A Cloud-Based User-Friendly DRM System

Suk Ja Lee*, Jing Wang**, Kyung-Hyune Rhee*
*Dept of IT Convergence and Application Engineering,
Pukyong National University
**Dept of Information Security, Pukyong National University

abstract

With the development and rapid growth of cloud computing, lots of application services based on cloud computing have been developed. In addition, cloud-based DRM systems have been developed to support those services' copyright and privacy protection. In this paper, we propose a new cloud-based user-friendly DRM system, which allows users to execute the same contents bought at most *n* times at any devices with license enforcement, which checks the validation of licenses before every execution, having no smart card, which has to carry a smart card reader that seems troublesome to a user, and providing the copyright and privacy protection.

Keywords: Cloud-Based User-Friendly DRM System, Copyright and Privacy Protection, License Enforcement

1. Introduction

With the vast use of the Internet and technology improvements of media streaming and compression, digital contents can be rapidly distributed over the Internet to content consumers. However, if these digital contents distribution is done without any protection and management of digital rights, digital contents can be illegally and easily copied, altered, and distributed to many recipients.

Digital Rights Management (DRM), one of the soluble methods of providing copyright protection and privacy protection, is used to protect the intellectual property of digital contents and to avoid digital piracy from unauthorized users who have not right access control. With the time, a variety of DRM systems have been developed from the different viewpoints according to the required and implemented methods.

In general, DRM systems can be categorized into device-based [3] and smart card-based systems [1]. Device-based DRM comes from enforcing usage of compliant players and unique global device identifiers. This kind of DRM, however, is constrained by its inflexibility, especially in the mobile Internet [1]. For example, a user in real world may change his/her mobile devices, but cannot use the licenses bought before because the new mobile devices cannot pass the verification of the DRM system. To overcome inflexibility of device-based DRM, a smart card-based DRM is proposed. Existing smart card-based DRMs are uneconomic and inconvenient, especially, in the mobile Internet. In addition, each mobile device needs a smart card reader [1]. When a user hopes to access digital content, a smart card reader must be carried and connected to a mobile device.

On the other hand, cloud computing provides services, computation, and storage from a remote and centralized facility or contractor [4]. Data can be easily and ubiquitously accessed in a cloud. One of the most important characteristics is its "pay-per-use" manner, which means that users can rent the corresponding services provided by cloud computing and pay for the actual use of services rather than buy software and physical hardware, which perhaps may be very expensive to buy. Therefore, first of all, users can save software/hardware cost by using cloud services. In addition, by using cloud-based DRMs, they can trust the services of cloud much more, specially in terms of privacy protection.

In this paper, we propose a new cloud-based user-friendly DRM system, which allows users to execute the same contents bought at most *n* times at any devices with license enforcement, which checks the validation of licenses before every execution, having no smart card, which has to carry a smart card reader that seems troublesome to a user, and providing the copyright and privacy protection.

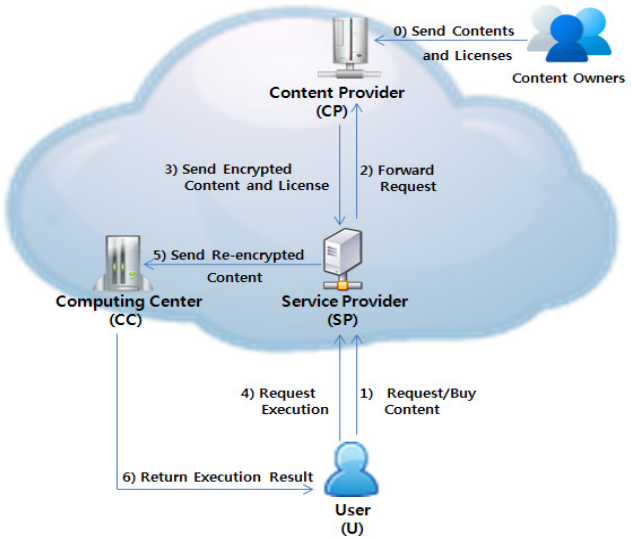
The rest of this paper is organized as follows. We present the proposed DRM system model in Section 2, the system operations in Section 3, and finally conclude the paper in Section 4.

2. System Model

In the proposed DRM scenario, a user requests and buys a content with the corresponding license from a

content provider via a service provider. When receiving a forwarded content request from a service provider, the content provider sends the encrypted content including the corresponding license to the service provider. The service provider stores it for later use. On receiving a command to execute the content from the user, the service provider firstly decrypts the encrypted content and checks the license whether the frequency (e.g. at most n times) field for execution is available or not. If valid, it re-encrypts the encrypted content again and sends it to an arbitrary computing center for execution. If not, it sends an error message to the user. After execution, the computing center sends the execution result to the user.

The architecture of the proposed DRM is shown in Figure 1.



(Figure 1) Architecture of proposed DRM model

The proposed DRM consists of users (U), content providers (CP), service providers (SP), and computing centers (CC). Their roles are as follows.

- Users (U): A user (u) requests and buys a content from a content provider via a service provider. When u wants to execute the content, u sends a partial re-encryption key and a condition key to the service provider. u can execute the same content at most n times specified in the *frequency* field of the license at any devices while the condition of frequency is true.
- Service providers (SP): A service provider (sp) acts as a proxy both between u and a content provider, and between u and a computing center. When receiving a content request from u, sp checks the signature of u. If right, sp forwards the request to a computing center (cc). On receiving a re-encryption key and a condition key from u, sp firstly checks the condition of the license whether the frequency of execution is available or not. If valid, sp sends the re-encrypted content to cc for execution. If not, sp informs an error message to u.
- Content providers (CP): A content provider (cp) has got contents and the corresponding licenses from content owners in advance. On receiving a forwarded content request from sp, cp sends the encrypted content with the corresponding license to sp.
- Computing centers (CC): On receiving a re-encrypted content from a sp, cc firstly decrypts it and executes the content. After execution, cc returns the result to u.

In addition, we make the following assumptions.

- Content providers can have got contents and the corresponding licenses from content owners in advance.
- A certificate authority (CA) issues a key pair, (ssk, ppk), for signature and a public key certificate to each entity.
- Only service providers have got both read and write access rights for licenses.
- Users only have got read access rights for licenses except the first time of buying licenses.
- None of parties involved collaborates to build a user profile.

3. System Operations

The concept of this model consists of three phases: initialization, software-buy, and software-execution phases.

For the proposed DRM, a C-PRE (conditional proxy re-encryption) scheme by Jian Weng and others [2] are used. [2] proved its chosen-ciphertext security in the random oracle model. In addition, to make system operations simple, some notations are used. They are as follows.

- sw: a special software requested from a user.
- l: license of sw. Here, the value of the frequency field in l is a condition.
- ppk/ssk: a public key and a private key generated from a CA for certificate and signature, respectively.
- pk/sk: a randomized (temporary) public key and a private key, respectively, generated from [14].
- $u \in U$, $sp \in SP$, $cp \in CP$, and $cc \in CC$.

- GlobalSetup (λ): The global setup algorithm takes a security parameter λ as input and outputs the global parameters $param$. The parameters in $param$ are implicitly given as input to the following algorithms.
- KeyGen (i): The key generation algorithm takes the user index i as input and generates a public key (pk_i) and a secret key (sk_i) for user U_i.
- ReKeyGen (sk_i, pk_j): The partial re-encryption key generation algorithm takes a secret key sk_i and a another public key pk_j as input and outputs the partial re-encryption key $rk_{i \to j}$. This algorithm is run by U_i.
- CKeyGen (sk_i, w): The condition key generation algorithm takes a secret key sk_i and a condition w as input and outputs the condition key $ck_{i,w}$. This algorithm is run by U_i.
- Encrypt (pk, m, w): The encryption algorithm takes a public key pk, a message m and a condition w as input and outputs the ciphertext C associated with w under pk. Here $m \in M$ where M denotes the message space.
- ReEncrypt $(rk_{i \to j}, ck_i, C_i)$: The re-encryption algorithm takes a partial re-encryption key $rk_{i \to j}$, a condition key $ck_{i,w}$ associated with condition w and a ciphertext C_i under the public key pk_i as input and outputs the re-encrypted ciphertext C_j under the public key pk_j. This algorithm is run by the proxy.
- Decrypt (sk, C): The decryption algorithm takes a secret key sk and a ciphertext C as input and outputs a message $m \in M$ or the error symbol $\perp$.

3.1. Initialization Phase

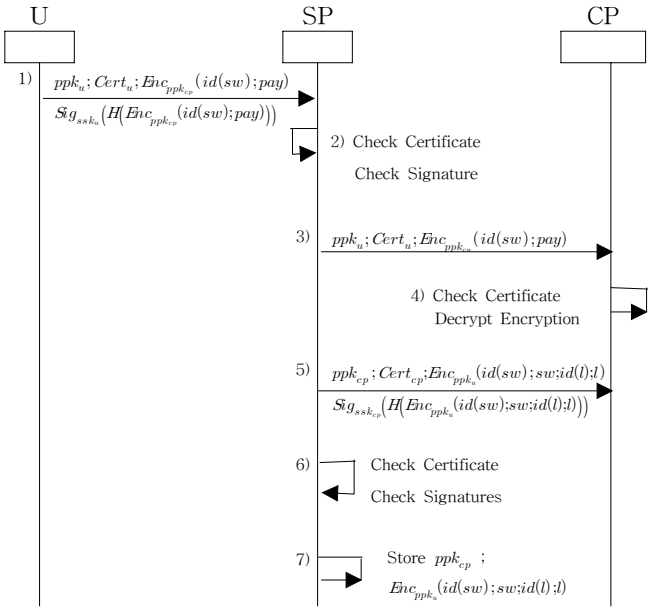
Before starting the DRM, each party has to register with a certificate authority (CA) and generates its own public and private keys (ppk, ssk) respectively. Approved content providers can get contents and the corresponding licenses from content owners at this phase or before this phase.

3.2. Software-Buy Phase

The message sequence chart of the software-buy phase is shown in Figure 2.

The detailed algorithms are as follows.

1) To request and buy sw, first of all, u has to fill fields of the corresponding license up and send the payment. u makes encryption $Enc_{ppk_{cp}}(id(sw);pay)$, where $id(sw)$ indicates which software is requested and pay is the payment for sw. u sends u's public key ppk_u, an authenticated certificate from the CA, $Cert_u$, the encryption $Enc_{ppk_{cp}}(id(sw);pay)$ and u's signature on it ,$Sig_{ssk_u}\left(H\left(Enc_{ppk_{cp}}(id(sw);pay)\right)\right)$, to sp. Here, H is a hash function for $Enc_{ppk_{cp}}(id(sw);pay)$. If u already has got the license of sw in the sp and has still got the frequency for execution, u goes directly to the software-execution phase



(Figure 2) Software-Buy Phase

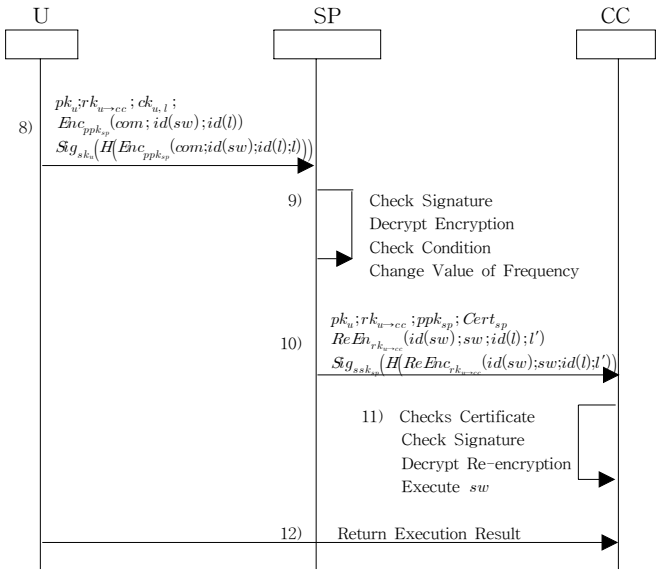
2) On receiving them, sp checks the certificate and signature of u.
3) If valid, sp forwards $ppk_u; Cert_u; Enc_{ppk_{cp}}(id(sw);pay)$ to cp. If invalid, sp informs an error message to u.
4) cp checks u's certificate $Cert_u$. If valid, cp decrypts $Enc_{ppk_{cp}}(id(sw);pay)$ under its private key ssk_{cp}.
5) cp checks the fields of the license l, e.g. whether the pay is right or not and so on. If valid, cp makes an encryption for sw associated with condition of the corresponding license l under u's public key ppk_u, $Enc_{ppk_u}(id(sw);sw;id(l);l)$. cp also makes a signature on it $Sig_{ssk_{cp}}\left(H\left(Enc_{ppk_u}(id(sw);sw;id(l);l)\right)\right)$. cp sends them with cp's certificate $Cert_{cp}$ and cp's public key ppk_{cp} to sp.
6) sp checks the certificate of cp. If valid, cp checks the signature under its private key ssk_{sp}.
7) If valid, sp keeps $Enc_{ppk_u}(id(sw);sw;id(l);l)$ to its storage for later use.

3.3 Software-Execution Phase

Figure 3 shows the message sequence chart of the software-execution protocol.

The detailed algorithms are as follows.

8) When u wants to execute sw, first of all, for execution, u's randomized (temporary) public and private key pair (pk_u, sk_u), a randomized partial re-encryption key $rk_{u\to cc}$ and a randomized condition key $ck_{u,l}$ are generated by KeyGen(), ReKeyGen() and CKeyGen() of [14] respectively. u sends u's public key pk_u, a partial re-encryption key $rk_{u\to cc}$, which is shared between u and cc, and a condition key $ck_{u,l}$, which checks the condition of the license l of u, an encrypted command under sp's public key ppk_{sp}, $Enc_{ppk_{sp}}(com;id(sw);id(l))$, where com is an execution command, and a signature under u's private key sk_u, $Sig_{sk_u}\left(H\left(Enc_{ppk_{sp}}(com;id(sw);id(l))\right)\right)$.



(Figure 3) Software-Execution Phase

9) On receiving them, sp checks firstly the signature of u. If valid, sp decrypts the encryption received under its private key ssk_{sp}. Before execution, sp checks the condition of l under the condition key $ck_{u,l}$, which checks u's license l whether the condition is right or not. If valid, sp changes the value of l's frequency field, e.g. from n times to $n-1$ times, and marks the timestamp field of l with current time, giving a changed license l'

10) sp re-encrypts the stored encrypted software into $ReEnc_{rk_{u\to cc}}\left(Enc_{ck_{u,l}}(id(sw);sw;id(l);l)\right)$ under the partial re-encryption key $rk_{u\to cc}$, giving $ReEnc_{rk_{u\to cc}}(id(sw);sw;id(l);l')$. Then sp sends it, the signature on it, the certificate $Cert_{sp}$, the u's public key pk_u and the partial re-encryption key $rk_{u\to cc}$ to cc.

11) On receiving them, first of all, cc checks the certificate of sp. If valid, cc checks the signature under its private key ssk_{cc}. If valid, cc decrypts the re-encrypted software under the partial re-encryption key $rk_{u\to cc}$, which is shared by between u and cc. And then cc executes sw.

12) Finally, cc returns either the execution result or an error message to u.

4. Conclusion

We proposed a new cloud-based user-friendly DRM system employing no anonymity payment scheme and a smart card, which have more complex computation and cost overhead. Our proposed DRM can be more user-friendly compared with DRMs having smart cards, which have to carry smart card readers that seem troublesome to users.

Even though we proposed a cloud-based user-friendly DRM system here, it still includes some inconvenient problems. Therefore, we hope that we study on it further and improve it to be more user-friendly in the near future.

References

[1] H. Sun, C. Hung, and C. Chen, “An improved digital rights management system based on smart cards,” in Proceedings of the Inaugural IEEE-IES Digital EcoSystems and Technoligies Conference (DEST ‘07), pp. 308-313, 2007.

[2] J. Weng, R. Deng, X. Ding, C. Chu, and J. Lai, “Conditional proxy re-encryption secure against chosen-ciphertext attack,” In ASIACCS, pp. 322-332, 2009.

[3] L. Wang, and C. Wang, “A DRM System for Mobile Digital Journals Based on OMA DRM Model,” IEEE International Conference on Electronic Engineering and Information Technology, pp. 2310-2313, 2011.

[4] S. Palavalli, U.Srinivas, A.Pais. “Identity Based DRM System with Total Anonymity and Device Flexibility using IBES,” Proceedings of the 2008 High Performance Computing & Simulation Conference ⓒEMCS, Waleed W. Smari(Ed.), 2012.