International Journal of Security and Its Applications
Vol. 11, No. 9 (2017), pp.23-40
http://dx.doi.org/10.14257/ijsia.2017.11.9.03

A Monitor System for Software Version Request in Offline Environments

Shin-Yan Chiou and Chia-Chun Lin

Department of Electrical Engineering, College of Engineering, Chang Gung University; Department of Nuclear Medicine, Linkou Chang Gung Memorial Hospital, Tao-Yuan, Taiwan
ansel@mail.cgu.edu.tw, M0021008@mail.cgu.edu.tw

Abstract

While the Internet allows licensees to easily disseminate digital content, it also facilitates misappropriation of such content along with violations of personal privacy. Digital rights management (DRM) measures were developed to address such problems. However, DRM can only be used to verify data correctness from software, but cannot be used to verify software soundness from data. In addition, although trusted platform modules (TPM) can be used to achieve system security, such measures to not provide DRM protection. We thus propose a version request system not only achieves DRM functions, but can also conduct version requests for software from data. In this system's version checking restrictions, only a secure operation system and application can store the cryptography component's secret key, which is then used to protect and manage the right object (RO).

Keywords*: TPM, DRM, version request, offline*

1. Introduction

Advances in information technology have driven the digitization of many activities and processes. The Internet offers a convenience means of sharing digital resources, overcoming barriers previously imposed by time or distance. Today, out accumulated memories and creative output, in the form of digital files, images, videos, music and animation, are stored in digital media on computers and mobile devices. While this makes these resources easy to share, it also opens them up to misappropriation or privacy violations. For example unauthorized cracked or pirated software may be distributed with malicious modifications or viruses which expose users to potential security issues. The concept of digital rights management (DRM) arose in response to such issues [0].

The Open Mobile Alliance (OMA)'s OMA DRM [[2]] specification provides for the security and management wireless delivery of digital content. This specification ensures that digital content used in mobile communication devices (e.g., mobile phones), complies with object-based authorization to secure the intellectual property (IP) rights of high-value digital content owners. Simply put, OMA DRM can allow digital content providers to provide conditional access to their content while maintaining their ownership rights and right of response. For example, the specification allows for the free pre-screening of DRM content, and restricted delivery based on the security level assigned to the DRM content. The specification supports the development of new business models for DRM content. In other words, OMA DRM provides a protective environment for digital content for the storage, protection and management of digital content objects (CO) for mobile device applications including JAVA, MMS, browsers and email programs.

Received (June 10, 2017), Review Result (August 30, 2017), Accepted (September 4, 2017)

ISSN: 1738-9976 IJSIA

Basically, digital content providers and mobile device users must clearly understand their respective rights and obligations in the distribution and use of downloadable media content. Mobile devices primarily access digital content through downloading, and DRM provides a mechanism by which access to such content can be controlled. DRM allows digital content providers to specify usage rights, and can specify different usage rights for various media objects. Furthermore, providers can set differing prices for different users. Therefore, digital content providers can allow users to share free content previews while applying appropriate pricing for various content objects. Since the object's value is determined by its usage rights rather than by the object itself, DRM can serve as an alternative means of selling media objects, thus putting media availability more in line with actual usage demand while increasing the effectiveness and reasonableness of DRM concepts.

However, the OMA DRM framework relies on using DRM to manage right object (RO) authorizations. However, such a management mechanism has yet to be defined. Therefore, when the DRM agent is checking an RO or a executing a content object (CO), conventional techniques may result in threats [[7]].

Thus, HP, IMB, Intel, Microsoft, and other large, well-known companies have jointly formed the Trusted Computing Group (TCG) [[3],[4]] to specify a secure chip called the Trusted Platform Module (TPM). This chip is installed on motherboards to protect the hardware and software. Some researchers have proposed the use of TPM chips in DRM process, but this raises some problems.

Wu et al. [[5]] proposed enhancing the TPM method and applying it to DRM. However, their method is overly reliant on TPM to ensure security, and fails to consider the computer system platform at the time. Yu et al. [[18]] proposed TBDRM as a way to use TPM to resolve security threats possibly raised by DRM. Their method mainly uses a version controller to achieve secure access through a license, but it still fails to account for system conditions at the time of application execution.

This paper proposes a software version request method which integrates a cryptography component and CBL, using version check such that only secure operation systems and secure applications can access the cryptography component's secret key. The key is used to protect and manage the RO. The proposed method can be used to resolve problems that may arise in implementation. For example, a user can write a program to respond to hacker attacks by verifying the credentials of the operating system or application. If the entire memory is recorded, or if there are operating system errors allowing other programs to access the CO, the version checking can be force upgraded. The same can be accomplished in other types of DRM applications when an error occurs in the operation system or other applications.

This paper presents the following key contributions: (1) It uses a password to integrate the CPL and protect the secret key. (2) It uses a key control management module to check the version of the operation system and application, ensuring that only the secure versions are accessible, thus providing the user with a secure usage environment through DRM. (3) It uses a rights management repository to manage and protect DRMRO rights. (4) It proposes a time management protocol to ensure that digital content rights management can be performed at execution.

The remainder of this paper is organized as follows: Section 2 discusses the research questions. Section 3 introduces the proposed method. Section 4 conducts a system security analysis and comparison. Section 5 shows our implementation result, and conclusions are drawn in Section 6.

2. Related Works

This section reviews the literature relevant to our proposed method, including an introduction of TPM components and their use in DRM, along with related TPM methods.

2.1 TPM

TPM was originally designed by IBM according to the TCG specification, and is a permanent security device which performs password operations on the motherboard with the aim to resolve security issues in computer platform frameworks in the hopes of fundamentally improving credibility. TPM provides cryptographic processing functions including platform integrity, hard disk encryption, key protection and digital signature.

TPM components include I/O, Non-Volatile Storage, Key Regeneration, SHA-1, a calculation engine, Random Number Generation, RSA Engine, Platform Configuration Register (PCR), Attestation Identity Key (AIK), Opt-In, Execute Engine, and Program Code.

TPM is a trusted computing platform, providing computer platform protection from the lowest hardware level [[8]]. The overall system's security verification and secure calls must be completed through TPM, thus establishing a chain of trust from hardware components to operating system to application. This train of trust is transmitted in a way that achieves overall security and integrity checks. In this process, RSA and SHA-1 are the fundamental security calculations for TPM, which uses the RSA encryption algorithm to compare key data for encryption, and typical keys use 1024 or 2048 bit encryption. Due to the limited processing power the components, large amounts of data cannot be encrypted. SHA-1 is used to calculate the measured values for data related to record integrity, while PCR is used to save these measured values. Therefore, PCR must be able to fend off software and hardware attacks.

TPM features many different kinds of keys, all of which use the Endorsement Key (EK) and Attestation Identity Key (AIK) methods. In each instance of TPM, there will be exactly one EK public-private key match provided by the manufacturer. Due to security concerns, EK is not directly used for encryption or signature, and is only used to produce an EK certificate, create an Owner, and generate the AIK certificate. AIK is a public-private key pair established through TPM, with each TPM instance providing one or more AIK pairs. AIK is primarily used to determine the platform's current status and configuration without divulging the platform's identity. TPM uses the AIK pair to measure the system's PCR value for signature. When a platform proves that the other platform is currently in a trustworthy state, it sends the other platform the PCR value of the private AIK key's signature, and the manufacturer-provided certificate of validity to demonstrate that the counterparty is currently a trusted platform.

2.2 Chain of Trust

The transmission of the chain of trust is a key means of establishing credibility. When the system is powered on, TPM will begin to self-check. At this stage, TPM hardware will provide a consistency certificate to certify the legitimacy and integrity of the hardware. Prior to loading the operating system, CRTM code within TPM will conduct an integrity self-test, and then conduct a trust test for BIOS. After this, BIOS is loaded and the system's control privileges are given to BIOS to test the operating system. These tests must be passed to reach the next stage, and the process proceeds level by level to ensure that the entire system is trustworthy. The entire process is illustrated in Fig. 1. Each time the system starts up, the test records are saved to PCR, with measured PCR values summed iteratively to create a hash for storage in PCR. Processing takes place as follows:

$$PCR_{new} = \text{SHA-1}(PCR_{old} + measured\ data) \quad (1)$$

where measured data is the current test value. The prior test values are encrypted and stored in TPM's external storage space.

2.3 Enriched Trusted Platform

Because TPM security is based on measurements of trust, all hardware and software to be subjected to this method must first be verified by TCG on the basis of the manufacturer's initial measurements, and all data must be included in a valid certificate. At the time of use, the measured values and data in the valid certificate are compared to ensure platform integrity. However, there are no clear specifications to determine which type of software should be certified, and which types do not need certification [[6]-[9]]. In other words, there are no specifications for which type of software needs to be seen as trustworthy. For example, at startup, commonly used programs such as Microsoft Internet Explorer may be subject to unexpected threats because they can read, execute and operate binary machine code. This ambiguity may allow an attacker to use a certified software to monitor or modify the internal state of another trusted application.

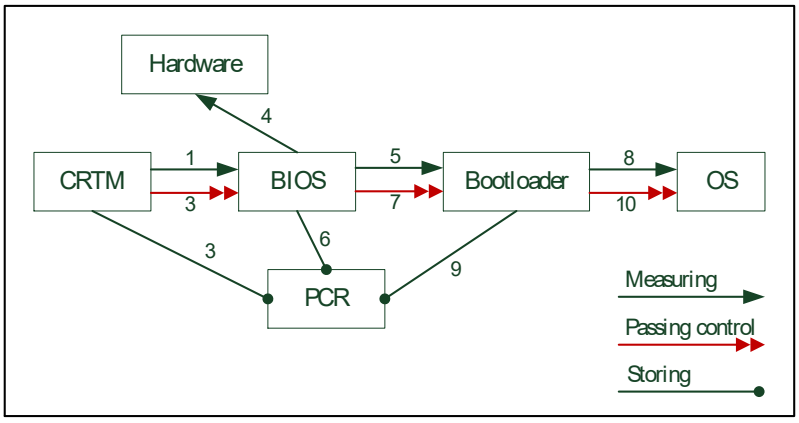


Figure 1. Chain of Trust

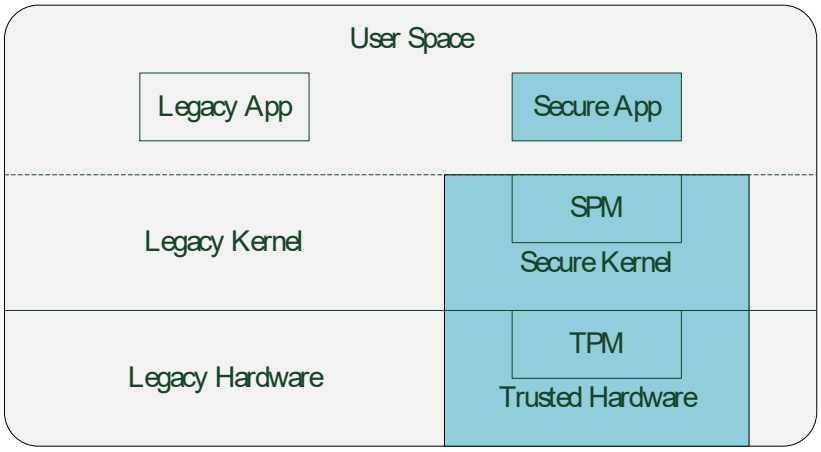


Figure 2. Enriched TPM

Thus Wu and Bao [[5]] proposed an enriched trusted platform for use in DRM. This method includes three levels (see Figure 2) to allow for the update of software operating on the platform. The hardware level is split into trusted hardware and legacy hardware. Trusted hardware includes the TPM chip, memory isolation assembly, and voltage monitors. The security core provides separate execution of applications and related services. They proposed a new element called the secure process manager (SPM), which provides applications and a trusted interface for the security core. When an application is loaded, SPM validates the application's certificate, which is supplied by the application provider. When the provider wants to update the software, it should provide a new

certificate for the new application. Thus, SPM can continuously verify any application. The also proposed a processing method to resolve issues related to expired application usage licenses, as well as methods to transfer updates and files between applications. However, this method still relies on TPM, and fails to account for operating system state at the time of execution. TPM can only check system conditions at boot time, but many users now leave their computers on for extended periods, and changes made between boot ups may not be checked.

Cooper and Martin [[14]] proposed an open framework to provide DRM execution in TPM, allowing licensees to select operating systems and applications without compromising security. However, this framework has no distribution stage, and does not consider protection of usage authorization. Thus attackers can still use a copy of an invalid license, similarly in [[15]-[17]]. Thus Yu *et al.*, [[18]] proposed a DRM framework based on TPM security (TBDRM), which is illustrated in Figure 3. TBDRM uses TPM to resolve security threats which may arise during DRM usage, and uses a version controller to ensure that the license is securely accessed. In the version controller, each DRM is matched with a counter cl to record the number of times a license is used. If the counter value cl is smaller than the value provided in the license, then the application can continue to be used. The version controller is protected by TPM. In other words, if the version controller is operating normally, it can ensure secure access to the license. By creating a license which is secure and can be executed with trust, this approach is resistant to replay attacks, or can reset the system without exposure to expired authorizations. However, this approach does not account for operating system status at the time of application execution.

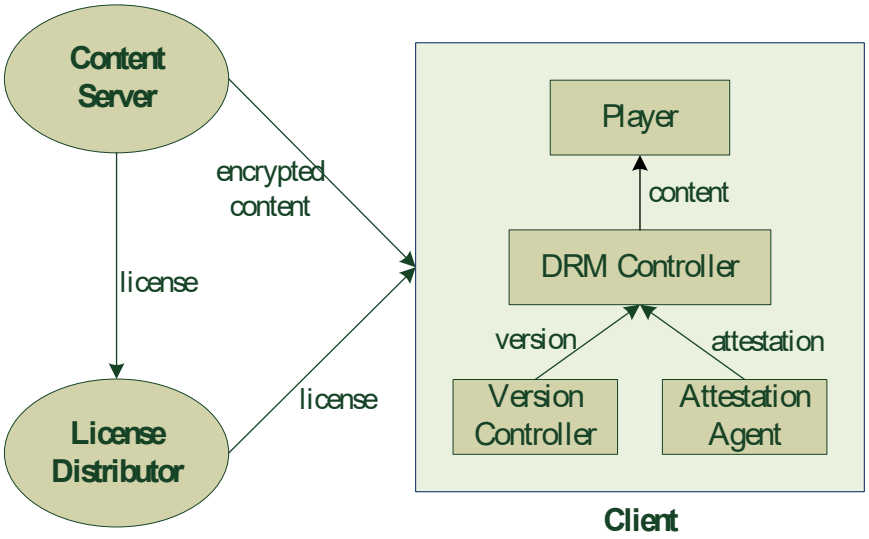


Figure 3. TBDRM System

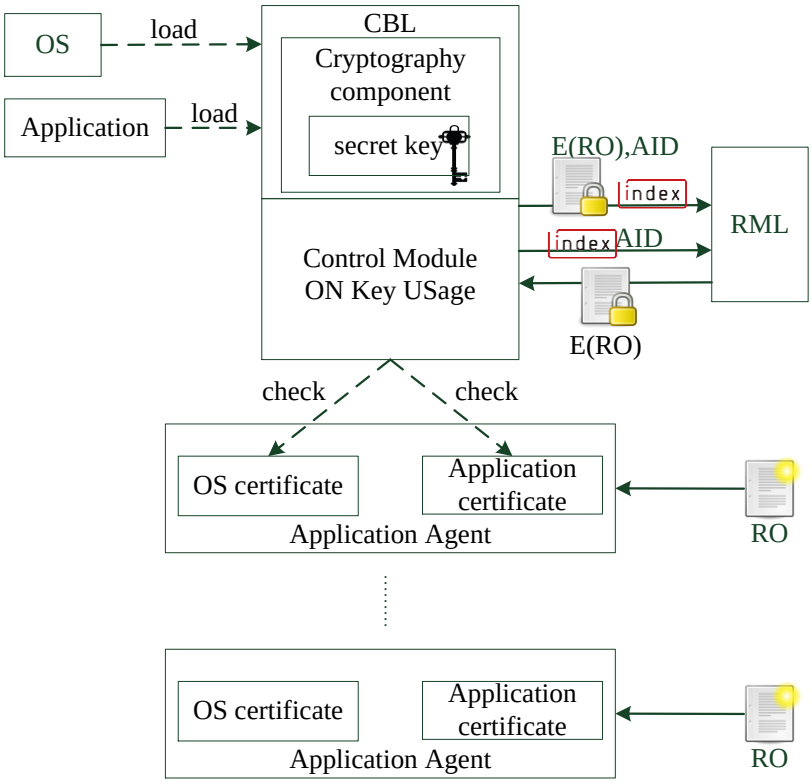


Figure 4. Software Version Request System

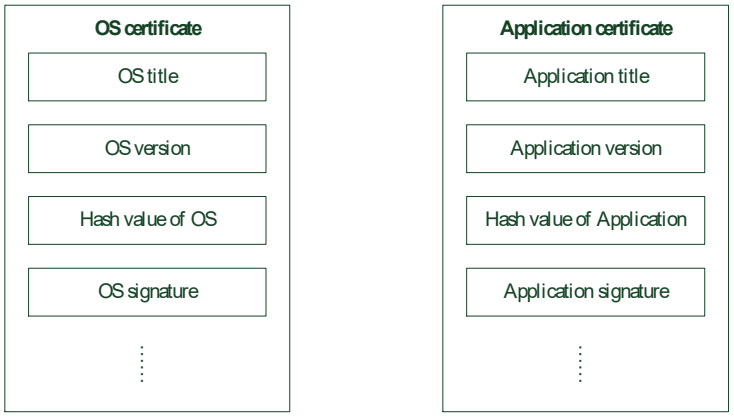
3. Proposed System

We propose a version request system (Fig. 4) which can be used to manage key access and version updates for the management of digital content, and can effectively prevent malicious programs or users from accessing keys. This proposed method provides a secure execution environment for operating systems and applications, and implements DRM. This method combines a cryptography component and Cryptography Boot Loader (CBL) to ensure that only secure operating systems and applications can access the cryptography component's secret key, which is used to protect the accessed DRMRO content. The version request system includes a cryptography component (or CBL), a control module on key usage, and one or more application agents. The control module on key usage can be implemented using software alone, software with the operating system, or hardware with the cryptography component.

The function of this system can be divided into the following five processes: (1) load operating system and application; (2) request the provider's certificate to access RO; (3) version checking; (4) RO protection and management; (5) time control manager.

3.1. Loading Operating System and Applications

In the proposed software version request system (Fig. 4), once the user boots up the device, the cryptography component (or the CBL) will automatically store the public key provided by the OS certificate authority and DRM rights center, and store the user's secret key. The secret key is stored in the cryptography component. Prior to loading the operating system, the control module on key usage will use the public key provided by the



(a) OS Certificate (b) Application Certificate

Figure 5. Certificates

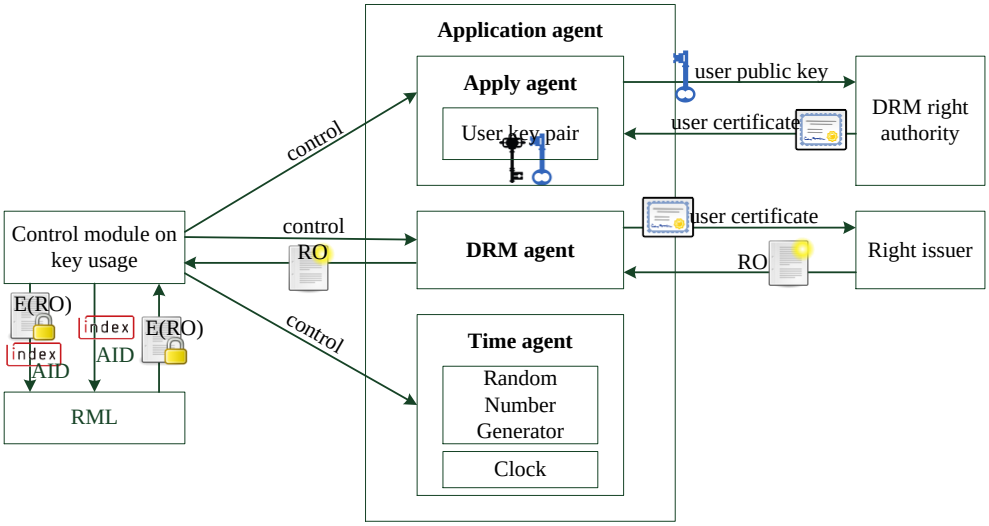


Figure 6. Application Agent

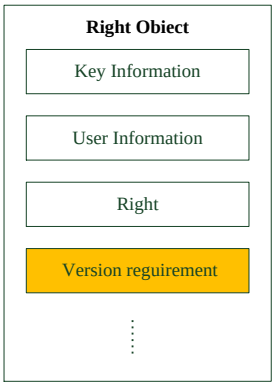


Figure 7. RO Concept

OS certificate authority to check the operating system, and the validity of the OS certificate provided by the OS certificate authority is used to determine whether the operating system can be loaded. If the OS certificate fails inspection, the operating system will not be loaded. Thus, the OS certificate (as shown in Figure 5(a)) is placed in the application agent, and its content includes at least the OS title, OS version, OS hash value and OS signature. During OS version checking, the OS hash value is compared with that of the OS loaded by the user's device.

An application agent may contain three different kinds of agents: an apply agent, time agent and DRM agent (see Figure 6). These three different application agents must have the OS vender's public key issued by the OS certificate authority, along with a certificate signed by the OS vender. The content of this signed certificate (Figure 5(b)) must include the application title, application version, application hash value and application signature. This application signature uses the application to provider's private key to sign the application's hash value or other related data. This application hash value is also compared against the actual loaded application. Before an application is loaded, the control module on key usage will use the public key issued to the OS vender from the OS certificate authority to check the application's certificate validity and thus determine whether the application should be loaded. If the application passes the inspection, it is loaded and given a unique index number (*i.e.*, application index, AID). If the inspection fails, the application is not loaded.

Once the user boots the device, the cryptography component (or the CBL) loads the device's operating system and certificate. If other applications are to be loaded, the control module on key usage is used to conduct a version comparison of subsequently loaded operating systems or applications.

3.2. Requesting User Certification for Right Object

This section explains how to obtain user credentials and how these are used to obtain the right object (RO). Prior to obtaining the RO, the control module on key usage will use the DRM agent to obtain the RO, or it will use the apply agent to request the DRM rights center provide the user's certificate. The user's certificate is an important means of accessing the RO which corresponds to the right issuer (RI). The user can use the apply agent to execute key generation to produce a public key cryptography system key pair made up of a user public key and a user private key. The user public key is used to request the DRM rights center provide the user certificate as follows:

1. Using the DRM right center's public key, the user public key and the user-related data are encrypted and then transmitted to the DRM rights center to request the user certificate for this application.
2. When the DRM rights center receives the request, it will use its own private key to decrypt and check the user-related data. Once the check is successfully completed, it uses this user public key to encrypt the user's certificate.
3. The DRM rights center returns the encrypted user certificate to the apply agent, which then uses the corresponding user private key to decrypt the user certificate.

Once the user certificate is obtained, the user can use the DRM AGENT to obtain the corresponding RO from the RI. This RO includes the CO key information for encryption, and the user-related data, and is used to describe the content usage rights (*e.g.*, number of uses or duration of authorization), and the minimum version information for the operating system and application as requested by the RI (Figure 7).

3.3. Version Checking

Version checking mainly relies on the control module on key usage for implementation. As previously described, before the operating system and application are loaded, the control module on key usage will check the validity of the certificate. The operating system or application will only be loaded if the certificate is found to be valid. Then, the application agent uses the user certificate to obtain the RO from the RI. Only the versions of the operating system and application being loaded are valid. If the minimum version of the operating system and application request is fulfilled, the control module on key usage can store the cryptography component's secret key. The RO uses the secret key for encryption, thus ensuring that only valid versions can successfully implement DRM. When the RO version is updated, the application agent will obtain a new RO from the RI, and then check the version of the operating system and application. If the versions are lower than the RO's requested minimal version, it will force the operating system and application version to perform a version update. If the operating system version or the application version are invalid, or if the loaded operating system and application version do not fulfill the minimum version requirements for the operating system and application, then the secret key cannot be accessed. By forcing the user to update the operating system version or application version, the user is unable to use the new RO.

Errors may occur in either the operating system or application, and upgrades to fix these errors take various forms. For example, the cryptography component can be tied to BIOS, or the control module on key usage can be tied to the operating system. BIOS or the operating system core contains an uncorruptible sector, which is the key sector for DRM. The control module on key usage can execute version checking through a relatively simple code in this sector. In other words, version checking for the operating system and application on the user's device is performed on a specific sector in the operating system core. Because this sector is relatively simple, thus it is unlikely to contain errors. The abovementioned code must be very short (*e.g.*, within several hundred lines of code), to ensure the likelihood of errors occurring in this usage control module remains extremely low. The control module on key usage uses this low-error sector to conduct version check, thus ensuring that, to use the new RO or CO, the user must first upgrade the operating system or application and must obtain the version requested by the new RO.

3.4. RO Protection and Management

We consider the tools potentially used by malicious users to hack the memory RO and thus obtain unauthorized content access. For example, a hacking toolkit called FairUse4WM can use wmv files to dismantle the originally protected license.

To protect the user's privileges from being maliciously falsified, when the user first uses a DRM-protected application, the user certificate is used to access the RO. The control module on key usage then uses the corresponding private key to decrypt the RO to obtain information including the CO's key information and other user-related data, including content usage rights (*e.g.*, usage times or term) and the minimum operating system and application version information as requested by the RI (Figure 7).

The control module on key usage will check whether the operating system version and application version installed by the user meets the minimum versions required by the RO. This check must be successful for the control module on key usage to obtain the cryptography component's secret key. To protect this key, we do not directly encrypt it, but rather use hash it with the application's index to create an encrypted key. This encrypted key is then used to encrypt the RO's key information and rights. This encrypted content is then combined with other data to create a new Rights Management Library (RML) in the RO.

When a user wants to use the DRM-protected application again, the control module on key usage will first send the index corresponding to the application to the RML. The RML then uses this index to find the corresponding RO, and sends this RO to the control module on key usage, which then obtains the encrypted RO. It then checks whether the operating system and application versions on the user's device meets the RO's minimum version requirements. If yes, then the control module on key usage also uses the secret key and the corresponding application index to create a decryption key, which is then used by the RO. We consider how to manage accessing or updating the new RO. When there is a protected new RO, we use the following method to produce an encryption and decryption keys, which are then used to generate other keys.

1. When obtaining RO, the control module on key usage uses the corresponding private key to decrypt the obtained RO content

$$RO = \{key \text{ inf, user inf, right, version req}\} \quad (2)$$

2. The control module on key usage uses RO's version request to check the operating system and application versions. If the check is successful, it can obtain the cryptography component's secret key k_s.

3. Use k_s and the corresponding Application index (AID_i , i =0,1,2) to create an encryption/decryption key

$$k_{i,j} = H^{j+1}(k_s + AID_i), j = 0,1,2,... \quad (3)$$

i is there application index, j is the number of times the RO has been updated, $H^t(.) = H(H^{t-1}(.)), t > 0$ is a hash function.

4. Use the key $k_{i,j}$ to encrypt the RO's key info and rights, along with other data to create a new encrypted RO:

$$E(RO) = \{E_{k_{i,j}}(key \text{ inf} \mid \text{right}), \text{user inf, version req}\} \quad (4)$$

5. Transfer $E(RO)$ and AID_i to the rights management library (RML) to manage the RO and record the RO update iteration.

6. When an application's RO is updated, the control module on key usage will use the previous key to perform a hash to generate a new key. This method is then used for RO encryption and management.

3.5. Time Management

To ensure the digital content rights management can implement time management, one or more application agents can include a time agent to implement time management programs and provide a time matching protocol. The "Time Matching Control Procedure" is provided as an example (Figure 8).

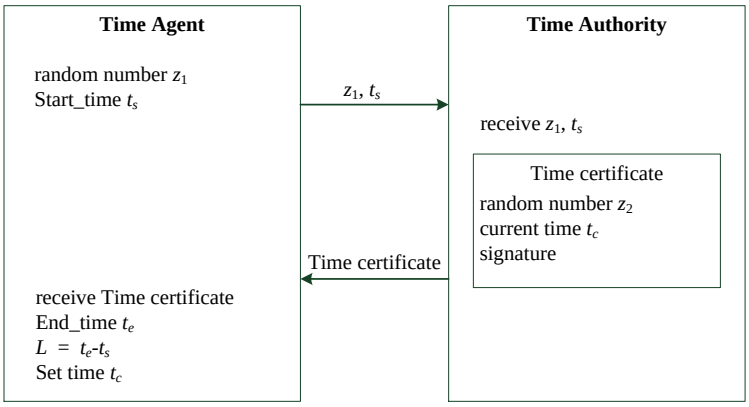


Figure 8. Time-matching Control Process

First, generate a random number z_1 and record the time time_start t_s. Then transmit it to a time authority, which will return a time certificate. This time certificate contains at least a random number z_2, a current time t_c, and the time authority's signature. Record this time as end_time t_e, and calculate the time duration L, where $L = t_e - t_s$. Finally, determine whether this time is the current time t_c on the time certificate. As long as the value of L is smaller than the predetermined threshold (*e.g.*, 1 minute), and if random numbers z_2 and z_1 are consistent, then this clock time is set as the current time t_c on the time certificate. It is worth mentioning that the following methods can be used to determine whether or not to implement time verification. A Flag_TrustedTime is set in the cryptography component (or the CBL). When the machine boots up, if this Flag_TrustedTime is false, then the first Internet connection will start the time agent and implement time matching.

4. System Comparison and Security Analysis

In this section, we compare TPM and the proposed system, and discuss system security.

4.1. System Comparison

Although the proposed system is similar to TPM, but differs in that the proposed system uses data to check the software, while TPM relies on hardware. TPM checks application reliability from the lowest hardware level, checking the hash value of each component against that of the certificate to determine component integrity. If a hardware component or application is suspected of falsification or damage, system operation terminates. While this feature is very powerful, it is also very inconvenient. When a damaged component needs to be replaced, the system must apply for a new reliability certificate. Failure to use pre-validated components will render the system unusable. This is very inconvenient for the user, and will result in a monopoly situation. In addition, TPM can only provide computer platform integrity, but cannot implement DRM encryption and authentication, and is thus unable to separately provide DRM protection.

Our proposed system uses the certificate to determine whether or not to load the operating system and applications. Then DRM can force updates for the operating system and applications based on the versions obtained from the RO, thus providing a secure DRM environment. We consider protection of the DRM right. When obtaining the RO, we ensure that only the control module on key usage can access the RO, thus preventing malicious users from tampering with the RO to gain unauthorized access. Because only

the system can access the control module on key usage, the user is unable to access the key control, thus preventing malicious users from falsifying the RO.

Prior research has proposed security mechanisms for a DRM system implementation environment based on TPM. The system proposed in this paper can provide security without TPM. Table 1 demonstrates the function comparison.

4.2. Security Analysis

We use a cryptography component (or CBL) to protect our secret key, and use a control module on key usage to ensure that the secret key for an encrypted component can only be used for operating systems and applications that are legitimate and meet the RO's minimum version request. Because the secret key is placed within the original password, only the control module on key usage can implement access, and the control module on key usage can only be accessed by the system, not by the user. At boot up, the control module on key usage uses the OS certificate authority's public key to verify the operating system prior to loading, thus ensuring the current legitimacy of the operating system. If the check is successful, the operating system is loaded. Next, the control module on key usage uses the DRM rights center's public key to check the legitimacy of the DRM application. When the user uses the DRM application, the control module on key usage will check the RO's minimum version required for the operating system and application. If these do not match the minimum version system requirements, it will initiate a forced update to ensure the validity of the operating system and application. When accessing the RO, the key control management module will decrypt and then use the cryptography component to protect the secret key, using the application index to produce a key to re-encrypt the RO. Only the control module on key usage can access the cryptography component's secret key.

Furthermore, the key is produced and used within the control module on key usage. No information about the secret key is ever revealed, thus only the control module on key usage can successfully access the RO. Even if a malicious user acquires the content rights management library, the content is encrypted or the index code does not contain any information about the secret key, thus ensuring a malicious user would be unable to modify the content. In addition, our proposed approach can resolve issues potentially arising during the implementation of the RO. For example, when subject to a malicious attack, the system can check the operating system and application's certificate. If the entire memory is recorded, or if an error occurs in the operating system and the CO is accessed by another program, version checking can be used to force an upgrade. When errors occur in other types of DRM applications or operating systems, or when errors occur in other applications, version checking can also be used to force an upgrade.

Table 1. Function Comparison

Description	Enriched TPM [[5]]	TBDRM [[17]]	Proposed method
Key protection	✓	✓	✓
Application validity		✓	✓
Usage right management		✓	✓
Operating system validity			✓
RO protection			✓
Time-matching management			✓
Forced version update			✓

5. Implementation

This section demonstrates the proposed scheme implemented in an environment of one laptop (simulating the user) with two PCs (simulating DRM right center and Time Authority) and provides its flow chart in Figure 9. The selected encryption algorithms are AES and RSA, and the selected signature system is RSA.

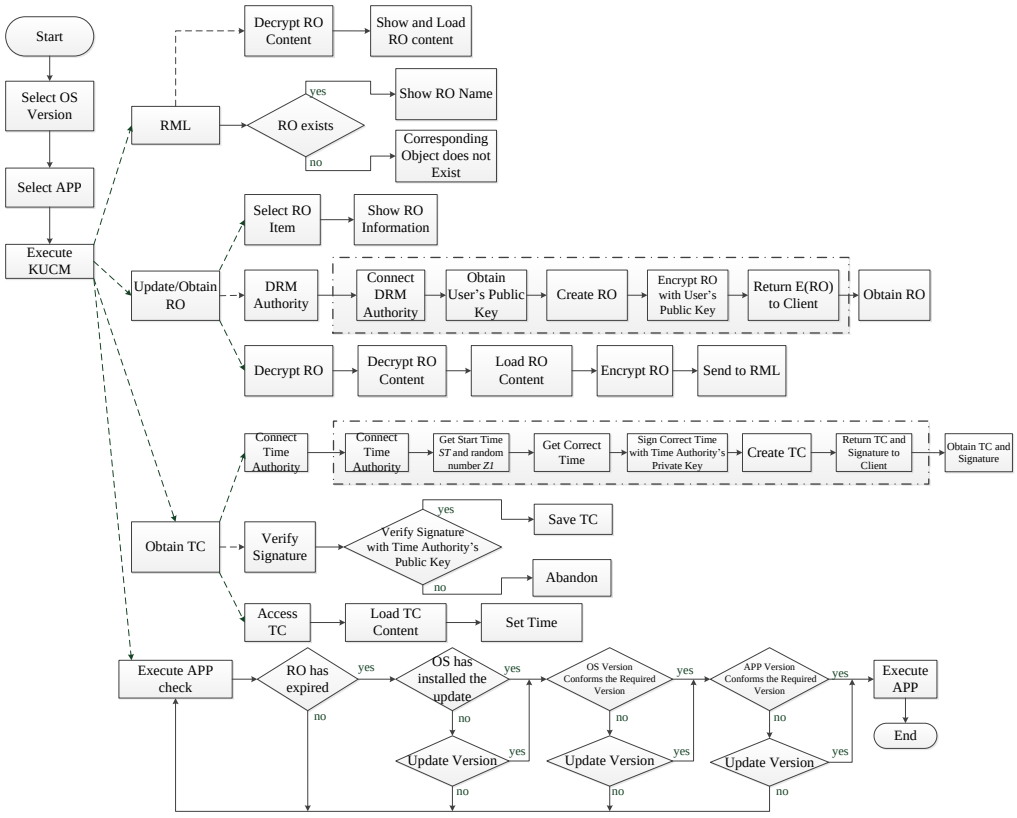
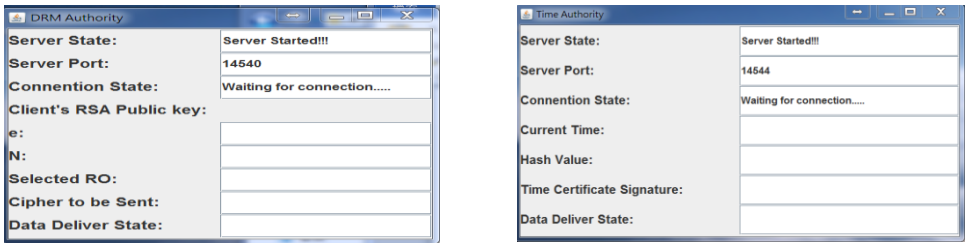


Figure 9. Flow Chart of Our Implementation

(a) A user choose OS version (b) The user choose application

Figure 10. User Choose OS Version and Application



(a) DRM authority is waiting (b) Time authority is waiting

Figure 11. DRM Authority and Time Authority are Waiting for Requests

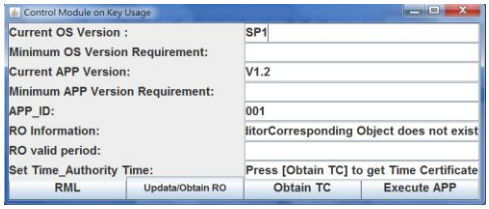


Figure 12. The Interface of Control Module on Key Usage

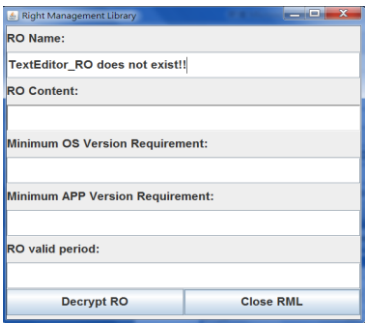


Figure 13. Right Management Library

(a) Information on DRM Agent (B) Select a RO item

Figure 14. DRM Agent

After booting the laptop, a user chooses the OS version (Figure 10 (a)) and then chooses the application (to be controlled) (Figure 10 (b)), while the DRM authority and Time authority are waiting for the user requests (Figure 11). The user can obtain the chosen OS and current application versions from the interface of Control Module on Key Usage, which automatically searches whether right objects of the application exist on the laptop and display the right objects information on the field of "RO information" (Figure 12).

Then the user clicks "RML". If right object does not exist, the field "RO Name" displays "RO does not exist!!", which means the user has to get one right object from the previous interface (Figure 13). Next, the user clicks "Update/Obtain RO" to obtain the information about the user's public key from DRM Agent (Figure 14 (a)), and selects a requested RO item to preview the RO content (Figure 14 (b)). The user then clicks "Obtain RO" to request RO from DRM Authority, which then creates the required RO content, encrypts it via user's public key, and sends the encrypted content to the user (Figure 15).

After receiving the data, the user obtains an encrypted RO content from DRM Agent (Figure 16 (a)), and then clicks "Decrypt RO" to get Decrypted RO content via the decryption using his/her private key (Figure 16 (b)). The user can also see the loaded right object information from the interface of Control Module on Key Usage (Figure 17) and the encrypted RO content from the Right Management Library interface (Figure 18 (a)). Through the "Decrypt RO" button, the decrypted RO can be decrypted and loaded (Figure 18 (b)).

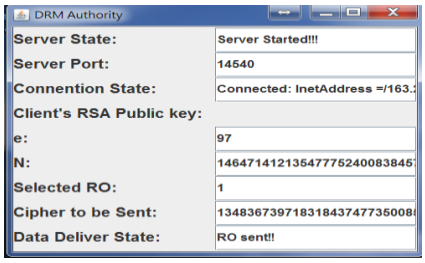
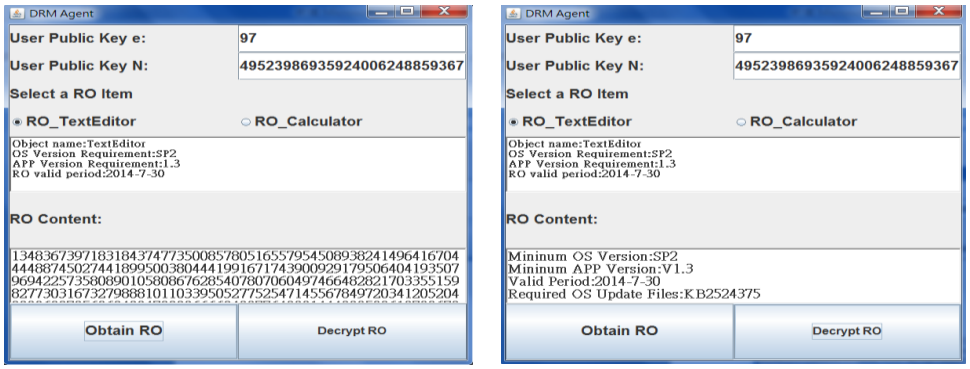


Figure 15. DRM Authority Get a RO Request



(a) Encrypted RO content (b) Decrypted RO content

Figure 16. RO Content in DRM Agent

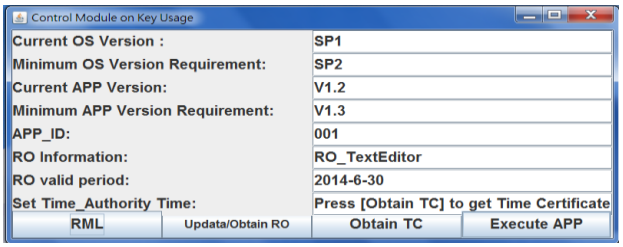
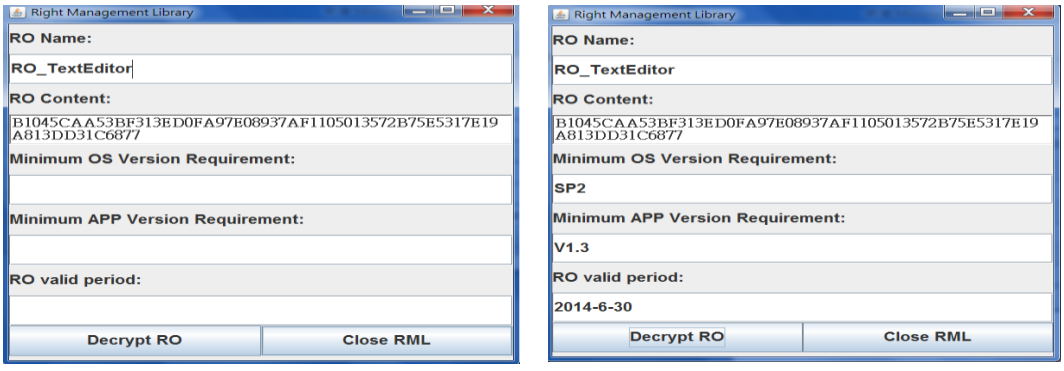
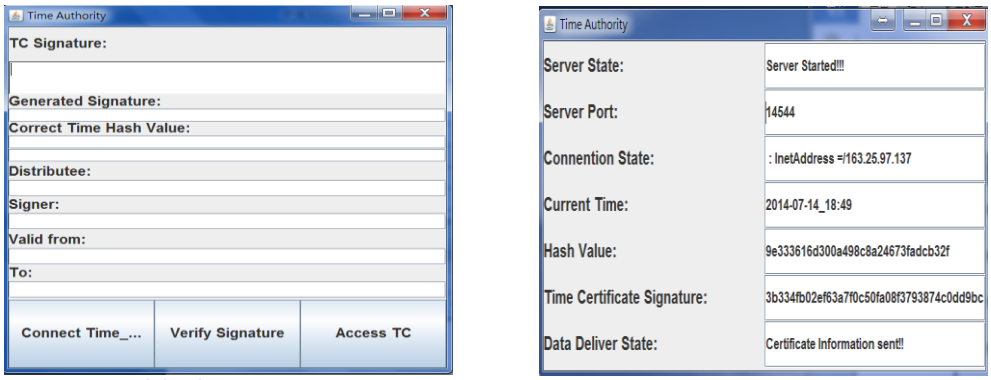


Figure 17. Right Object Information



(a) Encrypted RO content (b) Decrypted RO content

Figure 18. RO Information on Control Module on Key Usage Interface



(a) Time Agent (b) Time Authority get request

(c) Time Agent load in signature (d) Time Agent load in information

Figure 19. Time Agent and Time Authority

After that, the user clicks Time_Authority to enter the Time Agent interface (Figure 19 (a)), and clicks "Connect Time_Authority" to send random a number Z1 and a start time ST to Time Authority. After receiving them, Time Authority gets a correct time, signs them using its private key, and returns the Time certificate and the signature to the Time Agent (Figure 19 (b)).

After receiving them, Time Agent loads in the signature (Figure 19 (c)), uses the Time Authority's public key to verify the signature when the "Verify Signature" button is clicked by the user, stores the Time certificate in the system if the verification is correct, and load in the Time certificate information when the "Access TC" button is clicked (Figure 19 (d)). Control Module then setups the correct time and other information in the certificate (Figure 20), and make a serious checks as follows after the "Execute APP" button is clicked.

(1) Check whether the date of Right object usage exceeds limit of the correct date from system setup. If it does, the user will be alarmed (Figure 21(a)) and has to obtain a new Right object from Time authority before rerunning the application.

(2) Check whether the OS version fits the minimal version requirement or the OS has installed the required suite demanded by Right object. If it does not, the user will be alarmed (Figure 21(b)-(c)) and has to update the OS version or install the required suite before rerunning the application.

(3) Check whether the application version fits the minimal version requirement. If it does not, the user will be alarmed (Figure 21(d)) and has to update the application version before rerunning the application.

If all the above inspections are succeed, the application can be run accurately.

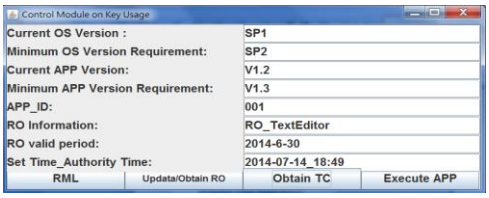
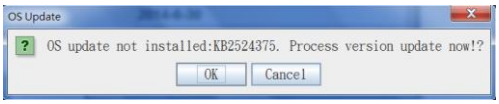
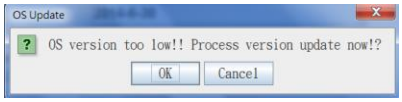


Figure 20. Control Module Setups the Correct time and Other Information

(a) Time expiration check



(b) Specific OS version check



(c) Minimal OS version check

(d) Application version check

Figure 21. Various System Checks

6. Conclusion

In this paper, we propose a version request for a copyright management system to provide a secure environment for the implementation of DRM applications. The approach uses operating system and application certificates to determine the current integrity and legitimacy of the operating system and applications. It then uses an application agent to access the application's RO, and then uses this object to check whether the versions of the operating system and application meet the RO's minimum version requests. If not, it forces the user to initiate an upgrade. Only when the RO's minimum version request is met can the control module on key usage access the cryptography component's secret key. We also consider the protection of the RO usage rights, using the cryptography component's secret key to generate an encryption key to protect the RO. Only the control module on key usage can access the cryptography component's secret key, thus ensuring that malicious users are unable to modify the content. We also use a rights management library to manage the encrypted RO. In addition, we propose a time management method to manage time-restricted access to the digitized content. We also present an analysis comparing the proposed system with TPM, and describe the proposed system's security.

Acknowledgments

This work is partially supported by the Ministry of Science and Technology under Grant MOST 104-2221-E-182-012 and by the CGMH project under Grant BMRPB46. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References

[1] W. Zeng, H. Yu and C. Y. Lin, "Multimedia Security Technologies for Digital Rights Management", Academic Press, **(2011)**.
[2] Open Mobile Alliance, OMA DRM Specification Draft Version 2.0, **(2004)**.
[3] TCG Group and others, "TCG specification architecture overview" , TCG Specification Revision, vol. 1, **(2007)**, pp. 1-24.
[4] L. Chen, R. Landfermann, H. Loehr and M. Rohe, "A Protocol for Property-Based Attestation", Proceedings of the first ACM workshop on Scalable trusted computing, **(2006)**, pp. 7-16.
[5] Y. Wu, and F. Bao, "Enriched Trusted Platform and its Application on DRM", Trusted Infrastructure Technologies Conference, APTC, Third Asia-Pacific, **(2008)**.

[6] Y. Wu, F. Bao, R. H. Deng, M. Mouffron and F. Rousseau, "Enhanced Security by OS-Oriented Encapsulation in TPM-enabled DRM", International Conference on Information Security and Cryptology, **(2007)**, pp. 472-481.
[7] M. Feng and B. Zhu, "A DRM System Protecting Consumer Privacy", Consumer Communications and Networking Conference, **(2008)**, pp. 1075-1079.
[8] T. Eisenbarth, T. G¨uneysu, C. Paar, A.-R. Sadeghi, D. Schellekens and M. Wolf, "Reconfigurable Trusted Computing in Hardware", Proceedings of the ACM Workshop on Scalable Trusted Computing, **(2007)**, pp. 15-20.
[9] A.-R. Sadeghi and C. Stuble, "Property-based Attestation for Computing Platforms: Caring About Properties, Not Mechanisms", Proceedings of the workshop on New security paradigms, **(2004)**, pp. 67-77.
[10] P. Röder, F. Stumpf, R. Grewe and C. Eckert, "Hades-Hardware Assisted Document Security", Second Workshop on Advances in Trusted Computing, **(2006)**; Tokyo, Japan.
[11] Z. Wang, "Application of Configuration Management in Software Project Management", Computer Systems Applications, vol. 17, no. 6, **(2008)**, pp. 101-104.
[12] M. Backes, M. Maffei and D. Unruh, "Zero-knowledge in the Applied Pi-calculus and Automated Verification of the Direct Anonymous Attestation Protocol", Security and Privacy, SP, IEEE Symposium, **(2008)**.
[13] R. MacDonald, S. Smith, J. Marchesini and O. Wild, "Bear: An Open-source Virtual Secure Coprocessor Based on TCPA", Computer Science Technical Report TR2003-471, Dartmouth College, **(2003)**.
[14] J. Marchesini, S. Smith, O. Wild and R. MacDonald, "Experimenting with TCPA/TCG Hardware, or: How I Learned to Stop Worrying and Love the Bear", Computer Science Technical Report TR2003-476, Dartmouth College, **(2003)**.
[15] A. Cooper and A. Martin, "Towards an Open, Trusted Digital Rights Management Platform", Proceedings of the ACM Workshop on Digital Rights Management, **(2006)**, pp. 79-88.
[16] A.-R. Sadeghi, M. Scheibel, C. Stuble and M. Wolf, "Play It Once Again, Sam-Enforcing Stateful Licenses on Open Platforms", 2nd Workshop on Advances in Trusted Computing, **(2006)**.
[17] R. Sandhu and X. Zhang, "Peer-to-peer Access Control Architecture Using Trusted Computing Technology", Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies, **(2005)**.
[18] A. Yu, D. Feng and R. Liu, "TBDRM: A TPM Based Secure DRM Architecture", Computational Science and Engineering, CSE, International Conference, **(2009)**.
[19] D. Mishra and S. Mukhopadhyay, "Privacy Preserving Hierarchical Content Distribution in Multiparty Multilevel DRM", Information and Communication Technologies (WICT), World Congress, **(2012)**, pp. 525-530.
[20] V. Varadharajan and U. Tupakula, "Counteracting Security Attacks in Virtual Machines in the Cloud using Property based Attestation", Journal of Network and Computer Applications, vol. 40, **(2014)**, pp. 31-45.

Authors

Shin-Yan Chiou received the PhD degree in Electrical Engineering from National Cheng Kung University, Taiwan, in 2004. From 2004 to 2009, he worked at Industrial Technology Research Institute as a RD Engineer. Since 2009, he joined the faculty of the Department of Electrical Engineering, Chang Gung University, Taoyuan, Taiwan, where he is currently an Associate Professor. His research interests include information security, cryptography, social network security, and secure applications between mobile devices.

Chia-Chun Lin received the MS degree in Electrical Engineering from Chang Gung University, Taiwan, in 2013. He is fulfilling his mandatory military service in Taiwan. His research interests include information security and secure applications between mobile devices.

