media object from a content issuer. The requested content, which is packaged in order to prevent unauthorised access, is then sent to the user's device. The packaging of the content may either be completed by the content issuer or by the content owner, before it is dispatched to the content issuer. The rights object associated with the requested media object is delivered to the user by the rights issuer. In practice, this rights issuer may be the same entity as the content issuer.

OMA DRM v1: Version 1 of the OMA specifications [74, 75] represents the OMA's initial attempt to define a DRM solution for a mobile environment. Three main goals were specified for OMA DRM v1 [71]. The solution was required to be timely and inexpensive to deploy. It was also required to be easy to implement on mass market mobile devices. Finally, it was required that the initial OMA DRM solution did not necessitate the roll-out of a costly infrastructure. In the development of OMA DRM v1 a trade-off was made, so that the objectives listed above could be met at the expense of certain security requirements.

Three classes of DRM functionality are specified in OMA DRM v1 [74, 75]. The first class of DRM functionality, forward lock, must be supported by an OMA DRM v1 agent on a device. Provision of combined delivery and separate delivery, the second and third classes of DRM functionality, is optional.

1. *Forward lock* prevents unencrypted content being forwarded from the device to which it was initially delivered. The protected content is wrapped inside a

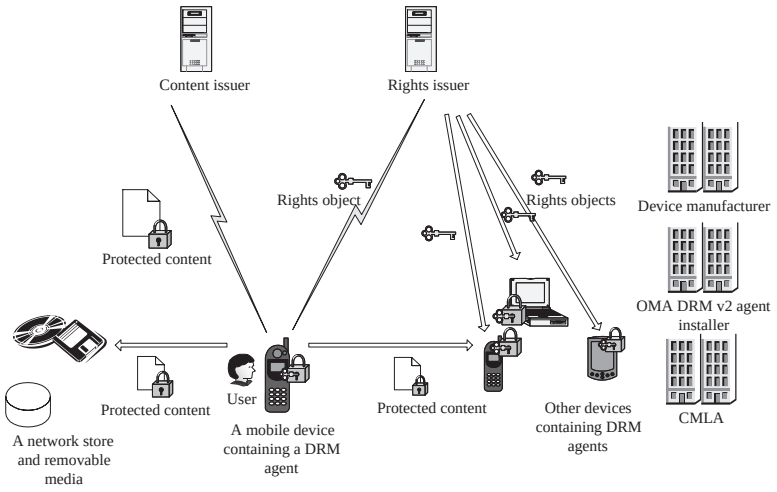

Fig. 5. OMA DRM system model

DRM message, which indicates to the OMA DRM v1 agent on the receiving device that the content is not to be forwarded. Protection is dependent on the OMA DRM v1 agent acting accordingly.

2. *Combined delivery* involves sending unencrypted content and its associated rights object together within a DRM message.
3. *Separate delivery* involves sending encrypted content and the associated rights object separately. The content is encrypted and sent in a format known as the DRM Container Format (DCF). Headers, which allow a receiving device to associate the correct rights object with the corresponding DCF object, are also contained in the transmitted file. The associated rights object, which contains the relevant permissions and constraints, and the decryption key for the associated content, is delivered via SMS.

OMA DRM v2: OMA DRM v2 [70, 76] builds upon the OMA DRM v1 specifications, with the primary objective of providing a more secure DRM solution. The following security vulnerabilities have been identified in OMA DRM v1 [71].

1. A rights issuer has no way of determining whether the requesting device supports DRM. When using the forward lock and combined delivery features, where the content is not encrypted, this particular security vulnerability enables an attack in which unencrypted content is initially sent to a PC made to look like a compliant phone. On receipt, content is then extracted and illegally distributed.

2. In the separate delivery DRM class, where the content is encrypted, the content encrypting key is not protected. This implies that the attack described above in step 1 is also possible in this case, although it is more complex and more difficult to complete successfully [71].
3. The device has no way of authenticating the rights issuer, and therefore may be sent bogus rights objects from an entity claiming to be the legitimate rights issuer.

OMA DRM v2 addresses the above security weaknesses by using additional security mechanisms.

- Both device authentication and rights issuer authentication are provided.
- Mechanisms are deployed in order to protect the confidentiality of media objects. Content is protected using a Content Encrypting Key (CEK). This CEK is sent in a rights object encrypted using a Rights Object Encrypting Key (REK). The REK itself is sent encrypted using the public key of the device or a pre-established domain key.
- Mechanisms are also deployed so that the OMA DRM v2 agent can determine whether a media object received from a Rights Issuer (RI) has been modified in an unauthorised way.

The OMA DRM v2 specifications are no longer mobile device specific, as was the case with the v1 specifications. It also provides a richer feature set which includes, most notably, support for [71]:

- the automatic preview of protected content;
- subscription services;
- continuous media such as streaming and progressive download of content;
- reward schemes;
- domains, i.e. collections of devices belonging to a single user. A domain can be established by a user. When a device joins a domain, the RI sends a domain key to the device, encrypted using the device's OMA DRM v2 public key. Following this, content and the associated access rights may be shared among the devices in the domain. Rights objects must be explicitly acquired for the domain rather than a specific device. A RI may control the number of devices allowed in a domain, although the user is entitled to add and remove devices at will, as long as the limit set by the RI is adhered to.
- unconnected devices, i.e. devices not network connected. This feature is supported by the implementation of domains. An unconnected device may be added to a domain, after which content and rights may be copied from a connected domain device to the unconnected device.

In order to support the additional security mechanisms, and, indeed, the expanded feature set described above, a dedicated suite of DRM security protocols, the Rights Object Acquisition Protocol (ROAP) suite, was developed by the OMA. In addition to the ROAP suite, it was agreed that the OMA DRM v2 specification set should be supported by a 'trust model'. Such a trust model

enables an RI to obtain assurances about DRM agent behaviour, and the robustness of the DRM agent implementation [70]. It is the responsibility of the Content Management Licensing Administrator for Digital Rights Management (CMLA DRM), or a similar organisation, to provide a trust model, i.e. robustness rules, and to define actions which may be taken against a manufacturer who builds devices which are not sufficiently robust.

The OMA DRM v2 Security-Critical Data: In order to use the ROAP suite, OMA DRM agents must be equipped with the necessary security-critical data.

Every OMA DRM v2 agent is assigned a unique key pair [70]. The private key from this key pair is used by an OMA DRM v2 agent to generate digital signatures, so that a rights issuer can authenticate a particular DRM agent. The public key from this pair is used by rights issuers in order to distribute rights object encryption keys, which are used to protect content encryption keys, that are themselves used to encrypt content.

A certificate, which identifies the DRM agent and binds the agent to the public key described above, is also provided to the DRM agent. The OMA DRM v2 certificate can be specified as part of one or more certificate chains. If so, the OMA DRM v2 certificate comes first in a chain, and each subsequent certificate contains the public key necessary to verify the certificate preceding it [73]. When the rights issuer with whom the OMA DRM v2 agent is communicating, indicates its preferred trust anchor(s), i.e. its trusted root CA(s), the OMA DRM v2 agent must select and send back a device certificate (chain) which points to an appropriate anchor [73], so that the RI can verify the OMA DRM v2 agent certificate.

The *device details* indicate the device manufacturer, model, and version number. Finally, the *trusted RI authorities certificate* is used to indicate which rights issuer trust anchor(s) are recognised by the OMA DRM v2 agent. This trusted RI authorities certificate may either be a single root certificate, as is the case in the CMLA trust model [77] where the trusted RI authorities certificate is a self-signed CMLA root CA certificate, or a collection of self-signed public key certificates representing the preferred trust anchors of the OMA DRM v2 agent.

The ROAP Suite: This dedicated suite of DRM security protocols was developed by the OMA to enhance the security of the DRM process and the functionality of the OMA DRM agent. The ROAP suite consists of five protocols.

The *4-pass registration protocol* is defined by the OMA as a "complete security information exchange and handshake between the RI and a DRM agent in a device" [76]. The protocol enables the negotiation of protocol parameters including protocol version, cryptographic algorithms, certificate preferences, optional exchange of certificates, mutual authentication of the mobile device and RI, integrity protection of protocol messages, and optional device DRM time synchronisation [76]. In this protocol, two messages are sent from the device to the RI, namely the device hello and the registration request, and two messages

are sent from the RI to the device, namely the RI hello and the registration response. There are three occasions when the 4-pass registration protocol can be used [76]:

- on first contact between the RI and a mobile device;
- when security information needs to be updated; and
- when the device time source is deemed to be inaccurate by the RI.

On receipt of the registration request message, and before the registration response message is sent, the RI may optionally perform a nonce-based OCSP request for its own certificate, using the device nonce sent in the registration request message [76]. An OCSP request may also be performed if the RI deems the device DRM time source to be inaccurate, or if the device is an unconnected device which does not support DRM time [76]. The device nonce is used to cryptographically bind an OCSP response to the corresponding OCSP request, to prevent replay attacks [78].

The *2-pass* and *1-pass rights acquisition protocols* allow a device to request and acquire a rights object from a RI.

The *2-pass join domain* and *leave domain protocols* are used to manage domains. Once a domain has been established by a user, and after devices have been added to the established domain, protected content and associated rights objects, which have been explicitly created for domain use, may be copied and moved between domain devices. Therefore, rather than requesting a separate rights object for each individual device, only one domain RO need be requested.

8.2 The Robustness Rules

In order to comply with the definition of a 'robust OMA DRM v2 implementation', as defined by the CMLA [77], a number of requirements must be met, as summarised below.

It is required that "an OMA DRM v2 agent can perform self-checking of the integrity of its component parts so that unauthorised modifications will be expected to result in a failure of the implementation to provide the authorised authentication and/or decryption function" [77].

A robust implementation of OMA DRM v2 must confidentiality-protect the OMA DRM v2 agent private key when loaded into and while stored and used on the device.

The OMA DRM v2 agent private key and OMA DRM v2 security critical data such as the OMA DRM v2 agent certificate (chains), the device details and the trusted RI authorities certificate, must be integrity-protected when loaded into, while stored on, and while in use on the device. While it is not necessary to integrity-protect the OMA DRM v2 agent certificate or the OMA DRM v2 agent certificate (chain), as any unauthorised modification will be detected when the certificate(chain)(s) are verified, the trusted authorities certificate, which is defined in the CMLA trust model as a self-signed CMLA root CA certificate, needs to be integrity-protected.

Domain context information, communicated by a rights issuer to a mobile device during a 2-pass join domain protocol, must also be protected:

- the domain ID, the expiry time of the domain context and the rights issuer's public key must be integrity-protected while stored and used on the device; and
- the domain key, used to protect domain rights objects, must be confidentiality and integrity-protected while in storage and in use on the device.

Rights issuer context information, established during the 4-pass registration protocol, such as protocol parameters, protocol version, RI certificate preferences, agreed RI identification information, RI certificate information and the context expiry time, must also be integrity-protected.

The secret keys used to protect the integrity and confidentiality of rights objects must be confidentiality and integrity-protected while in storage and in use on the device.

All of the elements mentioned above should only be accessible by authorised entities, namely the correctly functioning OMA DRM v2 agent.

A robust OMA DRM v2 implementation must incorporate a DRM time source synchronisation mechanism which is reasonably accurate and resistant to malicious modifications by the end user.

Finally, nonces generated on the OMA DRM v2 device and used in the 4-pass registration protocol, the 2-pass RO acquisition protocol or the join domain protocol must be both non-repeating and unpredictable in order to mitigate the threats of both replay and preplay attacks against the protocol suite.

8.3 A Robust Implementation of OMA DRM using Trusted Computing

In this section, we consider how trusted computing functionality can be used to help meet the requirements in the CMLA client adopter agreement [77] and summarised in section 8.2, thereby enabling a robust implementation of OMA DRM v2.

While TC functionality cannot guarantee the integrity of the OMA DRM v2 agent while it is being stored, TC mechanisms can be used to help detect malicious or accidental modifications or removal. Secure boot functionality can be used to ensure that a set of security-critical platform components boot into a predetermined state. Secure boot is not currently enabled by the TCG TPM main specifications. However, much work on secure boot has been conducted independently of the TCG, including by Tygar and Yee [51], Clark [52], Arbaugh, Farber and Smith [50] and Itoi et al. [54]. Each of these papers describe a similar process, in which the integrity of a pre-defined set of system components is measured, as described in section 4.2, and these measurements are then compared against a set of expected measurements which must be securely stored and accessed by the platform during the boot process. If, at any stage during the boot process, the removal or modification of a platform component, such as the OMA

DRM v2 agent, is detected, the boot process is aborted. While a secure boot process is not specified in the TPM specification set, the TCG mobile phone working group has recently released a specification for a Mobile TPM which enables a secure boot process [67].

Security-critical data associated with the OMA DRM v2 agent, such as the device details and the trusted RI authorities certificate, which require integrity protection while in storage, can also be verified as part of a secure boot process.

Alternatively, sealed storage functionality may be used in order to detect the malicious or accidental modification or removal of the OMA DRM v2 agent while in storage, and, indeed, to store data which needs 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 agent is executing in an isolated execution environment.

The security-critical data and any domain and RI context information to be protected is first associated with a 'digest at creation' and a 'digest at release', and then encrypted by the TPM, as described in section 6.2. While integrity-protection is not explicitly provided by the TPM, 20 bytes of authorisation data can be associated with the data to be sealed prior to its encryption, as described in section 6.2, thereby ensuring that the data is integrity-protected while in storage. The sealed data is asymmetrically encrypted and the corresponding private decryption key is securely stored within the TPM, thereby ensuring that the data is confidentiality-protected while in storage. The inclusion of the 20 bytes of authorisation data and the digest at release with the sealed data prior to encryption ensures that only an authorised entity can access the data, and that access can only take place when the platform is in the required software state. Finally, sealing the data to a specified platform configuration also ensures that any unauthorised modification and/or removal of security-critical software (e.g. the OMA DRM v2 agent) reflected in the digest at release will be detected, and access to the sealed data denied.

Rather than using the TPM merely to confidentiality and integrity-protect the OMA DRM v2 private key, the TPM can also be used to generate the required OMA DRM v2 agent asymmetric key pair as well as to protect the private key while in storage and in use on the device.

TC functionality also enables the isolation of security-critical software and data in a secure execution environment so that it cannot be observed and/or modified in an unauthorised manner 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 ROAP suite protocols, thereby mitigating replay and preplay attacks. The TPM may also be used to provide accurate time source synchronisation, as described in [15].

9 SIMLock

9.1 Use Case Description

Mobile device personalisation, or SIMLocking, is the process by which the device can be constrained to operate only with certain (U)SIMs. In earlier discussions of the GSM and DCS1800 technical specifications, the fundamental property of SIM mobility was praised as highly advantageous [79]. Over the years, however, the disadvantages associated with SIM mobility have also become apparent. Phone operators, for example, who subsidise the cost of mobile equipment, with the intention of recovering this initial loss from future profits from network or service subscriptions, may suffer a loss if mobile device users can, without authorisation from their current operator, move their phone to another network before the original subscription contract has been upheld. SIM mobility may also encourage handset theft for re-use or re-sale. These issues have led to the need for SIMLock functionality.

SIMLock has five personalisation categories:

- *Network*, where a network operator personalises a mobile device so that it can only be used with (U)SIMs from that particular network operator;
- *Network subset*, where a network operator personalises a mobile device so that it can only be used with a subset of (U)SIMs from that particular network operator;
- *Service provider*, where a service provider personalises a mobile device so that it can only be used with (U)SIMs from that particular service provider;
- *Corporate*, where a corporate customer personalises an employee's or customer's mobile device so that it can only be used with (U)SIMs belonging to that particular company; and
- *SIM/USIM*, where an end user personalises a mobile device so that it can only be used with a particular (U)SIM.

SIMLock Security-Critical Data: A personalisation indicator and a personalisation code or code group are associated with each personalisation category.

- A personalisation indicator is used to show whether a particular personalisation category is active (set to 'on') or deactivated (set to 'off'). Each category has an independent personalisation indicator. If an indicator is active it shows that the SIM has been locked to a network(s), network subset(s), service provider(s), corporate entity/entities or SIM/(U)SIM(s).
- A personalisation code or code group is used to personalise a device to a particular entity. An independent personalisation code or code group is defined for each category — see [80].

SIMLock-Related Processes: In order to personalise a device the required personalisation code or code group must be entered into the device and the

appropriate personalisation indicator set to 'on'. The relevant control key, used for device de-personalisation, must be also be stored within the device.

When a (U)SIM is inserted into the device, or when the device is powered on, the mobile device checks which personalisation indicators are set to 'on'. The personalisation agent 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. Should this checking process fail, the device enters 'limited service state' in which only emergency calls can be made [80]. This process is shown in figure 6.

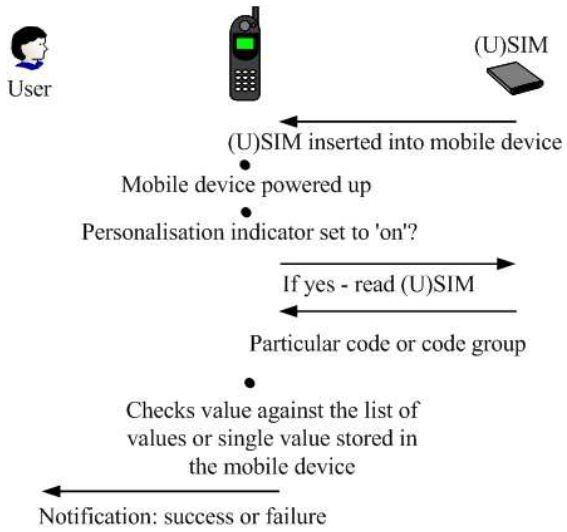

Fig. 6. SIMLock

In order to de-personalise a device, the control key for the particular personalisation category must first be entered into the device. This is then compared against the control key stored by the device. If the entered control key matches the stored value, then the personalisation indicator for the category in question is set to 'off'.

9.2 Threat Analysis

The fundamental threats to the SIMLock process include the following:

- Unauthorised modification or removal of the device personalisation agent software while in storage on or while executing on the device.
- Unauthorised reading/copying of a control key while in storage or in use on the device.

– Unauthorised modification or deletion of a personalisation code/code group, control key or personalisation indicator while in storage or in use on the device.

9.3 Secure SIMLock using Trusted Computing

The unauthorised modification or removal of the device personalisation agent cannot be prevented using trusted computing technology. However, while the software is in storage, secure boot functionality can be deployed 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. That security critical data requiring integrity protection, such as network, network subset, corporate and service provider codes or code groups and indicators, can also be incorporated into the secure boot process. TC isolation mechanisms 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 code/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 in storage. 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.

10 Software Download

10.1 Use Case Description

Two distinct types of software can be downloaded to a mobile device, namely *application software* (e.g. games) and *core software* (e.g. operating system software/updates/patches) [81]. For the purpose of this use case, we will focus on the secure download of core software, including updates or patches to the device's native OS, such as DRM agents or browsers, or firmware updates or patches. Core software download enables efficient device management, and can also be used to support applications such as Software Defined Radio (SDR) and Digital Video Broadcast (DVB) in a mobile environment.

Device Management: Core software not only enhances device management, but can also be used to enhance the end user experience. As devices become more complex, it is increasingly likely that they have to be recalled because of core software bugs [82]. The ability to download core software, however, enables more efficient bug fixing. It is also desirable that users are able to upgrade core

software on their devices, e.g. to give added functionality or enhanced security or performance [82]. As devices become more open, it is also likely that users will wish to extend the capabilities of their devices through the addition of new software, including, for example, device drivers.

SDR: A *software defined radio* is a communications device "whose operational modes and parameters can be changed or augmented, post manufacturing via software" [83]. 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. Software defined radio is an important innovation for the communications industry, providing many advantages over purely hardware-based wireless networking infrastructures and terminals. Importantly, cost reductions may result from the deployment of a generic hardware platform which can be customised using software [81]. The value of terminals is increased as public/private sector radio system sharing becomes possible, and as terminals can be upgraded to comply with evolving communications standards.

SDR also enables operation and maintenance cost reductions, as bugs can be fixed by software download rather than terminal recall. Re-configurable radios can also be adapted to meet evolving user and/or operator preferences. A terminal can moreover be reconfigured to efficiently cope with changing network conditions such as utilization, interference or radio channel quality, thereby offering an enhanced user experience [82]. Efficient roaming is also enabled, as air interface and frequency bands can be reconfigured as required.

However, while there are many advantages associated with the introduction of SDR terminals, there are also some significant security and safety issues. If SDR is to be accepted, then the security threats introduced by reconfigurable terminals and core software download must be analysed, and measures taken to mitigate these threats.

DVB: It is expected that the next generation of mobile communications systems will be able to interwork with broadcast networks to provide wireless access to video content from a wide range of mobile devices [84]. For a service like this to achieve its full commercial potential, the owners of the content will require assurance that their material is not illegally accessed. Current broadcast systems accomplish this by using conditional access systems to ensure that only bona fide subscribers have access to the content. The DVB organisation has developed several standards defining a common interface to conditional access systems at both the transmission site and at the receiver, while allowing the systems themselves to remain proprietary [85–87].

Services broadcast today are protected by a range of proprietary access control systems. The DVB common interface solution requires receiving devices to have a pc-card interface and the user to possess a number of modules, each of which implements a different conditional access system. The cost of adding such an interface to a small mobile device, as well as the practical design issues, could

make this an infeasible solution for the mobile environment. The cost of the modules may also deter some subscribers. The alternative solution, Simulcrypt, involves broadcasting each service under the control of as many conditional access systems as possible; this is likely to prove prohibitively complex and expensive for many broadcasters, especially small 'niche' providers. Both current solutions therefore have potential difficulties when applied in a mobile environment, which is likely to significantly restrict the content available to mobile receivers.

In order to overcome these limitations, the mobile platform could be reconfigured to be compatible with the appropriate conditional access system, if the proprietary system is implemented entirely in software. Such software could be delivered to the mobile device on demand. Of course, such a solution presents major security challenges.

Core Software Download — State of the Art: The model under consideration is illustrated in figure 7, and involves three parties: the user, a mobile device, and the software provider.

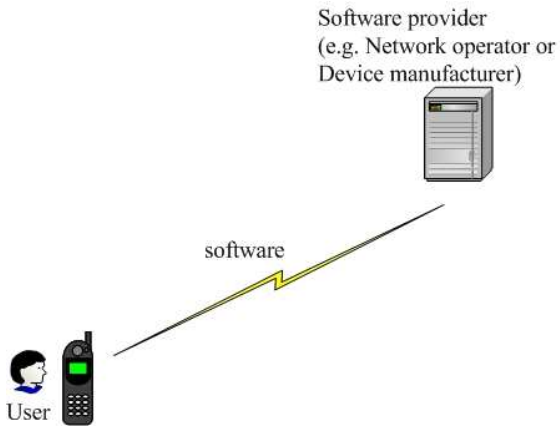

Fig. 7. Core software download model

Core software download, as defined by the OMA Device Management Working Group (DMWG) [82], consists of five stages, as shown in figure 8.

1. *Core software download initiation.* The software download process may be either network initiated or user initiated. The software provider may initiate a data connection with a device in order to:
   - request an inventory of the core software installed on the device so that the necessary software can be updated/patched/installed; or
   - inform the user of available upgrades and/or additional core software.

   Alternatively, a user may initiate a data connection with a software provider in order to request additional software over the default configuration. This

initiation results in an open data connection between the device and the software provider.

2. *Device information exchange* enables a device to inform the software provider about its current configuration. In this way the software provider can ensure that the appropriate software/updates/patches are delivered to the device. This exchange may require user authorisation.
3. *Core software download* is the process by which the core software is downloaded from the software provider to the mobile device.
4. *Core software installation* is the method by which the software download is processed on the device.
5. Finally, the software provider and/or the end user may be notified of the result of the download.

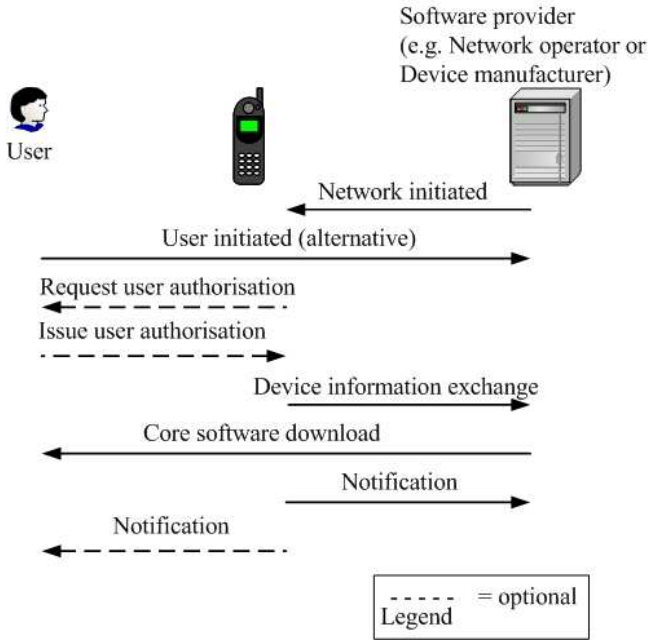

Fig. 8. Core software download

Two mechanisms have been proposed to secure non-application software download. We now briefly review these two approaches.

Firstly, the software provider could digitally sign the core software before it is downloaded to the device, thereby providing software origin authentication and software integrity protection (so that any unauthorised modification to, or addition of, incoming software can be detected by the mobile device) [81]. On receipt of the software, the digital signature of the software provider must be

verified by the mobile device. Depending on the outcome of this check and the policy of the mobile device, the software is either executed or discarded. This approach, as described in [81], is, however, susceptible to a replay attack. The mobile device has no way to determine whether the incoming signed software is fresh. An attacker could therefore replay an older version of the software, which will then be installed on the device.

It has therefore been recommended [81] that, in order to mitigate the risk of a replay attack against downloaded software, either timestamps or nonces should used so that freshness of the software can be checked. Either the mobile device generates and transmits a nonce to the software provider, which is then concatenated with the download, digitally signed by the software provider, and returned to the mobile device, or the download is concatenated with a timestamp, digitally signed by the software provider, and delivered to the mobile device. On receipt of the software, the digital signature of the software provider and the freshness mechanism must be verified by the mobile device. Depending on the outcome of the checks and the policy of the mobile device, the software is then either executed/installed or discarded.

A second approach to securing non-application software download is the use of HTTPS, i.e. HTTP carried over one of the following protocols: Transport Layer Security 1.0 (TLS 1.0); Secure Sockets Layer v3 (SSL v3); or Wireless Transport Layer Security (WTLS).

SSL, TLS and WTLS involve two protocol layers. The *record protocol* takes messages to be transmitted, optionally compresses the data, computes a Message Authentication Code (MAC), encrypts, and then transmits the result [88]. Received data is decrypted, the MAC verified, decompressed and passed to higher level clients such as HTTP for processing. The record protocol has been designed to provide: software origin authentication; confidentiality and integrity protection; and freshness, so that the replay of messages can be detected by the mobile device.

The second layer of protocols, the *record protocol clients*, includes the *handshake protocol*, the *change cipher suite protocol* and the *alert protocol*. The handshake protocol enables a client and a server to authenticate each other and to negotiate the security parameters for a client/server session, i.e. an association between a client and a server [89]. Sessions are used so that the expensive process of security parameter negotiation does not have to be completed for each connection between the client and the server [89]. Security parameters include: a session identifier, peer certificate, compression method, cipher suite and a master secret. The cipher suite specifies the MAC and encryption algorithms which will be used to protect data transmitted in an SSL/TLS/WTLS record. The handshake protocol also enables the agreement of a pre-master secret which is used by both the client and the server in order to generate a master secret for an SSL/TLS/WTLS session. This shared master secret is in turn used by the client and the server in order to generate shared MAC and encryption keys for each SSL/TLS/WTLS connection (i.e. a transient peer to peer relationship between a client and a server [89]).

The change cipher suite protocol consists of only one message. This message is sent either by the client or the server at the end of the handshake protocol to notify the other party that the newly negotiated ciphersuite and master secret will be utilised in the protection of all subsequent records. The alert protocol is used to convey alerts to the peer entity [89].

10.2 Threat Analysis

The threats which impact upon a reconfigurable mobile device can divided into:

- those which impinge on the security of the downloaded reconfiguration software; and
- those which impinge on host security.

The fundamental threat to the security of the downloaded reconfiguration software is unauthorised reading of software while in transit between the software provider and the end host, or while in storage or executing on the end host. This threat could result in an infringement of the intellectual property rights associated with the downloaded reconfiguration software. It might also result in unauthorised access to, and execution of, software.

Fundamental threats to end host security include:

- malicious or accidental modification or removal of security-critical software and data when incorporated into, while in storage on, or when executing on, the end host;
- the download of inappropriate reconfiguration software which does not meet the capability requirements of the mobile device; and
- malicious or accidental modification, addition or removal of downloaded software in development, in transit or while in storage or executing on the end host.

As a result of these threats, a device might be rendered inoperable (a denial of service attack), or user applications and/or data could be compromised by malicious software.

More specifically, in the case of SDR these threats could result in the following.

- An inoperable device. If, for example, a device used software modulation, an improper change of the modulation format could render it inoperable [90].
- Violation of Radio Frequency (RF) spectrum rights. This could, for example, result in RF interference. If a device can be programmed to transmit on a frequency for which it is not authorised, signals from nodes which are authorised to use this frequency might be jammed [90]. Also, spurious emissions resulting from unauthorised radio spectrum use could violate user safety [91].
- Increased output power. If, for example, a device operated at maximum power, its performance may be increased at the expense of other users in the communications network [90]. This in turn may force other users to use increased power. As a result, the device battery life would be severely shortened; moreover, if the radiated power is sufficiently high, user safety may also be put at risk [91].

10.3 Secure Software Download using Trusted Computing

We now investigate some of the ways in which trusted computing functionality can be used to address the threats outlined above or, failing that, to limit the level to which a threat may be exploited.

Protecting the Reconfiguration Software: TC mechanisms can be used to confidentiality-protect the reconfiguration software while in transit between the software provider and the end host, while in storage or executing on the end host, and to ensure that only the intended recipient device can access the software. For example, a software download protocol which exploits trusted computing functionality is described in [35, 92]. This protocol builds upon sealed storage, platform attestation, and isolation techniques. This protocol is now summarised.

Before the required reconfiguration software can be downloaded to a TP, the TPM is used to generate an asymmetric key pair. This key pair 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 only when the platform is in the specified state. The public key from this pair, and the integrity metrics with which its private key are associated, are then certified by the TPM using a TP AIK, as described in section 6.3, so that the state to which the private key is bound can be shown to the software provider. The certified public key and the corresponding AIK credential are then sent to the software provider.

On receipt of the certified key and the AIK credential, the software provider verifies the TP's AIK credential and the signature of the TPM on the public key and the associated integrity metrics. If these two elements can be verified, and if the software provider considers the platform software state to which the key is bound to be trustworthy, the provider computes a MAC on and encrypts the reconfiguration software, encrypts the secret MACing and encryption keys using the public key received from the TP, signs the encrypted keys using its private signature key, and transmits this data to the TP. The secret keys received by the TP, and therefore the reconfiguration software, can only be accessed when the TP is in the state deemed trustworthy by the software provider. The software provider may require that the integrity metrics to which the private key is bound, represent an isolated execution environment executing on a specified isolation layer, which is turn is supported by a TP which incorporates hardware extensions that enable efficient and secure isolation, as described in section 6.4.

The confidentiality of the reconfiguration software can thus be protected while it is in transit between the software provider and the TP, in storage, and executing on the device. The software provider is also given assurance that only a specified TP in a particular state can access the software.

Alternatively, if a more traditional mechanism such as SSL/TLS/WTLS is used in order to secure the download of the reconfiguration software, TC functionality can be used in order to 'harden' the SSL/TLS/WTLS implementation. In this case, prior to the completion of any SSL/TLS/WTLS protocol, the TPM is used in order to generate the client-side (the mobile device) asymmetric key pair for authentication, which is bound to a set of integrity metrics so that the

private key can only be utilised by the TPM on which it was generated, and only when the platform is in the required state. This key is then certified using a TP AIK. Evidence that this SSL/TLS/WTLS key pair has been generated on, and certified by, a TPM is then provided by a Certification Authority (CA) in an extension of the mobile device's X.509 SSL/TLS/WTLS certificate.

During an SSL/TLS/WTLS protocol run between a software provider and the mobile device, the information provided in the extension of the mobile device's X.509 SSL/TLS/WTLS public key certificate enables a software provider to trust that the mobile device's private SSL/TLS/WTLS key is held within a TPM, and that the key can only be used when the platform is in a particular state. As above, the software provider may require that the integrity metrics to which the private key is bound, represent an isolated execution environment into which the software will be downloaded and executed. This hardened implementation of SSL/TLS/WTLSS gives the software provider assurance that the mobile device's SSL/TLS/WTLS 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. This process is described in [93].

Protecting the Host's Security-Critical Software: While the integrity of security-critical software while in storage cannot be ensured using TC functionality, TC mechanisms can be used to help detect its malicious or accidental modification or removal, through the deployment of secure boot functionality. Security-critical data, such as device private keys, the public key certificate store and the core software download policy, can also be verified during a secure boot process.

Alternatively, sealed storage functionality can be used to ensure that security-critical data is stored in encrypted form and only accessible when the mobile platform is in a predefined state. As described above, the TPM can also be used to generate asymmetric key pairs, as well as protect any private keys while in storage and in use on the device.

TC functionality also enables the isolation of security-critical software and data in a secure execution environment so that it cannot be observed or modified when in use by software executing in parallel insecure execution environments.

Protecting the Host from Reconfiguration Software: A capability exchange could be performed by the network and the mobile device prior to software download, to ensure that the appropriate software entities and parameter sets are selected for a particular mobile device. The use of platform attestation, as described in section 6.3, could be used to ensure that the reports sent by the device are accurate.

TC cannot prevent denial of service attacks resulting from the removal/deletion of the downloaded reconfiguration software, either in development, or in transit between the software provider and the host or in storage on the host. However, standard cryptographic mechanisms, such as digital signatures and/or MACs,

can be used in combination with freshness mechanisms to mitigate these threats. TC functionality can be used to make such mechanisms more robust. A good quality random number generator is provided by a TPM, thereby enabling the generation of non-repeating unpredictable nonces which may then be sent to a software provider and returned from to a trusted mobile platform in conjunction with the requested software, thereby mitigating replay and preplay attacks against the exchange. If timestamps are used in order to guarantee the freshness of the downloaded software, the TPM could be used to provide an accurate and trusted time source, as described in [15].

In the advent of malicious or buggy software being downloaded to and executed on a device, there are a number of ways in which TC can lessen the impact of this threat. If the downloaded software is isolated in its own execution environment, as described in section 6.4, then any malicious behaviour can be controlled and its effects limited. If sealed storage is used to protect private user data (e.g., credit card numbers), then the impact of malicious software is lessened, as it cannot gain access to security sensitive data which has been protected. On connection/reconnection to a service provider, a trusted mobile platform could be required to attest to its state so that a decision can be made as to whether the device should be considered trusted for a particular purpose.

11 Conclusions

In this article we have reviewed the main functional components of a trusted platform. We have also considered why such functionality is necessary, and how the technology might be used. We have then considered possible applications of this technology to mobile devices, and have considered in detail three specific applications. In each case we have discussed how the security functionality necessary for the application could be supported using the trusted computing capabilities.

References

1. TCG: TCG Specification Architecture Overview. TCG Specification Version 1.2, The Trusted Computing Group (TCG), Portland, Oregon, USA (April 2003)
2. Grawrock, D.: The Intel Safer Computing Initiative. Intel Press, Oregon, USA (2006)
3. Proudler, G.: Concepts of trusted computing. In Mitchell, C.J., ed.: Trusted Computing. IEE Professional Applications of Computing Series 6. The Institute of Electrical Engineers (IEE), London, UK (April 2005) 11–27
4. Pearson, S., ed.: Trusted Computing Platforms: TCPA Technology in Context. Prentice Hall, Upper Saddle River, New Jersey, USA (2003)
5. Gollmann, D.: Computer Security. 2nd edn. John Wiley and Sons Ltd. (2005)
6. Pfleeger, C.P.: Security in Computing. 2nd edn. Prentice Hall PTR, Upper Saddle River, NJ (1997)
7. Department of Defense: DoD 5200.28-STD: Department of Defense Trusted Computer System Evaluation Criteria. (1985)

8. IBM: PCI Cryptographic Processor: CCA Basic Services Reference and Guide, Release 2.41. (September 2003)
9. Dent, A.W., Mitchell, C.J.: User's Guide to Cryptography and Standards. Artech House, Boston, MA (2005)
10. Pearson, S.: Trusted computing platforms, the next security solution. Technical Report HPL-2002-221, Hewlett-Packard Laboratories (November 2002) Available at `http://www.hpl.hp.com/techreports/`.
11. Varadharajan, V.: Trustworthy computing. In Zhou, X., Su, S., Papazoglou, M., Orlowska, M.E., Jeffery, K.G., eds.: Web Information Systems — WISE 2004: 5th International Conference on Web Information Systems Engineering, Brisbane, Australia, November 22-24, 2004, Proceedings. Volume 3306 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2004) 13–16
12. Felten, E.W.: Understanding trusted computing: Will its benefits outweigh its drawbacks? IEEE Security & Privacy **1**(3) (May/June 2003) 60–62
13. Balacheff, B., Chen, L., Pearson, S., Proudler, G., Chan, D.: Computing platform security in cyberspace. Information Security Technical Report **5**(1) (2000) 54–63
14. Chen, L., Pearson, S., Proudler, G., Chan, D., Balacheff, B.: How can you trust a computing platform? In: Proceedings of Information Security Solutions Europe (ISSE 2000). (2000)
15. TCG: TPM Main, Part 1: Design Principles. TCG Specification Version 1.2 Revision 94, The Trusted Computing Group (TCG), Portland, Oregon, USA (March 2006)
16. TCG: TPM Main, Part 2: TPM Data Structures. TCG Specification Version 1.2 Revision 94, The Trusted Computing Group (TCG), Portland, Oregon, USA (March 2006)
17. TCG: TPM Main, Part 3: Commands. TCG Specification Version 1.2 Revision 94, The Trusted Computing Group (TCG), Portland, Oregon, USA (March 2006)
18. TCG: TCG Software Stack (TSS) Specification. TCG Specification Version 1.2 Level 1, The Trusted Computing Group (TCG), Portland, Oregon, USA (January 2006)
19. Chen, Y., England, P., Peinado, M., Willman, B.: High assurance computing on open hardware architectures. Microsoft Technical Report MSRTR–2003–20, Microsoft Corporation (March 2003)
20. England, P., Lampson, B., Manferdelli, J., Peinado, M., Willman, B.: A trusted open platform. IEEE Computer **36**(7) (July 2003) 55–62
21. Peinado, M., Chen, Y., England, P., Manferdelli, J.: NGSCB: A trusted open system. In Wang, H., Pieprzyk, J., Varadharajan, V., eds.: Proceedings of 9th Australasian Conference on Information Security and Privacy, ACISP 2004. Volume 3108 of Lecture Notes in Computer Science (LNCS)., Sydney, Austrailia, Springer–Verlag, Belin–Heidelberg, Germany (13–15 July 2004) 86–97
22. Peinado, M., England, P., Chen, Y.: An overview of NGSCB. In Mitchell, C.J., ed.: Trusted Computing. IEE Professional Applications of Computing Series 6. The Institute of Electrical Engineers (IEE), London, UK (April 2005) 115–141
23. Garfinkel, T., Rosenblum, M., Boneh, D.: Flexible OS support and applications for trusted computing. In: Proceedings of the 9th USENIX Workshop on Hot Topics on Operating Systems (HotOS-IX), Kauai, Hawaii, USA, USENIX, The Advanced Computing Systems Association (18-21 May 2003) 145–150
24. Pfitzmann, B., Riordan, J., Stuble, C., Waidner, M., Weber, A.: The PERSEUS system architecture. Technical Report RZ 3335 (#93381), IBM Research Division, Zurich Laboratory (April 2001)

25. Sadeghi, A., Stuble, C.: Taming "Trusted Platforms" by operating system design. In Chae, K., Yung, M., eds.: Proceedings of Information Security Applications, 4th International Workshop (WISA 2003). Volume 2908 of Lecture Notes in Computer Science (LNCS)., Jeju Island, Korea, Springer-Verlag, Berlin-Heidelberg, Germany (25–27 August 2003)
26. Kuhlmann, D., Landfermann, R., Ramasamy, H., Schunter, M., Ramunno, G., Vernizzi, D.: An open trusted computing architecture — secure virtual machines enabling user-defined policy enforcement. www.opentc.net (June 2006)
27. Sadeghi, A.R., Stueble, C., Pohlmann, N.: European multilateral secure computing base — open trusted computing for you and me. White paper (2004)
28. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauery, R., Pratt, I., Warfield, A.: XEN and the art of virtualization. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003), Bolton Landing, New York, USA, ACM Press, New York, USA (19–22 October 2003) 164–177
29. Intel: LaGrande technology architectural overview. Technical Report 252491-001, Intel Corporation (September 2003)
30. Balacheff, B., Chen, L., Plaquin, D., Proudler, G.: A trusted process to digitally sign a document. In Raskin, V., Hempelmann, C.F., eds.: Proceedings of the 2001 New Security Paradigms Workshop, ACM Press (2001) 79–86
31. Spalka, A., Cremers, A.B., Langweg, H.: Protecting the creation of digital signatures with trusted computing platform technology against attacks by Trojan Horse programs. In Dupuy, M., Paradinas, P., eds.: Trusted Information: The New Decade Challenge, IFIP TC11 Sixteenth Annual Working Conference on Information Security (IFIP/Sec'01), June 11-13, 2001, Paris, France. Volume 193 of IFIP Conference Proceedings., Kluwer Academic Publishers, Boston (2001) 403–419
32. Schechter, S.E., Greenstadt, R.A., Smith, M.D.: Trusted computing, peer-to-peer distribution, and the economics of pirated entertainment. In: Proceedings of The Second Annual Workshop on Economics and Information Security. (2003) College Park, Maryland, May 29–30, 2003.
33. Kinateder, M., Pearson, S.: A privacy-enhanced peer-to-peer reputation system. In Bauknecht, K., Min Tjoa, A., Quirchmayr, G., eds.: E-Commerce and Web Technologies, 4th International Conference, EC-Web, Prague, Czech Republic, September 2-5, 2003, Proceedings. Volume 2738 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2003) 206–216
34. Balfe, S., Lakhani, A.D., Paterson, K.G.: Securing peer-to-peer networks using trusted computing. In Mitchell, C.J., ed.: Trusted Computing. The Institute of Electrical Engineers (IEE), London, UK (2005) 271–298
35. Gallery, E., Tomlinson, A.: Secure delivery of conditional access applications to mobile receivers. In Mitchell, C.J., ed.: Trusted Computing. IEE Professional Applications of Computing Series 6. The Institute of Electrical Engineers (IEE), London, UK (2005) 195–238
36. Pashalidis, A., Mitchell, C.J.: Single sign-on using trusted platforms. In Boyd, C., Mao, W., eds.: Information Security, 6th International Conference, ISC 2003, Bristol, UK, October 1-3, 2003, Proceedings. Volume 2851 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2003) 54–68
37. Chen, L., Pearson, S., Vamvakas, A.: On enhancing biometric authentication with data protection. In Howlett, R.J., Jain, L.C., eds.: Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Volume 1., IEEE (2000) 249–252

38. Mont, M.C., Pearson, S., Bramhall, P.: Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. In: 14th International Workshop on Database and Expert Systems Applications (DEXA '03), September 1-5, 2003, Prague, Czech Republic, IEEE Computer Society (2003) 377–382
39. Mont, M.C., Pearson, S., Bramhall, P.: Towards accountable management of privacy and identity information. In Snekkenes, E., Gollmann, D., eds.: Computer Security — ESORICS 2003, 8th European Symposium on Research in Computer Security, Gjøvik, Norway, October 13-15, 2003, Proceedings. Volume 2808 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2003) 146–161
40. Pridgen, A., Julien, C.: A secure modular mobile agent system. In: Proceedings of the 2006 international workshop on Software engineering for large-scale multi-agent systems (SELMAS '06), Shanghai, China, ACM Press, New York, USA (22–23 May 2006) 67–74
41. Pearson, S.: How trusted computers can enhance for privacy preserving mobile applications. In: Proceedings of the 1st International IEEE WoWMoM Workshop on Trust, Security and Privacy for Ubiquitous Computing (WOWMOM '05), Taormina, Sicily, Italy, IEEE Computer Society, Washington, DC, USA (13–16 June 2005) 609–613
42. Pearson, S.: Trusted agents that enhance user privacy by self-profiling. Technical Report HPL-2002-196, HP Labs, Bristol, UK (15 July 2002)
43. Crane, S.: Privacy preserving trust agents. Technical Report HPL-2004-197, HP Labs, Bristol, UK (11 November 2004)
44. Balfe, S., Mohammed, A.: Final fantasy — securing on-line gaming with trusted computing. In: Proceedings of the 4th International Conference on Autonomic and Trusted Computing (ATC-07), Hong Kong (July 2007)
45. Yan, Z., Cofta, Z.: A method for trust sustainability among trusted computing platforms. In Katsikas, S., Lopez, J., Pernul, G., eds.: Trust and Privacy in Digital Business: First International Conference, TrustBus 2004, Zaragoza, Spain, August 30 – September 1, 2004, Proceedings. Volume 3184 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2004) 11–19
46. Anderson, R.: Cryptography and competition policy — Issues with 'trusted computing'. In: Proceedings of PODC '03, July 13-16, 2003, Boston, Massachsetts, USA, ACM (2003) 3–10
47. Schoen, S.: Trusted computing: Promise and risk. Electronic Frontier Foundation Article (October 2003)
48. von Lohmann, F.: Meditations on trusted computing. Electronic Frontier Foundation Article (2003)
49. Reid, J., Gonzalez Nieto, J.M., Dawson, E.: Privacy and trusted computing. In: 14th International Workshop on Database and Expert Systems Applications (DEXA '03), September 1-5, 2003, Prague, Czech Republic, IEEE Computer Society (2003) 383–388
50. Arbaugh, B.: Improving the TCPA specification. IEEE Computer **35**(8) (August 2002) 77–79
51. Tygar, J., Yee, B.: Dyad: A system for using physically secure coprocessors. Technical Report CMU-CS-91-140R, Carnigie Mellon University, Pittsburgh, Pennsylvania, USA (May 1991)
52. Clark, P., Hoffman, L.: BITS: a smartcard protected operating system. Communications of the ACM **37** (November 1994) 66–94

53. Arbaugh, W., Farber, D., Smith, J.: A secure and reliable bootstrap architecture. In: 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) 65–71
54. Itoi, N., Arbaugh, W., Pollack, S., Reeves, D.: Personal secure booting. In: Proceedings of the 6th Australasian Conference on Information Security and Privacy ACISP 2001. Volume 2119 of Lecture Notes In Computer Science (LNCS)., Sydney, Australia, Springer–Verlag, London, UK (11–13 July 2001) 130–141
55. Lie, D.: Architectural Support for Copy and Tamper Resistant Software. PhD thesis, Department of Electrical Engineering, Stanford University, Stanford, California, USA (December 2003)
56. Lie, D., Thekkath, C., Mitchell, M., Lincoln, P., Boneh, D., Mitchell, J., Horowitz, M.: Architectural support for copy and tamper resistant software. In: 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) 169–177
57. Suh, E., Clarke, D., Gassend, B., van Dyke, M., Devadas, S.: The AEGIS processor architecture for tamper–evident and tamper-resistant processing. In: 17th Annual ACM International Conference on Supercomputing (ICS'03), San Francisco, California, USA, ACM Press, New York, USA (23–26 June 2003) 160–171
58. Barrett, M.F.: Towards an open trusted computing framework. Masters thesis, Department of Computer Science, The University of Auckland, New Zealand (February 2005)
59. TCG: TCG PC client specific implementation specification for conventional BIOS. TCG specification Version 1.2 Final, The Trusted Computing Group (TCG), Portland, Oregon, USA (July 2005)
60. Abraham, D., Jackson, J., Muthrasanallur, S., Neiger, G., Regnier, G., Sankaran, R., Schionas, I., Uhlig, R., Vembu, B., Wiegert, J.: Intel virtualization technology for directed i/o. Intel Technology Journal **10**(3) (August 2006) 179–192
61. TCG: TCG EFI platform — for TPM family 1.1 or 1.2. TCG specification Version 1.2 Final, The Trusted Computing Group (TCG), Portland, Oregon, USA (June 2006)
62. TCG: TCG Credential Profiles. TCG Specification Version 1.1 Revision 1.014 For TPM Family 1,2; Level 2, The Trusted Computing Group (TCG), Portland, Oregon, USA (May 2007)
63. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. Technical Report HPL-2004-93, Hewlett-Packard Laboratories (June 2004) Available at `http://www.hpl.hp.com/techreports/`.
64. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In Pfitzmann, B., Liu, P., eds.: Proceedings of CCS '04, ACM Press (2004) 132–145
65. Brickell, E., Camenisch, J., Chen, L.: The DAA scheme in context. In Mitchell, C.J., ed.: Trusted Computing. IEE Professional Applications of Computing Series 6. The Institute of Electrical Engineers (IEE), London, UK (2005) 143–174
66. TCG MPWG: Use Case Scenarios. TCG Specification Version 2.7, The Trusted Computing Group, Mobile Phone Working Group, Portland, Oregon, USA (September 2005)
67. TCG MPWG: The TCG mobile trusted module specification. TCG specification version 0.9 revision 1, The Trusted Computing Group (TCG), Portland, Oregon, USA (September 2006)

68. TCG MPWG: Mobile trusted module specification overview document. Mobile trusted module specification support documents, The Trusted Computing Group (TCG), Beaverton, Oregon, USA (2006)
69. Gallery, E.: Authorisation issues for mobile code in mobile systems. Technical Report RHUL-MA-2007-3, Department of Mathematics, Royal Holloway, University of London (2007)
70. OMA: DRM architecture v2.0. Technical Specification OMA-DRM-ARCH-V2_0-2004071515-C, The Open Mobile Alliance (OMA) (July 2004)
71. Irwin, J., Wright, T.: Digital rights management. Vodafone internal newsletter, Vodafone, Newbury, England, UK (August 2004)
72. OMA: OMA DRM V1.0 approved enabler specification. Technical Specification OMA-DRM-V1_0-20040625-A, The Open Mobile Alliance (OMA) (June 2004)
73. OMA: OMA DRM V2.0 approved enabler specification. Technical Specification OMA-ERP-DRM-V2_0-20060303-A, The Open Mobile Alliance (OMA) (July 2004)
74. OMA: Digital Rights Management v1.0. Technical Specification OMA-Download-DRM-V1_0-20040615-A, The Open Mobile Alliance (OMA) (June 2004)
75. OMA: DRM architecture specification v1.0. Technical Specification OMA-Download-ARCH-V1_0-20040625-A, The Open Mobile Alliance (OMA) (June 2004)
76. OMA: DRM specification v2.0. Technical Specification OMA-DRM-DRM-V2_0-20040716-C, The Open Mobile Alliance (OMA) (July 2004)
77. CMLA: Client adopter agreement. Technical Report Revision 1.00-050708, The Content Management License Administrator Limited Liability Company (CMLA, LLC) (August 2005)
78. Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 internet public key infrastructure: Online certificate status protocol — OCSP. RFC 2560, Internet Engineering Task Force (IETF) (June 1999)
79. Mouley, M., Pautet, M.: The GSM System for Mobile Communications. Cell & Sys. Correspondence, Palaiseau, France (1992)
80. 3GPP GSM TSGS: Personalisation of mobile equipment (ME), Mobile functionality specification (release 5). Technical specification TS 22.022 v5.0.0, 3rd Generation Partnership Project (3GPP), Global System for Mobile Communications (GSM) — Technical Specification Group Services and System Aspects, Sophia Antipolis, France (2002)
81. NTT DoCoMo, IBM, Intel Corporation: Trusted Mobile Platform. (May 2004)
82. OMA: Device management requirements candidate version 1.2. Technical Specification OMA-RD-DM-V1_2-20060424-C, The Open Mobile Alliance (OMA) (April 2006)
83. SDRF: Overview and definition of software download for rf reconfiguration. SDRF Archived Approved Document DL-DFN Document SDRF-02-A-0002-V.0.0, The Software Defined Radio Forum (SDRF) (August 2002)
84. Tuttlebee, W., Babb, D., Irvine, J., Martinez, G., Worrall, K.: Broadcasting and Mobile Telecommunications: Interworking — Not Convergence. EBU Technical Review **293** (January 2003) 1–11
85. European Committee for Electrotechnical Standardization (CENELEC) Brussels, Belgium: Common Interface Specification for Conditional Access and other Digital Video Broadcasting Decoder Applications. (February 1997)
86. European Telecommunications Standards Institute (ETSI) Sophia-Antipolis, France: Digital Video Broadcasting (DVB): Head-End Implementation of DVB Simulcrypt. (January 2003)

87. European Telecommunications Standards Institute (ETSI) Sophia-Antipolis, France: Digital Video Broadcasting (DVB); Support for use of Scrambling and Conditional Access (CA) within Digital Broadcasting Systems. (October 1996)
88. WAPF: Wireless transport layer security version 06. Technical Specification WAP-2610WTLS-20010406-a, The Wireless Application Protocol Forum (WAPF) (April 2001)
89. Stallings, W.: Cryptography and Network Security, Principles and Practices. 2nd edn. Prentice Hall, Upper Saddle River, New Jersey (1999)
90. Hill, R., Myagmar, S., Campbell, R.: Threat analysis of GNU software radio. In: Proceedings of the World Wireless Congress (WWC 2005), Palo Alto, California, USA (24–27 May 2005)
91. SDRF: Security considerations for operational software defined radio devices in a commercial wireless domain. SDRF working document, The Software Defined Radio Forum (SDRF) (October 2004)
92. Gallery, E., Tomlinson, A.: Protection of downloadable software on SDR devices. In: Proceedings of the 4th Software Defined Radio Forum Technical Conference (SDR 2005), Orange County, California, USA, Software Defined Radio Forum (SDRF) (14–18 November 2005)
93. TCG: Subject key attestation evidence extension. TCG specification version 1.0 revision 7, The Trusted Computing Group (TCG), Portland, Oregon, USA (June 2005)