Enriched Trusted Platform and its Application on DRM

Yongdong Wu and Feng Bao
Cryptography and Security Department
Institute for Infocomm Research, Singapore
{wydong, baofeng}@i2r.a-star.edu.sg

Abstract

The TCG (Trusted Computing Group) is an industry working group which aims to establish industry standards for trust and security in computing platforms. This paper enriches the TCG architecture by adding a SPM (Secure Process Manager) into the trusted platform as a kernel component for the purpose of process management. To attest a process/software to a remote peer, SPM will attest itself first and then sign the software description for remote verification. In comparison with the direct process attestation method, this indirection method simplifies the attestation significantly in the case of software updating, since the number of SPM versions is much fewer than that of softwares. Moreover, this paper introduces a DRM (Digital Right Management) scheme over the enriched architecture so as to enforce usage control with the standard resource in TPM (Trusted Platform Module) chip.

1 Introduction

General-purpose computer platforms provide the convenience for developing and running software applications. However, it also allows a malicious user to tamper with and possibly control any of its software components at will so as to make illegal profit. Hence, many software vendors employ software protection technologies to counter such threats. For instance, AsProtected [1] is developed as a prevent tool against dumping or debugging. Nonetheless, it has been generally accepted in the information security research community that software alone cannot provide an adequate foundation for building a high-assurance trusted application [2]. Therefore, many big companies such as HP, IBM, Intel and Microsoft founded an industry organization called as TCG [3] who defines the specification for a security chip called as TPM [4] who provides on-chip key pair generation using a hardware random number generator, as well as RSA cryptographic functions. The TPM communicates with the system CPU, and acts only under the control of the system CPU and the policies codified in the OS and TPM-enabled software applications. Since TPM and its associated technologies have many important applications, they are rapidly gaining momentum in the computing world. For example, Windows Vista makes use of TPM to facilitate BitLocker drive encryption; the U.S. Army requires that every new small PC it purchases must come with a TPM; and IDC (International Data Corporation) [6] reported essentially all portable PCs and the vast majority of desktops will include a TPM chip by 2010.

Besides defining the TPM specification, TCG also specifies an authenticated boot process for binary codes so as to give an indication of which software configuration runs on a given computing platform. Therefore, it requires the (local or remote) verifier to know all possible “trusted” configurations as well as managing updates and patches that change the configuration. Obviously, it will be very difficult to massively deploy this kind of trusted platform since it will result in high false rejection rate unless a complete configuration database is available all the time. To increase TCG platform availability, the researchers designed two technologies.

- Property attestation: Chen et al. [7] replaced the binary attestation with property-based attestation [8] which requires to only attest whether a platform or an application fulfills the desired (security) requirements without revealing the specific software and hardware configuration. Similarly, Kuehn et al. [9] presented an enhanced boot loader which translates binary measurements into properties, allowing to attest properties of unmodified operating systems loaded. However, it is not clear in their papers how to map software vari-
Visualization: Presti [10] proposed the Tree of Trust concept which is a tree whose nodes represent the various platform components, from the hardware TPM up to the running applications, annotated with trust and security statements. Since virtualisation technology provides good abstract on the applications and TPM chips, it increases the flexibility of the system configuration.

Although TCG allows a TPM-enabled platform to run an authenticated software or its versions only, it does not provide any clear specification on what kind of software can be regarded as trusted and hence be authenticated. In fact it is unlikely that there will be a clear line between the software that should be authenticated and those should not, e.g., the popular softwares, such as debugger SoftICE for developing binary codes and MICROSOFT Internet Explorer for running applets, may have unexpected side-effects since they can load/execute/manipulate binary executables. This ambiguity may incur abuse attack [11] which enables an attacker to use an authenticated software to monitor and change the internal states of other trusted applications. For example, abuse attack enables to tamper a DRM (Digital Right Management) application on the fly so as to violate its access privilege. To defeat the abuse attack, Wu et al. [11] provide the OS-encapsulation scheme which ensures that only the genuine software loader can start and authenticate a to-be-trusted application.

Albeit DRM is one of major applications of TCG but TCG does not specify a concrete DRM solution. The OS-encapsulation scheme [11] enables TPM-enabled DRM in a loose but more practical environment, where people are allowed to use the debugger, web browser, etc. However, it merely cares about the local access control and omit addressing the document delivery process. On the contrary, as a document sharing scheme, Hades [12] is based on a client/server model beyond trusted platform. It protects the document access by detecting and handling a comprised platform so as to manage the document decryption key. However, schemes [11, 12] do not address how to handle the security of software upgrading.

To enable the DRM application which may be changed/upgraded from time to time, this paper adds a Secure Process Manager (SPM) into the kernel of TPM platform so as to enforce access control policies beyond the trusted hardware. SPM monitors and verifies the integrity and property of running software in a platform using the functions of trusted computing. As an example application of SPM, this paper presents a DRM scheme based on TCG architecture with SPM. It enables to enforce usage control with the standard resource in TPM chip.

The remainder of this paper is organized as follows. Section 2 overviews TPM. Sections 3 introduces the enriched trusted platform, and Section 4 elaborates the DRM scheme over the enriched platform. Section 5 discusses the security and performance of the proposed scheme. Section 6 summaries the paper.

2 Overview of TPM

2.1 TPM component

To provide the trusted computing platform with root of trust, protected storage, key generation and attestation, a TPM module includes: (1) processor, (2) hardware Random Number Generator (RNG), (3) cryptographic engine, (4) non-volatile memory, (5) tamper control circuitry, (6) non-volatile storage space, and (7) real-time clock1.

The TPM processor controls the functions and sequencing of the entire TPM module. It moves data between CPU and the internal TPM memory, sequences the cryptographic engine, and helps offload the RSA computation from CPU.

The RNG is the source of seed numbers for the cryptographic engine’s encryption and decryption functions, and for the generation of RSA key pairs. It also provides random values for generating signatures.

Secure storage stores sensitive data such as the permanent Endorsement Key (EK) and Storage Root Key (SRK). EK is a 2,048-bit RSA public and private key pair, which is created randomly on the chip at manufacture time and cannot be changed. The private key never leaves the chip, while the public key is used for attestation and for encryption of sensitive data sent to the chip. EK is bound to a unique platform by the manufacturer, while SRK is tied to the TPM owner who takes ownership of the TPM. All other cryptographic keys are protected by the SRK. An Attestation Identity Key (AIK), generated by TPM, is a special key used for generating digital signatures. Each TPM can have as many AIKs as it needs for the sake of anonymity. Secure storage protects private information by binding it to platform configuration information including the software and hardware being used, i.e., the data can be read only by the same combination of software and hardware.

Generally, TPM platform also provides trusted path and memory curtaining. Trusted path protects data path between the computer user and the software, while memory curtaining extends common memory protection techniques to provide full isolation of process memory space.

---

1TCG specification 1.2 does not include the build-in clock, but some TPM products have tick counter with tamper detection, e.g. http://www.infineon.com/cms/en/product/channel1.html?channel1=ff80808112ab681d0112ab6921ae011f
2.2 Transitive trust

In the trusted platform, the initialization and management functions in the TPM allow the owner to turn its functionality on and off, reset the chip, and take ownership of the chip. This group of functions provides strong separation of what can be done at BIOS boot time and what can be done at normal run-time, so that sensitive operations such as reading the endorsement key cannot be performed by malicious applications trying to compromise the TPM. Concretely, the trusted boot functions compute hashes of configuration information throughout the boot sequence, and store the hashes in Platform Configuration Registers (PCRs). If a TPM-enabled computer is reset, all the PCRs are reset too. But at any other time, the PCRs can only be extended. Concretely, when a TPM-enabled system is started, an authenticated booting sequence will be performed as shown in Fig.1. The execution units consist of the Core Root of Trust for Measurement (CRTM), the Master Boot Record (MBR), boot sector, boot code, boot manager, the OS loader, and the OS. Each unit in the process will have its integrity measured before execution control is handed over to it. If any of the integrity measurements is invalid, all the subsequent units will not run. Only if all the units are intact, the application will be verified and executed.

![Transitive trust applied to system boot from a static root of trust](image)

Figure 1. Transitive trust applied to system boot from a static root of trust[13]. Trusted Building Blocks (TBB) are the parts of the Roots of Trust that do not have shielded locations or protected capabilities. Normally these include just the instructions for the Root of Trust for Measurement (RTM) and TPM initialization functions (e.g. reset).

3 Enriched Trusted Computing Architecture

3.1 Objectives

According to the TCG specification, if a trusted platform wants to communicate with others, it has to start a remote attestation process with a TPM-generated certificate stating which software is currently running such that authorized parties can detect the changes to the user’s computer. Technically, in order to convince a remote party that a local software has not been tampered with, the local TPM will sign PCR (Platform Configuration Register) of the software and send the signature to the remote party. The remote party will not authenticate the software unless the signature matches the authentication token that the verifier has. This remote attestation mechanism is inconvenient when the software is changed/upgraded from time to time. To weaken this attestation requirement so as to run legacy applications and trusted applications simultaneously, an enriched architecture should meet the following goals:

- Legacy application and trusted application can run on the same platform simultaneously. And each of them can run as it virtually monopolies the resource except the performance in terms of speed etc.

- Neither the legacy application nor trusted applications can undermine the integrity and confidentiality of the protected runtime environment for each application. For example, the memory space of an application is private, which is not accessible to other applications even for devices with direct memory access (DMA) capability.

- The trusted application such as DRM application can run correctly despite the components of the trusted platform are upgraded.

To meet the above requirements, the trusted applications are assumed to be signed by its producer such that the integrity of the code can be verified by anyone.

3.2 Enriched architecture

With regard to Fig.2, the enriched architecture includes three layers so as to handle the software upgrading over trusted platform. The hardware layer has legacy hardware and trusted hardware. As specified by TCG, the trusted hardware must include a TPM chip, memory curtaining components and the voltage monitor. The secure kernel provides separation of execution between upper layer applications and related services such as Next-Generation Secure Computing Base (NGSCB). Additionally, we embed a new component which is called SPM (Secure Process Manager).
so as to provide a trusted interface between application and secure kernel. In the system booting stage, SPM will be started/booted authentically since it is a component of the trusted kernel.

![Figure 2. Enriched Trusted Computing.](image)

When an application App is to be loaded, SPM will verify the original application code based on its signature from the application producer. Technically, assume there is a certificate authority CA who issues certificate to software companies. Given that a software App which is produced by a software company who has public/private key pair PK/SK and its certificate. Whenever the software producer upgrades its software, it should generate a new signature of the new App. Therefore, the SPM can always verify any version of App. To attest the application App to a remote party, the local SPM will attest itself first, and then sign the description of App so that the remote party can trust the App. This indirect attestation process between two platforms is illustrated in Subsection 4.4 for securely delivering document transmission in DRM applications.

4 Application on DRM

4.1 DRM architecture

The underlying motivation for having DRM applications is that digital objects should remain under the control of their creators, rather than the owner of the platform containing the objects. One major threat to DRM is the tampering of the DRM application and its platform since a traitor can modify them so as to circumvent the DRM protection even though the genuine DRM application enforce access right correctly. Currently, a lot of commercial DRM systems are available in the market, such as Microsoft Windows DRM, and Adobe Web Buy. The architecture of these products and some standards such as [14] can generally be described as shown in Fig.3.

- An author generates digital content such as a confidential document.
- A DRM-enabled application will create a rights-protected version of the file and assign usage rights at the content server. A portal server provides a Web site so that the consumer can preview, purchase and download content that interests him.
- A license server will be responsible for the enrollment and certification of trusted entities, and for licensing rights-protected content.
- Finally, the consumer’s DRM client software interacts with the DRM server, and renders the content according to the consumer’s assigned privileges.

Naturally, the general DRM architecture and workflow shown in Fig.3 are applicable in a trusted environment. But it is not straightforward to design TPM-based DRM software in terms of the access right enforcement, DRM application variants and verification. In order to exemplify a DRM application to enable software upgrading over TPM architecture, the following subsections only describes expiry control (Subsection 4.2), DRM software up-gradation (Subsection 4.3) and DRM attestation (Subsection 4.4) respectively.

4.2 Expiry control

Expiry control is used to control the usage of a protected document within a threshold value, typically, the access deadline or access times. In a legacy computer, it is usually very difficult to enforce the expiry control on a protected content offline since a malicious user can change the system states easily. Fortunately, TCG defines a trusted mechanism which is suitable for usage control.

On one hand, TPM has a real-time clock which provides tamper-proof date stamping for authentication and attestation processes. To employ the clock in access deadline control, SPM will check the system clock against the standard
time when the computer owner takes ownership, and then store the correct time into the secure storage. When an application \texttt{App} wants to check the expiry \(T\), from the license, it will read the TPM clock \(T\). If and only if \(T < T_e\), \texttt{App} will grant the user the access to the document.

On the other hand, the content owner admits that the user can render the content for at most \(U_e\) times. Hence, \texttt{App} shall enforce access control on the number of document rendering. Thanks to TPM specification 1.2, TPM has built-in support for monotonic counters which is a tamper-resistant counter embedded in a device whose value, once incremented, cannot be reverted back to a previous value. Hence, the process for enforcing the times control may be:

- \texttt{App} reads the license file and extracts the number \(U_e\) of legal usages.
- \texttt{App} reads the number of counter \(U\) from sealed storage.
- If \(U < U_e\), \texttt{App} renders the protected document. Otherwise, reject the access. If rendering correct, \texttt{App} increases the counter as \(U ← U + 1\).

Note that the low-cost TPM chip can only afford to store four independent monotonic counter values at a time, and only one of these counters is usable during a particular boot cycle, hence we do not have built-in counters in DRM application. Fortunately, Sarmenta and Dijk [15] show how one can implement a very large number of virtual monotonic counters on TPM platform.

4.3 Software up-gradation

Usually, a DRM application \texttt{App} is embedded with a secret \(KEY\) so as to prevent a pirate player from illegal access to protected documents. Furthermore, in order to have long-term security, the producer of application \texttt{App} will periodically issue new versions which contain refresh secret \(KEY\) so that a compromised \texttt{App} is not able to read or to access documents protected by newer versions. Meanwhile, it is highly desirable that documents protected by an old version of \texttt{App} be readable or accessible by its newer versions.

In order to have these features, the secret values in different versions of \texttt{App} are managed as follows.

- Given that there are at most \(V\) versions of application \texttt{App}. Let \(K_v\) be a secret seed and \(\mathcal{H}(\cdot)\) be a one-way hash function.
- The secret value to be embedded in the \(v\)th version of \texttt{App} is
  \[ K_v = \mathcal{H}(K_{v+1}), v = V - 1, V - 2, \ldots, 1. \]

That is to say, the \(v\)th version of \texttt{App} embeds a new secret key \(K_v\). In addition, such an \(v\)th version also includes its own version number \(v\). Afterwards, the producer releases \texttt{App} starting from version 1, then version 2, 3, \ldots, \(V\) sequentially and periodically.

- For each original document \(D\), the content server generates an encrypted document \(D_i = E(K_i, D)\).
- When the \(v\)th \texttt{App} is going to render the protected document \(D_i\), it downloads the protected document as described in Subsection 4.4. Then it requests for a licence which includes the access right from the license server and the earliest version \(i\) of \texttt{App} module which can render the document \(D_i\). If the user has access to the document, \texttt{App} computes
  \[ K_i = \mathcal{H}^{v-i}(K_e), \text{ if } i < v \]
  where \(\mathcal{H}(\cdot) = \mathcal{H}(\mathcal{H}(\cdots(\cdot)))\), \(j > 0\). The \texttt{App} module decrypts the document with the specific key \(K_i\).

4.4 Document delivery process

It is common that the application \texttt{App} has many variants due to versions presented in Subsection 4.3 and patches. Hence, it is inapplicable that a remote verifier can check every variant of an application \texttt{App}. To cope with this diversity, each application and its versions are signed by its producer. That is to say, a software package will be represented as

\[ < \texttt{App}, \text{version } v, \text{SIGN}(\texttt{App}, SK) >, \]

where \(\text{SIGN}(\cdot)\) is a signature on the code \texttt{App}. When an \texttt{App} is loaded, the loader will create its PCR as usual TPM application. Besides, SPM will construct

\[ M = (\texttt{App} \text{ name, producer cert, version } v). \tag{1} \]

When DRM \texttt{App} requests for a protected document from the content provider, the document delivery process is as follows:

- \texttt{App} sends a request to its local SPM. The request message consists of the document name, source address etc. The SPM will send an attestation request to the remote side so as to start an attestation protocol which can be any of the remote attestation ones, e.g., [16, 17, 18].
- The content server sends a challenge nonce \(r\) to the SPM. After receiving the challenge message, SPM returns \(M\) in Eq.(1) and its signature \(\sigma_{AIK} = \text{SIGN}(M, r, AIK)\) to the server.
• The content server will check the authenticity of the signature. If success, it sends the SPM the protected document as well as the document license file (or from license server).

• The App will check the license and access policy so as to recover the document for rendering.

The content server will check the authenticity of the signature. If success, it sends the SPM the protected document as well as the document license file (or from license server).

Figure 4. Document delivery protocol.

5 Discussion

5.1 Security

To build the DRM application beyond trusted platform, we should handle the following security issues:

• Trusted booting: According to TPM booting mechanism, TCG ensures the security of booting sequence such that the kernel including SPM is loaded in a trusted way.

• Anti-abuse attack: Each DRM application is authentically loaded by SPM other than by other loader. Hence, abuse attack can not be mounted. Additionally, since SPM will check the signature of the DRM application App from its producer, App will not be loaded if it is illegally manipulated before loading.

• Anti-replay attack: In the document delivery protocol, the content server forces to perform a challenge-response protocol such that an attacker can not use the old session to download document. Secondly, every document delivery request will be recorded into the database so as to trace the usage of the document. This property is useful in corporate environment.

• Version upgrading: Since the security of the DRM application is based on the secret key, an adversary can forge a new software to obtain all the past and future documents if the key is disclosed. To frustrate this threat, the scheme employs hash chain to refresh the software periodically such that the adversary has to find the software keys of all versions other than exploiting the old keys.

In all, the present scheme provides multiple-layers protection against digital right infringement.

5.2 Performance comparison

Sandhu et al. [19] presented an access control scheme by adding a TRM (Trusted Reference Monitor) module into the user space over a trusted platform. TRM seals secrets and policies so that it can enforce the policies correctly without disclosing the secrets. In comparison, our scheme embeds into the secure kernel a SPM which monitors the process requests. It brings the following advantages:

• Prompt response: Usually, the kernel module has higher priority over process in user space. Hence, SPM can handle the user request promptly.

• Robustness against tampering: SPM residents in the kernel space which is usually tightly protected by the software and hardware. An adversary has to spend much more effort to break the protection since he/she must have the administrator priority to the computer.

• Universal software: TRM handles trusted applications only, it does not mention how to handle the coexistence of legacy applications and trusted applications. On the contrary, SPM can separate trusted applications from legacy ones such that the later can co-exist on the same computer without disturbing the former.

• Software attestation: TRM enforces DRM policy assume that the verifier know the configurations of the prover’s platform. In practice, this assumption is not easy to meet since the verifier usually knows nothing about the prover. In contrast, SPM authenticates the DRM application locally, and asks the verifier to check the suitability of the software versions and producer. In other words, SPM acts as a proxy in attesting the DRM application. As a result, it simplifies the attestation process and broadens the application scope.

6 Conclusion

Although TCG targets for DRM application on trusted platform, it is not trial to implement DRM with TPM. This paper embeds into kernel a software module called as SPM which enables to handle software attestation request locally. This attestation method can handle the security of software updating in a simple but effective way. Moreover, it applies SPM for a DRM application. The present DRM scheme enables that the users can share documents transparently, but enforce usage control with the standard resource in TPM chip.
References


