Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2011, Article ID 837209, 19 pages
doi:10.1155/2011/837209

Research Article

CS-DRM: A Cloud-Based SIM DRM Scheme for Mobile Internet

Chaokun Wang,[1, 2, 3] Peng Zou,[1] Zhang Liu,[4] and Jianmin Wang[1, 2, 3]

[1] *School of Software, Tsinghua University, Beijing 100084, China*
[2] *Key Laboratory for Information System Security, Ministry of Education, Beijing 10084, China*
[3] *Tsinghua National Laboratory for Information Science and Technology (TNLIST), Beijing 100084, China*
[4] *Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*

Correspondence should be addressed to Chaokun Wang, chaokun@tsinghua.edu.cn

Received 2 June 2010; Revised 8 September 2010; Accepted 26 October 2010

Academic Editor: Damien Sauveron

With the rapid development and growth of the mobile industry, a considerable amount of mobile applications and services are available, which involve Internet scale data collections. Meanwhile, it has a tremendous impact on digital content providers as well as the mobile industry that a large number of digital content have been pirated and illegally distributed. Digital Rights Management (DRM) aims at protecting digital contents from being abused through regulating their usage. Unfortunately, to the best of our knowledge, fewer of these DRM schemes are concerned with the cost of the servers in a DRM system when the number of users scales up, and consider benefits of content providers who can be seen as tenants of a content server. In this paper, we propose CS-DRM, a cloud-based SIM DRM scheme, for the mobile Internet. The SIM card is introduced into CS-DRM to both reduce the cost and provide higher security. Also, the characteristics of cloud computing enable CS-DRM to bring benefits for content providers, and well satisfy the performance requirements with low cost when the number of users increases significantly. Furthermore, we have implemented a prototype of our DRM scheme, which demonstrates that CS-DRM is efficient, secure, and practicable.

1. Motivation

With the rapid development and growth of the mobile industry, a considerable amount of mobile applications and services are available, which involve extremely large, Internet scale data collections, for example, image, e-books, audio, and video. As a result, users are capable of sharing and distributing digital content easily through mobile Internet. However, a large number of digital content have been pirated and illegally distributed, which has a tremendous impact on digital content providers as well as the mobile industry.

Digital Rights Management (*DRM*) is a mechanism which protects digital content from being abused through regulating its usage. *A DRM scheme* is a model of DRM and consists of related components, protocols, and algorithms (the formal definition of a DRM scheme is given in Section 2.2.1). *A* DRM system is an implementation of a *DRM scheme*. In the context of a DRM system, only an authorized user, who has obtained a license, can access the digital content according to the rights information defined in the license.

Current DRM schemes can be summarized into the following two categories.

Device-Based DRM. In a device-based DRM scheme, such as OMA (Open Mobile Alliance) DRM [1], and Microsoft DRM [2], the security comes from mandatory usage of customized players (e.g., Windows Media Player 10 Mobile for Pocket PC) and unique global device identifiers (e.g., IMEI—International Mobile Equipment Identifier). However, this kind of DRM scheme is constrained by its inflexibility, especially in the mobile Internet. For example, a user, in reality, may change or switch her/his mobile devices, but cannot use the licenses bought before because the new mobile device fails to pass the verification of the DRM system.

Smart Card-Based DRM. In order to overcome the inflexibility of device-based DRM schemes, the smart card-based DRM scheme is proposed. A "smart card" is a chip card with storage, identification, and encryption/decryption capabilities [3] and is the core component of the scheme. The security of this kind of DRM scheme rests on key generation-related algorithms and protocols protected by the smart card. However, there are security issues, such as imposter attacks [4, 5], in some existing smart card-based DRM schemes. Meanwhile, some solutions (e.g., [6]) are proposed to solve these issues, but the complexity and cost of this kind of DRM schemes dramatically increase. Especially, existing smart card-based DRM schemes are uneconomic and inconvenient in the mobile Internet. In detail, besides a smart card, each mobile device needs a smart card reader [5, 6]. When a user wants to access digital content, a smart card and a smart card reader have to be carried and connected to a mobile device.

Taking note of shortcomings of the smart card-based DRM scheme, for the mobile Internet, we propose SIM DRM (SIM card-based DRM system) in Section 2.2.2, which uses a SIM card instead of a smart card. It overcomes the deficiency of existing smart card-based DRM schemes.

According to the investigation report on the China mobile Internet and 3G networks by CNNIC [7], the number of mobile Internet users has reached 155 million in China by late June 2009, 46% of the mobile phone users, and with the proportion of mobile Internet users increased by 6.5% within six months. A large amount of mobile media, such as mobile video and music, are downloaded or enjoyed online. On one hand, large numbers of mobile content providers emerge, such as Arphiola and telecomCONTENT. Specifically for the domain of DRM, these content providers may upload digital content with different formats using divergent channels to the content server. However, two pivotal requirements need to be concerned by both content providers and the content server. First, the content server of a DRM system can only deal with and encrypt some certain formats of contents so that these contents can be used under the framework of a DRM scheme. Preliminary works for contents, such as content editing and format conversion, have to be done by content providers themselves. However, it is clearly a burden for content providers. Second, as a tenant of the content server, each content provider uploads digital content to the content server and derives benefits from its contents being consumed by users. Since contents of each content provider are all stored and handled in the same place, the data security, sharing, and isolation among content providers becomes a rather important issue. Fortunately, the cloud environment is one of the best solutions. In a cloud, the content server provides rich and powerful services for handling contents. Each content provider only needs to use these services to do complex preliminary works without any extra cost. Meanwhile, the virtualization technology used above the infrastructure of the cloud guarantees the data security, sharing, and isolation among tenants.

On the other hand, while the stable growth of the mobile Internet active users brings huge economic interests, the performance of the mobile applications and services becomes more and more important. The low access speed and long response time have been the most important factors limiting the development of the mobile Internet, which is voted by 55% of mobile Internet users [7]. Specifically in a DRM system, with the number of active users scaling up, mass of the data requests and data operations place a heavy burden on the DRM system. The capabilities of computation, storage as well as the performance of data management system are main constraints on the performance of the whole DRM system. The existing DRM schemes are still required to purchase huge amount of equipment and perform maintenance, which is a large investment and takes high daily expenses, when the number of user visits increases. The huge capability of computation and storage of the cloud environment makes the cloud one of the best solutions for satisfying performance requirements of the entire DRM system when the number of user visits grows to infinity.

The above situations call for a new DRM scheme which is low-cost, flexible, secure, efficient, and practicable. In addressing this problem, we propose a novel DRM scheme, called CS-DRM, which is a Cloud-based SIM DRM scheme for media protection in the mobile Internet. In summary, our main contributions are as follows.

(i) We propose CS-DRM, a 4-tuple model, in which the usage of a SIM card instead of a smart card not only reduces the unnecessary cost, but also provides higher security. In addition, the cloud computing is introduced in the scheme to provide more efficient and higher quality services.

(ii) We propose a practicable use case for our CS-DRM scheme. It is a concrete process which consists of five phases, that is, preparation phase, rights customization phase, license acquisition phase, play phase, and download/upload phase.

(iii) We have implemented a prototype of our proposed scheme, called Phosphor, which demonstrates that CS-DRM is efficient, secure, and practicable.

The rest of this paper is organized as follows. Some preliminaries are introduced in Section 2. We detail the CS-DRM scheme in Section 3. In Section 4, we focus on the CS-DRM use case. Security issues and system characteristics of the CS-DRM are analyzed and discussed in Section 5. The implementation details of Phosphor are stated in Section 6. Section 7 reports the results of our experiments and analyzes the effectiveness and efficiency of our scheme. The related works are reviewed in Section 8. Finally, we conclude this paper in Section 9.

2. Preliminaries

In this section, we state the preliminaries of CS-DRM. First, we discuss differences between a SIM card and a smart card. Second, we elaborate on the concepts of a common DRM scheme, SIM DRM, and cloud computing. Finally, we give some fundamental statements and explain the symbols used in the rest of the paper.

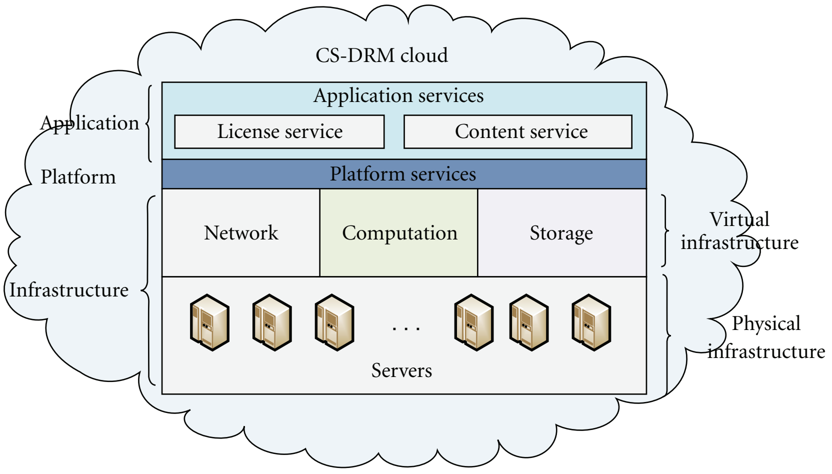


Figure 1: The architecture of a cloud.

2.1. Differences between a SIM Card and a Smart Card. A smart card follows the Standard ISO 7816 [3] which defines the physical and electrical characteristics of smart cards. A SIM card is produced according to the specification GSM 11.11 [8] and GSM 11.14 [9] besides the standard ISO 7816. GSM 11.11 defines the interface between the SIM card and the mobile device as well as the interaction between the SIM card and the mobile operator. GSM 11.14 describes the SIM Application Toolkit which is a set of applications and related procedures used in a GSM session.

A SIM card distinguishes itself by the following three points. Firstly, a SIM card has the proactive SIM mechanism [9] which allows a SIM card to send proactive commands. In this way, a SIM card can do many things that normal smart cards could not do, such as proactive information synchronization between the SIM card and the mobile operator. Secondly, the $A3$, $A5$, $A8$ and $A38$ algorithms [8] embedded in the SIM card can help SIM card authenticate the subscriber, encrypt/decrypt data and generate a secret key. Finally, a SIM card owns a secure file system which has a complete access control mechanism [3, 8]. All these characteristics make the SIM card more efficient and secure than the normal smart card.

2.2. Concepts

2.2.1. DRM Scheme. Formally, a DRM scheme is a 3-tuple, shown as follows:

$$\text{DRM} = (\mathcal{C}, \mathcal{P}, \mathcal{A}), \tag{1}$$

where $\mathcal{C}$ is a set of main components, $\mathcal{P}$ is a set of protocols which are in charge of communication and data transmission among components, and $\mathcal{A}$ is a set of auxiliary algorithms in the scheme. However, the denotation of each tuple could be different depending on different schemes. Taking the OMA DRM scheme as an example, $\mathcal{C}$ contains five components—DRM Agent, Content Issuer, Rights Issuer, User and Off-device Storage; ROAP and other protocols form $\mathcal{P}$; $\mathcal{A}$ consists of kinds of hash algorithms as well as symmetric and asymmetric encryption/decryption algorithms.

2.2.2. SIM DRM. As the name suggests, a SIM DRM scheme, abbreviated as SIM DRM, is a DRM scheme based on the SIM card. Formally, SIM DRM is also a 3-tuple $(\mathcal{C}, \mathcal{P}, \mathcal{A})$ where a SIM card, as a component, is added to $\mathcal{C}$. Meanwhile, some protocols and algorithms specially designed for SIM DRM are added to $\mathcal{P}$ and $\mathcal{A}$, such as the LSWP protocol and the License-Generation algorithm presented in sequel.

2.2.3. Cloud Computing. The cloud computing provides services, computation, and storage from a remote and centralized facility or contractor [10, 11]. In a cloud, data can be easily and ubiquitously accessed. One of the most important characteristics of cloud computing is its *pay-as-you-go* manner. It means that users only need to rent corresponding services provided by cloud computing and pay for the actual utilization of services, rather than buy software and physical hardware which users may consider too expensive.

A cloud system is a system implementing cloud computing. Without ambiguity, a cloud system is abbreviated as a cloud in the rest of the paper. A cloud refers to not only application services delivered over the Internet, but also the hardware and system software in the system. As shown in Figure 1, the typical architecture of a cloud consists of three layers. The infrastructure layer includes the physical infrastructure and the virtual infrastructure. The former is composed of tens of thousands of commercial machines; the latter contains network, computation, and storage, which communicates with physical servers by unified interfaces. In the cloud, the infrastructure can also be seen as a service provided to customers. Some businesses are based on infrastructure services such as Amazon. The platform layer, for example, Google Application Engine (GAE), provides platform services to develop new applications. Above the platform layer, all kinds of custom application services comprise the application layer. For example, in the CS-DRM scheme described later, the license service and content service are established within the application layer.

Any client connected to a cloud, via wired or wireless means, is called a cloud client. Correspondingly, the cloud is also called the backend.

2.3. Fundamental Statements. As known to all, SIM cards are issued by mobile operators which have mobile networks and a large number of users. We have some fundamental statements as follows.

2.3.1. Trust in the Mobile Operator. The mobile operator is trustworthy. If the mobile operator fabricates facts or divulges user privacy so that the interests of mobile users are undermined, the mobile operator will lose the reputation, credit and market, which the mobile operator could not afford.

2.3.2. Cooperation with the Mobile Operator. It is of practical significance to cooperate with the mobile operator. First, a SIM card issued by the mobile operator is of vital importance for a client. It is the foundation of algorithms and protocols for the security of CS-DRM. Since SIM cards used in CS-DRM are provided by the mobile operator, we need to cooperate with the mobile operator naturally. Second, the servers of CS-DRM need corresponding algorithms and data from the mobile operator, such as $A5/A8$ algorithms and K_i of the SIM card [8]. Please note that K_i must be safe, which is a secret key stored in the SIM card and the mobile operator. It only can be acquired from the system of the mobile operator by a special interface. For the privacy and the security of this acquisition process, the hash value of the International Mobile Subscriber Identity (IMSI) is required as a parameter of the interface. Third, the mobile operators have the mobile network and huge number of users which are two crucial factors of a successful business system of CS-DRM.

For the convenience of presentation, some symbols used in the remainder of the paper are listed in Table 1.

3. The CS-DRM Scheme

In this section, we propose the CS-DRM scheme which is different from existing DRM schemes. The definition of the CS-DRM scheme is presented at first. Then, we elaborate on its elements one by one. The original idea of the CS-DRM scheme is presented in [12].

3.1. Definition. A CS-DRM scheme, abbreviated as CS-DRM, is a 4-tuple

$$\text{CS-DRM} = (\mathbb{E}, \mathbb{S}, \mathbb{P}, \mathbb{A}), \tag{2}$$

where $\mathbb{E}$ is the set of entities in the cloud client, $\mathbb{S}$ the set of application services based on cloud computing, $\mathbb{P}$ the set of protocols among $\mathbb{E} \cup \mathbb{S}$ ($\mathbb{E} \cup \mathbb{S}$ denotes all elements in both $\mathbb{E}$ and $\mathbb{S}$), and $\mathbb{A}$ the set of auxiliary algorithms used in $\mathbb{P}$.

Users of a system implementing the CS-DRM scheme utilize or enjoy digital content by a frontend, that is, a cloud client, after purchasing corresponding licenses. The backend of the system contains kinds of application services based on cloud computing, which are in charge of generating, storing and transmitting encrypted digital content and corresponding licenses.

Because the SIM card and cloud are introduced into CS-DRM, the roles and functions of the client and backend are more distinguished. In our scheme, we use $\mathbb{E}$ and $\mathbb{S}$ instead of $\mathfrak{C}$ defined in Section 2.2.1 so that characteristics of these two parts can be expressed clearly, such as the better security provided by the SIM card, and the elasticity, significant economic advantages [11] as well as multitenant nature brought by the cloud.

Table 1: Symbols.

Symbol	Definition
K_i	A secret key stored in the chips of SIM card. In our scheme, its copy is also stored in the storage of license service.
$A3(\cdot)$	An authentication algorithm for authenticating the subscriber.
$A5(\cdot)$	An algorithm for enciphering/deciphering data. The inputs are K_c and data.
$A8(\cdot)$	An algorithm for generating K_c. K_c is the cryptographic key used by the $A5$ algorithm. The inputs are K_i and a random number.
$A38(\cdot)$	A single algorithm performing the functions of $A3$ and $A8$.
K_{cek}	A key by which the content service encrypts the digital content.
UK	A key by which the license server encrypts K_{cek}.
EK	The encrypted K_{cek}.
$A_{UK}(\cdot)$	An algorithm used to generate UK.
$E_k(\cdot)$	A symmetric encryption algorithm with key k.
$DE_k(\cdot)$	A symmetric decryption algorithm with key k.
$H(\cdot)$	A public one-way hash function.
$GetKI(\cdot)$	A function for fetching K_i in the license service.
$\|\|$	Concatenation operation.

3.2. Entities. The cloud client of CS-DRM has four main entities: a SIM card, a DRM agent, a custom player, and a CS-DRM compliant browser.

SIM Card is the Subscriber Identity Module card used in the mobile device [8]. At the client, besides the network operations such as identifying a subscriber of the mobile network, the SIM card is in charge of security issues, for example, authentication between the cloud client and the license service, verification for license integrity and generating K_{cek}, as shown later in Algorithm 4. It stores a secret key K_i and a unique IMSI. The $A3$, $A5$, $A8$, and $A38$ algorithms embedded in a SIM card are implemented and protected by hardware.

DRM Agent is a “bridge” among other entities, and consists of a set of logical rules of cloud clients in the device. For example, if the client wants to render the encrypted digital media when it has acquired the corresponding license, the DRM agent will send an APDU (Application Protocol Data Unit) command to the SIM card for K_{cek}. After that, the DRM agent decrypts encrypted digital media by K_{cek} and then sends it to the compliant player for rendering. Here K_{cek} is generated by decrypting EK using UK, and more details will be presented later in Algorithms 3 and 4.

Custom Player is for rendering digital media. The player could not illegally copy or distribute the decrypted digital media when it is rendering the content.

CS-DRM Compliant Browser is for browsing web sites of the backend. It notifies the DRM agent to do the next action when it receives corresponding responses or events.

3.3. Application Services. As shown in Figure 1, the backend of CS-DRM provides cloud clients with two application services within the application layer. The implementation of the platform layer and infrastructure layer is not considered in this paper because it is out of the scope of this work.

Content Service deals with all content related requests from clients and provides high-quality services to content providers for dealing with contents. It encrypts digital content with the key K_{cek} and hosts distribution channels, such as a web site used to search TV menus. Content service will transmit K_{cek} to the license service for generating licenses.

License Service is in charge of all license-related requests, such as dealing with SIM card authentication and generating, transmitting licenses to the DRM agent according to rights, K_{cek} and other necessary information, as shown in Algorithm 2. The rights information is the usage rules of the content.

These application services in CS-DRM get new feathers from the cloud such as elasticity, multitenancy, and distributed storage, which is quite different from common DRM servers. Taking Content Issuer (CI) for example, our content service guarantees the data security, sharing, and isolation among multiple content providers and provides rich as well as powerful services for handling contents, which is not well considered by CI of OMA DRM.

For convenience of explanation, we assume that each application service is provided by a group of servers in the cloud. Therefore, we let certain servers represent corresponding application services. For instance, the content service is provided by a group of servers called content servers, and the license service is provided by those called license servers. In the rest of the paper, we do not differentiate between *application services* and *corresponding servers*.

3.4. Protocols. The CS-DRM scheme has a collection of protocols. Besides several common protocols such as HTTP, RTP [13], RTSP [14], and SSL, we designed two new protocols—License State Word Protocol (LSWP) and License Acquisition Protocol (LAP)—to guarantee the integrity of license files, as well as the safety of the whole scheme. The processes of both LSWP and LAP are illustrated in Figure 2.

3.4.1. The License State Word Protocol. We propose a data structure, License State Word (LSW), stored in the SIM card file system. We apply the data hiding technique proposed in [15] on LSW to make sure that LSW is invisible. The structure of LSW is

$$\text{LSW} = License\_ID \| Rand \| License\_Hash \| Version, \quad (3)$$

where $License\_ID$ is the license index and $Rand$ is a random number in the license and is also a parameter of Algorithms 1 and 2. $License\_Hash$ is the hash value of the license for ensuring the license integrity. $Version$ is an identity for uploading LSW. Only if the LSW version in the license service equals the version of the uploaded LSW, the LSW in the license service will be replaced by the uploaded LSW. When the upload subprotocol of LAP carried out successfully, the version domain of LSW in both the SIM card and license service will be updated.

Table 2: *LSW* Commands.

LSW Command	Function Description
CREATE_LSW	Creating LSW when the DRM agent acquires a license at the first time.
UPDATE_LSW	Updating LSW after the license is consumed.
DELETE_LSW	Deleting the LSW when no license is available.
CHECK_LSW	Checking the integrity of the LSW.
GENERATE_UK	Generating *UK* when the DRM agent wants to render the content.

In order to regulate the operations on LSW between the SIM card and DRM agent, LSWP is designed and adopted in the cloud client. The instructions of the ADPU command for LSWP are *CREATE_LSW*, *UPDATE_LSW*, *DELETE_LSW*, *CHECK_LSW*, and *GENERATE_UK*. The functional descriptions of these instructions are listed in Table 2.

An example of the APDU command *CREATE_LSW* is shown in Table 3. The parameter *CLA* is the class of instruction; “0x88” represents the instruction class for LSWP. *INS* is the instruction code; “0x20” means *CREATE_LSW* command. *Lc* is the number of bytes in the data field of the command; “0x10” means that the *Data* domain is the next 16 bytes behind the *Lc* domain. In the *Data* domain, the first 4 bytes represent *License_ID*, the second 4 bytes represent *Rand*, and the last 8 bytes represent *License_Hash*. *Le* is the maximum number of bytes expected in the data field of the response command; “0x01” means that only 1 byte is expected for the data field of the response.

3.4.2. The License Acquisition Protocol. License Acquisition Protocol (LAP) is a protocol based on XML for license acquisition and is shown in Figure 2. LAP consists of four subprotocols: Trigger subprotocol, Authentication subprotocol, License Transmission subprotocol, and Upload subprotocol. The trigger subprotocol is a 4-pass protocol existing in the rights customization phase, through which users can customize the rights for the related content. The response trigger containing a license server URL and other useful information (e.g., price) will be sent to the DRM agent. After the trigger subprotocol, the license server

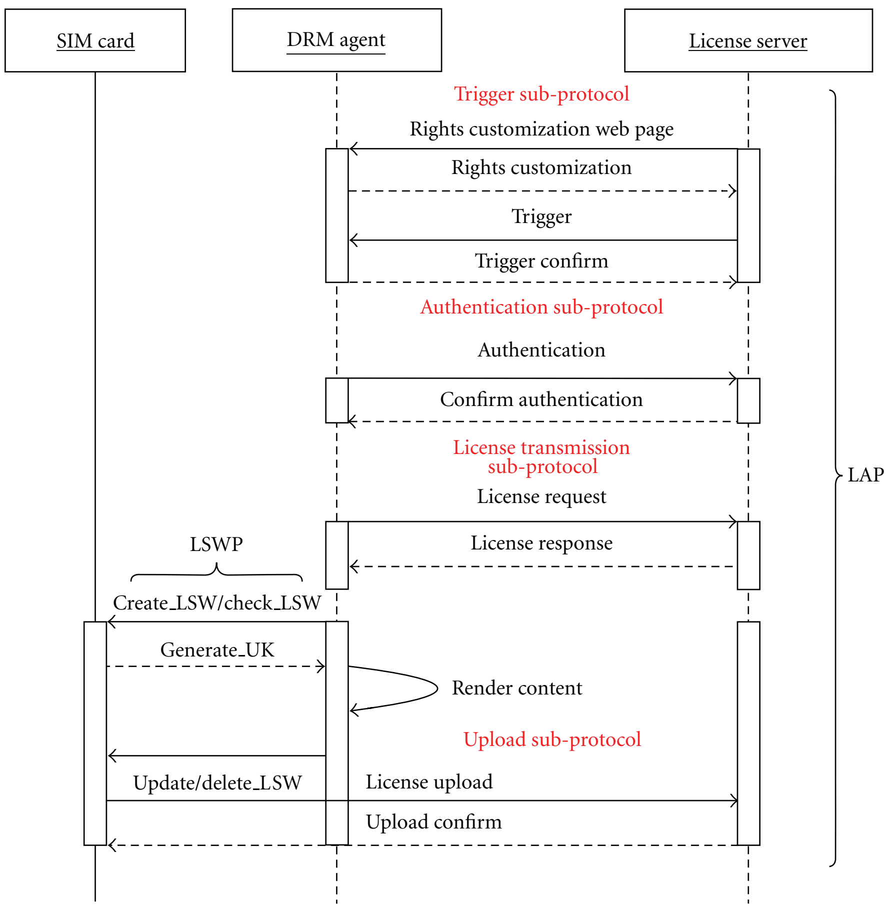


FIGURE 2: The processes of both LSWP and LAP.

TABLE 3: An example of APDU command CREATE_LSW.

CREATE_LSW							
Parameter	CLA	INS	Lc	Data			Le
				License_ID	Rand	License_Hash	
Length	1	1	1	4	4	8	1
Value	0x88	0x20	0x10	0x00000001	0x1d3a9f62	0x1d31e4049a672dc8	0x01

generates the rights for a license. The DRM agent starts a 2-pass authentication subprotocol after confirming the trigger message. The authentication subprotocol is a 2-pass protocol for transmitting and authenticating identity information of the SIM card to the license server as seen from Algorithm 3. The license transmission subprotocol is a 2-pass protocol for transmitting a license to the DRM agent during which the sensitive data have been encrypted by Algorithm 2. The upload subprotocol is a 2-pass protocol for synchronizing the license, and LSW stored in the license server. The upload command is a proactive command [9] initiated by the SIM card.

3.5. Algorithms. There are four main algorithms in the CS-DRM scheme. In this subsection, each algorithm is presented in detail.

```
    Input: $Rand$ : a 4-byte random number from the DRM agent;
           $K_i$ : a secret key stored in both the SIM card and the database of the license server;
    Output: $CK$ : a byte array for SIM-Authentication
(1) $K_c := A8(K_i, Rand)$;
(2) $CK := A5(K_c, Rand)$;
```

ALGORITHM 1: SIM-Authentication $(Rand,\ K_i)$.

3.5.1. SIM Card Authentication Algorithm. The SIM card authentication algorithm, as shown in Algorithm 1, is executed within the SIM card and used to generate the SIM card authentication information CK. After receiving a random

```
Input: $Rand$: a 4 byte random number from the DRM
        Agent;
        $IMSI\_hash$: the index for $K_i$ stored in the
        database of the mobile operator;
        $CK$: the authentication value;
        $Rights\_ID$: the index of rights information stored
        in the database of the license server.
Output: $Message$: the transmission message to the
        DRM Agent
(1) $K_i := GetKI(IMSI\_hash)$;
(2) $K_c := A8(K_i, Rand)$;
(3) $CK^* := A5(K_c, Rand)$;
(4) if $CK^*$ is equal to $CK$ then
(5)     $UK := A_{UK}(K_c, Rand)$;
(6)     $EK := E_{UK}(K_{cek})$;
(7)     $Rights := GetRight(Rights\_ID)$;
(8)     $license := \{License\_ID, Content\_ID, Rights,$
(9)             $EK, Rand, H(HKey \| Rights \| EK \| Rand)\}$;
(10)    $Message := GenerateMessage(license)$;
(11) else
(12)    $error := ErrorInfo(\text{“Unauthentic SIM Card”})$;
(13)    $Message := GenerateMessage(error)$;
```

Algorithm 2: License-Generation (*Rand*, *IMSI_hash*, *CK*, *Rights_ID*).

number *Rand* from the DRM agent, the SIM card generates an intermediate number K_c by K_i, *Rand*, and $A8$. Then, the SIM card generates *CK* and sends it to the license server to distinguish the legal SIM card from impostors.

3.5.2. License Generation Algorithm. The license generation algorithm is executed in a license server to generate licenses. As shown in Algorithm 2, *Rand* is a 4-byte random number from the DRM agent; IMSI_hash is used to get K_i; *CK* is the SIM card authentication information generated by Algorithm 1; *Rights_ID* is the index of usage rights which is generated through the trigger subprotocol. The Hash function needs a special key *HKey* as a parameter. *HKey* is generated by hashing the concatenation of the random number and K_i which can be acquired through IMSI. Since attackers could not get the K_i, they cannot generate *HKey* and tamper the message and hash value without being aware. The license server first verifies *CK* for the SIM card authentication. After that, the license server generates *UK*, and then encrypts K_{cek} by E_{UK}, that is, the symmetric encryption algorithm E_k where the value of k is UK. EK is the encrypted K_{cek}. At the end, the license server generates a license.

A license is an XML file. An example of the license is shown in Figure 3. In general, a license contains two parts. One is the rights related part, which defines the rights information of the digital content, in the *protectLicense* domain, such as the encrypted K_{cek} in the *keyInfo* domain and the constraint of the rights information in the *constraint* domain. The other is the part which describes the user and server information, for example, the *IMSI_hash* domain and the *LicenseServerID* domain. The integrity of a license is ensured by kinds of Hash algorithms.

```
    <LicenseResponse id="0152_0003_BraveHeart"
status="Success">
  <IMSI_hash>
    C6357835D8FA407894653EAC5749CC94
    BC219D155FFB5590C1F46E50E9383196
  </IMSI_hash>
  <LicenseServerID>1</LicenseServerID>
  <nonce>32efd34de39sdwefqwer</nonce>
  <protectedLicense>
    <Payload id="0152" stateful="true" version="1.0">
      <rights id="C.1">
        <context>
          ... ...
          <Uid>0152</Uid>
        </context>
        <agreement>
          <asset>
            <context>
              <Uid>BraveHeart</Uid>
            </context>
            <keyInfo>
              ... ...
              <CipherValue>
                9204587f85f16040a3808ca50b743164
                7aec0fc08012a627a3a9f570404ae4a2
              </CipherValue>
            </keyInfo>
          </asset>
          <permission>
            <play>
              <constraint>
                <count>3</count>
              </constraint>
            </play>
          </permission>
        </agreement>
      </rights>
      <cekKeyInfo>
        <cekKeyRand>76BAC0E6</cekKeyRand>
        <Content_ID>0003_BraveHeart</Content_ID>
        ... ...
      </cekKeyInfo>
      ... ...
    </Payload>
    ... ...
  </protectedLicense>
  <paymentFeedback>6</paymentFeedback>
  <hashValue>
    4CD943D00BA424EF8B1F421066A3AE5F76F31AF8
  </hashValue>
</LicenseResponse>
```

Figure 3: An example of a license.

3.5.3. Key Generation Algorithm. The key generation algorithm is used to generate K_{cek} by decrypting *EK*, which is executed within a SIM card. This algorithm, as shown in Algorithm 3, is the pivotal step of Algorithm 4. The related $A8$ and A_{UK} algorithms are embedded in the hardware of a SIM card. And the output K_{cek} cannot be acquired by attackers because of LSWP.

3.5.4. Content Decryption Algorithm. The content decryption algorithm, as shown in Algorithm 4, is used to decrypt the encrypted content after the DRM agent acquires an available license. The DRM agent first checks whether the rights information is available and then gets the useful information such as *Rand*, *License_ID*, *Rights*, *EK*, and *Hash*. If a license is acquired by LAP for the first time,

```
Input:  $Rand$: a 4 byte random number;
        $EK$: the encrypted $K_{\text{cek}}$ in the license;
Output: $K_{\text{cek}}$: the key for decrypting the encrypted
        content.
(1) $K_c := A8(K_i, Rand)$;
(2) $UK := A_{UK}(K_c, Rand)$;
(3) $K_{\text{cek}} := DE_{UK}(EK)$;
```

Algorithm 3: Key-Generation ($Rand$, EK).

```
Input:  $License$: the license for decrypting the content;
        $En\_Content$: the encrypted content;
Output: $Content$: the decrypted content.
(1) $Flag := Check\_License(License)_{Agent}$;
(2) if $Flag == True$ then
(3)     $License\_ID := GetID(License)_{Agent}$;
(4)     $Rand := GetRand(License)_{Agent}$;
(5)     $Rights := GetRights(License)_{Agent}$;
(6)     $EK := GetEK(License)_{Agent}$;
(7)     $Hash := H(HKey \| License\_ID \| Rights)_{Agent}$;
(8)     if $It\ is\ the\ first\ time$ then
(9)         $CreateLSW(License\_ID,\ Rand,\ Hash)_{\text{SIM}}$;
(10)        $K_{\text{cek}} := Key\text{-}Generation(Rnad, EK)_{\text{SIM}}$;
(11)    else
(12)        if $CheckLSW(License\_ID,\ Rand,\ Hash)_{\text{SIM}}$;
(13)        then
(14)            $K_{\text{cek}} := Key\text{-}Generation(Rnad, EK)_{\text{SIM}}$;
(15)        else
(16)            $error :=$
                $ErrorInfo(\text{“}License\ is\ unavailable''})_{Agent}$;
(17)            $return := GenerateMessage(error)_{Agent}$;
(18)    $Content := DE_{K_{\text{cek}}}(En\_Content)_{Agent}$;
(19)    $UpdateLSW(License\_ID,\ Rand,\ Hash)_{\text{SIM}}$;
(20) else
(21)    $error := ErrorInfo(\text{“}License\ is\ unavailable'')_{Agent}$;
(22)    $return := GenerateMessage(error)_{Agent}$;
```

Algorithm 4: Content-Decryption ($License$, $En\_Content$).

the SIM card creates a corresponding LSW. After that, the SIM card generates K_{cek} by Algorithm 3 and sends it to the DRM agent. The DRM agent decrypts the content by K_{cek} in the secure memory of the device. The subscript of the operation represents the entity where the operation is executed. For example, the subscript “SIM” of the operation $A5(K_c, Rand)_{\text{SIM}}$ means the operation is executed in the SIM card.

4. A Case Study

In this section, we propose a case study for our proposed CS-DRM scheme. This case consists of preparation phase, rights customization phase, license acquisition phase, play phase, and download/upload phase. Figure 4 illustrates the process of the CS-DRM use case, where protocols among $\mathbb{E} \cup \mathbb{S}$ are also shown. The play phase, not appearing in the figure, is executed in the cloud client, and more details are presented in Section 4.4.

4.1. Preparation Phase. The preparation phase does preliminary works for the entire process. The initialization operation of the CS-DRM backend is carried out in this phase.

(a) Content providers upload content to a content server by SSL protocols.

(b) The content server generates the key K_{cek} and encrypts digital content by K_{cek} using symmetric encryption. After that, the content server hosts encrypted content for users to download.

(c) The content server transmits $Content\_ID$, $I_{Content\_ID}$, K_{cek} to the license server in the format of $E_{PKls}(Content\_ID \| K_{\text{cek}} \| I_{Content\_ID})$ where E_{PKls} is an asymmetric encryption operation using the public key PK_{ls} of the license server. $Content\_ID$ is a content identity. In order to make $Content\_ID$ unique, $Content\_ID$ consists of local content identifier and the corresponding content provider identifier. $I_{Content\_ID}$ is the rights description information, such as price and play type.

(d) The license server stores K_{cek}, $Content\_ID$, and $I_{Content\_ID}$, and then generates the rights customization web page which is hosted on the license server for the rights customization phase.

(e) The license server transmits $Content\_ID$ and the URL of corresponding rights customization web page to the content server.

4.2. Rights Customization Phase. The rights customization phase is used to customize the rights of a digital content.

(a) The user browses the content server web site and selects the content (s)he wants.

(b) The DRM agent looks for a license with a $Content\_ID$ suffix in the device. If the DRM agent finds a corresponding license, it will check the license against LSW according to $License\_ID$. The rights customization phase starts if the result is not correct or the rights is unavailable. For example, the rights is available if the rights information allows a user to play the content for 5 times. But the rights is unavailable if the value of play times is 0.

(c) The DRM agent requests the URL of rights customization web page from the content server. As the URL is received, the browser acquires rights customization web page from the license server with this URL and renders it. Then, the user customizes the rights on the web page and confirms the rights information (s)he selects. After that, the trigger message TM, that is, (*Content_ID, License_ID,* rights, *license_url*) is generated and sent by the license server. Here *license_url* is the URL for acquiring the corresponding license.

(d) After the user affirms TM, the license server stores the rights information for generating a license later.

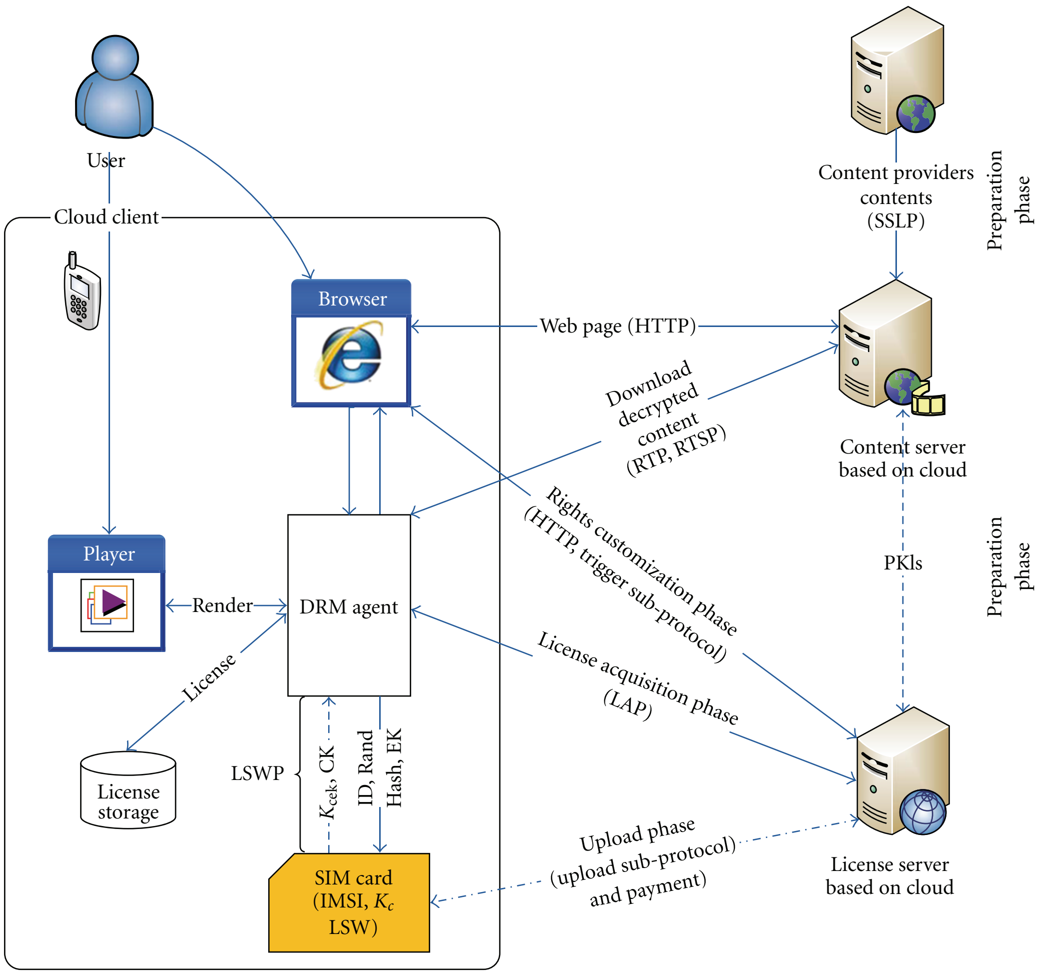

User
Cloud client
Browser
Player
Render
DRM agent
License
License storage
LSWP
K_{cek}, CK
ID, Rand
Hash, EK
SIM card (IMSI, K_c LSW)
Content providers contents (SSLP)
Preparation phase
Web page (HTTP)
Download decrypted content (RTP, RTSP)
Content server based on cloud
PKIs
Preparation phase
Rights customization phase (HTTP, trigger sub-protocol)
License acquisition phase (LAP)
Upload phase (upload sub-protocol) and payment
License server based on cloud

FIGURE 4: The typical process of a use case.

4.3. License Acquisition Phase. The license acquisition phase is used to acquire a license from the license server. The process of both this phase and play phase is shown in Figure 5.

(a) The DRM agent first generates a random number, *Rand*, and passes *Rand* to the SIM card for calculating K_c and *CK* which are used for the SIM card authentication by Algorithm 1. After that, the DRM agent combines *Rand*, *IMSI_hash*, *Rights_ID*, and *CK* as a request and transmits the request to the license server.

(b) The license server checks whether *CK* is equal to the *CK** which is calculated by the same method using *Rand* and *IMSI_hash*. The SIM card is authentic if *CK* and *CK** are identical. The license server creates a license. More details of the license generation are shown in Algorithm 2.

(c) After the DRM agent receives a license for the first time, it creates LSW in the SIM card. The corresponding license and LSW will be updated after the license is consumed. The upload phase is launched in order to synchronize licenses and LSW with the license server when the user starts up the client of CS-DRM.

(d) The payment for the license is accomplished by the SIM card after the user receives the license [16]. Before the purchase, the license server will check the user account first. As long as the SIM card connects to the network of the mobile operator, it will send proactive comments [9] to the system of the mobile operator for the payment.

4.4. Play Phase. The play phase describes steps of rendering a digital content. The process of generating K_{cek} is the key of this phase, as shown in Algorithm 4.

(a) The play phase is initiated when the DRM agent gets a correct license with the *Content_ID* suffix.

(b) The DRM agent extracts *Rand* and *Hash* from the license and checks *Hash* with LSW in the SIM card. If they are equal, the DRM agent sends *Rand*, *EK*, and *License_ID* to the SIM card for generating K_{cek}, as shown in Algorithm 3.

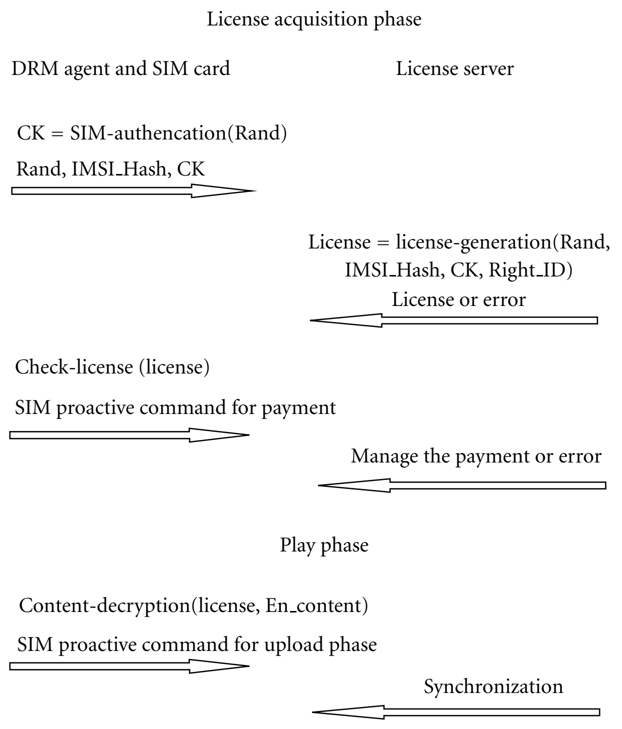

License acquisition phase
DRM agent and SIM card
License server
CK = SIM-authencation(Rand)
Rand, IMSI_Hash, CK
License = license-generation(Rand, IMSI_Hash, CK, Right_ID)
License or error
Check-license (license)
SIM proactive command for payment
Manage the payment or error
Play phase
Content-decryption(license, En_content)
SIM proactive command for upload phase
Synchronization

Figure 5: The process of License Acquisition Phase and Play Phase.

(c) The DRM agent decrypts the encrypted content by K_{cek}. The decrypted content is rendered by the player. The K_{cek} and the decrypted content are stored in the secure memory of mobile device when a user is enjoying the content.

(d) Once the user stops playing, the K_{cek} and the decrypted content will be deleted. At the same time, the DRM agent updates the rights information in the license as well as the LSW in the SIM card.

(e) If the rights information is unavailable, the DRM agent deletes the license and notifies the SIM card to delete LSW.

4.5. Download/Upload Phase. The download/upload phase is for the flexibility and integrity of CS-DRM, where the download phase allows users to change the device and the integrity is guaranteed by the upload phase.

In the download phase, if a user inserts a SIM card into another device which does not have the content and license, the user selects the "Download License" menu item. The DRM agent downloads the license from the license server according to the LSW. The DRM agent checks the new license against the LSW in the SIM card. If the verification fails, the new license will be deleted. Clearly, a user cannot assist another user, say Bob, to "get" the appropriate keys or contents without giving the SIM card to the user Bob.

The upload phase is launched once the user starts up the client of CS-DRM. The SIM card sends the upload proactive command to synchronize the license and LSW stored in the license server when the upload phase starts.

5. Characteristics of CS-DRM

In this section, we discuss characteristics of CS-DRM by analyzing security, privacy issues, the cost of CS-DRM, and how the cloud enhances our DRM scheme. As shown in Table 3, in order to elaborate on these characteristics more clearly, we compare CS-DRM with several typical DRM schemes.

5.1. Security Analysis. The security of our proposed CS-DRM scheme is distinguished by the utilization of the SIM card and LSW. The scheme we proposed is one solution to focus on the encryption of data exchanged between the SIM card and the DRM agent. It is noteworthy that the last releases of the SIM card can support SSL over USB for secure communications between the SIM card and the applications outside the SIM card. When this kind of SIM card is widely used in the mobile industry, it may be the more popular solution for this encryption problem. Although the security of cloud is also an important aspect to CS-DRM, it is not the focus of this paper. There are a lot of research works in this area [18]. Meanwhile, most commercial solutions of cloud, for example, GAE, have their own security principles.

A SIM card is safe enough to store K_i and other important information due to the following points, though it is perhaps running in a hostile environment. Firstly, it is difficult to crack a SIM card thanks to well-developed industrial standards, such as ISO 7816, GSM 11.11, and GSM 11.14. Secondly, algorithms used in a SIM card are kept strictly confidential by the mobile operator. The encryption/decryption schema of CS-DRM can be established on most of the complex and advanced encryption/decryption algorithms, which only need to customize the encryption/decryption algorithms and adjust corresponding encryption/decryption steps in the SIM card and license service. In this paper, for convenience of explanation, the Advanced Encryption Standard (AES) is adopted as the encryption/decryption schema of CS-DRM. Thirdly, data hiding approaches applied in the file system of the SIM card make LSW secure and invisible to crackers. Finally, the interaction between a SIM card and a DRM agent is protected by LSWP.

Another important security issue is how we can make the DRM agent and the player in the mobile device enough trustworthy. CS-DRM solves this issue from two aspects. On one hand, in industry, there are kinds of certification systems based on different mobile operating systems. The mobile device can trust the applications if they pass the verifications of these certification systems. For instance, since we are the Symbian partner in business, we own a special certificate for the electronic signature. The programs of the DRM agent and the player are signed by our special key. The SIM card checks signatures of the DRM agent and the player in order to make sure that they are enough trustworthy. On the other hand, the data stream for the communication between the applications running on the mobile device, such as the DRM agent and

Table 4: A Comparison of Some DRM Schemes.

DRM Scheme	OMA DRM [1]	Conrado et al.'s DRM [4]	TMP-based DRM [17]	CS-DRM
Characteristic	Device based	Smart card based	Trust mobile platform based	SIM card and cloud based
Encrypted Content	Yes	No	Yes	Yes
Privacy Protection	No	Yes	Yes	Yes
PKI used in the license delivery	Yes	No	Yes	No
Information Stored	Public/Private key	PK/SK, SK_p, and SK'_p	Public/Private key and sensitive information for proving the identity	K_i, LSW, and sensitive information for proving the identity

the player, are in the decoded format, which guarantees the security of the communication.

Three types of attacks are listed as follows. We analyze these attacks and present the solutions in CS-DRM.

Attack 1: Tamper the License. CS-DRM guarantees the integrity of a license by the LSW and SIM card. A hash value (Hash) will be calculated according to the license. The DRM agent compares the computed Hash with License_Hash in the LSW by LSWP, before it uses the license. Once the license has been tampered, the computed Hash is different from the License_Hash. The modification can be noticed by the DRM agent, and then a mark is made on the license. The tampered license is not available anymore; therefore, the attacker could not use the tampered license.

Attack 2: Tamper or Detect the LSW. First, the LSW is stored in the SIM card file system which owns complete access control mechanism for the file system. Only authorized program can access and modify the SIM card file system. Because of that, a malicious user could not tamper the SIM card file system. Second, a malicious user can scan the SIM card file system using special tools such as SIMbrush [15]. However, the data hiding approach applied on the LSW makes LSW invisible. Therefore, a malicious user will not detect the LSW information even if (s)he can scan the file system. Then, the attacker could not access the content by tampering or detecting the LSW.

Attack 3: SIM Card Replication Attack. An attacker may replicate a SIM card and attempt to illegally access the content many times. However, the LSW will be modified immediately once the attacker stops playing the content. The information in the license server will be updated when the upload phase starts. In the upload phase, only the LSW with the correct version can be updated in the license server. Otherwise, if another SIM card uploads LSW with the wrong version, the license server will send a command to ask the SIM card to delete LSW. So only one user can access the content by the SIM card at a time.

5.2. Privacy Analysis. CS-DRM has the privacy protection for users. In our scheme, IMSI of the SIM card is the only sensitive information which can be used to know the user identity. However, the hash value of IMSI instead of IMSI itself is used during the process of the license acquisition, so that the license server could not match this hash value with a certain user. Meanwhile, the mobile operator does not divulge user privacy, which makes sure that the license server can only get K_i and is impossible to acquire the user identity information from the mobile operator. Without the permission of the mobile operator, the license server has no rights to check IMSI. Let alone surveying user privacy. Therefore, no one could acquire the privacy information of a user, such as the license list of contents consumed by a user and the user identity.

However, the certain user would be identified by exhaustively searching all possible IMSI instead of reversing the hash value. The number of the legal IMSI is limited. The attack seems to be possible. In order to prevent this kind of exhaustively searching attack, we can apply an encoding/decoding function on the IMSI_hash. Only the SIM card and the mobile operator know how to encode/decode the message. In this way, the IMSI_hash is safe enough for the exhaustively searching. This kind of encoding/decoding function can be flexible. Here we give the mode we used, as shown in Table 5. The IMSI_hash is expressed in hexadecimal. The original IMSI_hash is 20 bytes in the prototype. After the IMSI_hash is encoded in the frontend, the new IMSI_hash is 37 bytes. The first byte denotes the mode of the encoding. If the first bit is 0, the encoding starts from the left-hand side, otherwise from the right-hand side. If there are more 1's than 0's in the rest 7 bits, the distribution of 1's in the next 4 bytes denotes the positions of the original 20 bytes appearing in the new one. The rest positions are padded with random numbers. Otherwise, the distribution of 0's denotes the positions. Only the mobile operator can decode this new IMSI_hash to the original one, and then find the corresponding K_i.

5.3. Cost Analysis. In this subsection, we elaborate on the cost of CS-DRM. Comparing with existing DRM schemes, the cost of CS-DRM is much lower. First, for smart card-based DRM schemes, a SIM card replaces a smart card in CS-DRM, which reduces the cost for issuing a smart card and a smart card reader. Second, the sensitive data, for example, K_{cek} in the license, are encrypted by the

Table 5: The encoding/decoding mode of IMSI_hash.

1 byte		2–5 bytes	6–37 bytes expressed in hexadecimal
1 bit	2–8 bits	32 bits	The original IMSI_hash: C635D8403E49CC949D155FFB90C16E50E9383196
0	1001110	10011010001001110011110110111111	The new IMSI_hash in the encoded format: C6*357*835D8*FA*407*89465*3E*AC57*49CC94B*C219*D155FFB*5*590C1*F46*E50E9383196
Left	The distribution of 1's denotes the positions		

encryption/decryption and authentication mechanisms designed in CS-DRM. Comparing with some DRM schemes (such as OMA DRM) which protect the sensitive data depending on the mechanisms of Certificate Authority (CA) and PKI, CS-DRM removes CA from the scheme and then reduces the cost of purchasing certifications for each cloud client. Third, as we mentioned before in Section 2.2.3, CS-DRM is a cloud-based DRM scheme whose most important characteristic is its "pay-as-you-go" manner. The owner of CS-DRM does not need to purchase the infrastructure which may be too expensive to be afforded, such as software and physical servers. CS-DRM only demands to rent services provided by the cloud computing. The cost of renting services is much lower than that of buying software and hardware. Meanwhile, the high elasticity of the cloud brings capabilities of matching resources to workload much more closely by adding or removing resources at an acceptable time of minutes rather than weeks, which makes CS-DRM satisfy the requirements of cloud clients automatically according to the current demands with a low cost. Also, the disaster recovery and maintenance cost of the entire system is reduced by the cloud.

5.4. Cloud DRM. CS-DRM is a cloud-based DRM scheme. The cloud enhances our DRM scheme at three following aspects. First, the virtualization technology used above the infrastructure of the cloud guarantees the data security, sharing, and isolation among tenants of the content server. Second, because of the cloud, CS-DRM has high elasticity as well as the "pay-as-you-go" manner of the cloud. The cost of our DRM scheme is significantly lower than that of others, especially when the number of active users scales up. Third, cloud computing is a large service platform. We can integrate the most popular services nowadays to our DRM scheme. Based on the cloud, CS-DRM can own powerful service support, which makes CS-DRM a flexible and humanistic system with wonderful user experience. For instance, the content server can provide kinds of content editing and format conversion services for content providers.

6. Implementation

We have implemented a prototype, called "Phosphor", of our proposed CS-DRM scheme. Phosphor contains both the frontend and the backend. Specifically, Phosphor is designed for protecting mobile streaming multimedia. Obviously, we can easily extend Phosphor to protect other kinds of media. A preliminary description on Phosphor was presented in [19]. In this section, we discuss the implementation details of Phosphor. Firstly, we present the frontend and backend of Phosphor in Sections 6.1 and 6.2, respectively. Secondly, video encoding/decoding methods used in Phosphor are brought forth. Finally, we concern the user experience of the phosphor client.

6.1. Frontend. To verify our DRM scheme, we developed a DRM video client, which is allowed to use the standard video on demand services via RTSP protocol [14], on a mobile device. It is a Symbian C++ application using the Nokia S60 3rd Edition Feature Pack 2 Software Development Kit [20]. We implemented a browser and a player besides the DRM agent and SIM card. A user can browse the portal web site hosted on the content server and the rights customization web site on the license server by the browser. Meanwhile, the player is for rendering video stream. A client receives the encrypted video data from the content server via RTP protocol [13] and applies for the decryption key from the SIM card via APDU commands. If the acquisition of the key is successful, the client decrypts the video data, decodes the decrypted data to video frames, and renders the video frames on the screen of the mobile device periodically.

6.2. Backend. We developed and deployed application services of the CS-DRM scheme in both a private cloud and a public cloud. The private cloud is a cluster of local machines with abundant resources such as large storage and efficient computation. Meanwhile, the public cloud is based on the GAE under the J2EE framework. GAE [21] has several features such as dynamic web serving, automatic scaling, and load balancing. In the public cloud, the data of CS-DRM is stored in the file system of Google through JDO. In order to test and update the prototype easily and conveniently, Phosphor mainly runs in the private cloud. The public cloud is used to learn the characteristics of public cloud platform and do comparative experiments with the private cloud. In the content service, the content portal web site is hosted and media are transmitted to cloud clients by Darwin Streaming Server (DSS) [22]. However, DSS based on C programming language could not be hosted on GAE which only supports Java and Python. Therefore, DSS is only deployed locally. We set up the license customization web site for license customization phase on the license server. The communication for license acquisition between cloud clients and the license server is implemented based on HTTP, Web services, and Java servlet technologies.

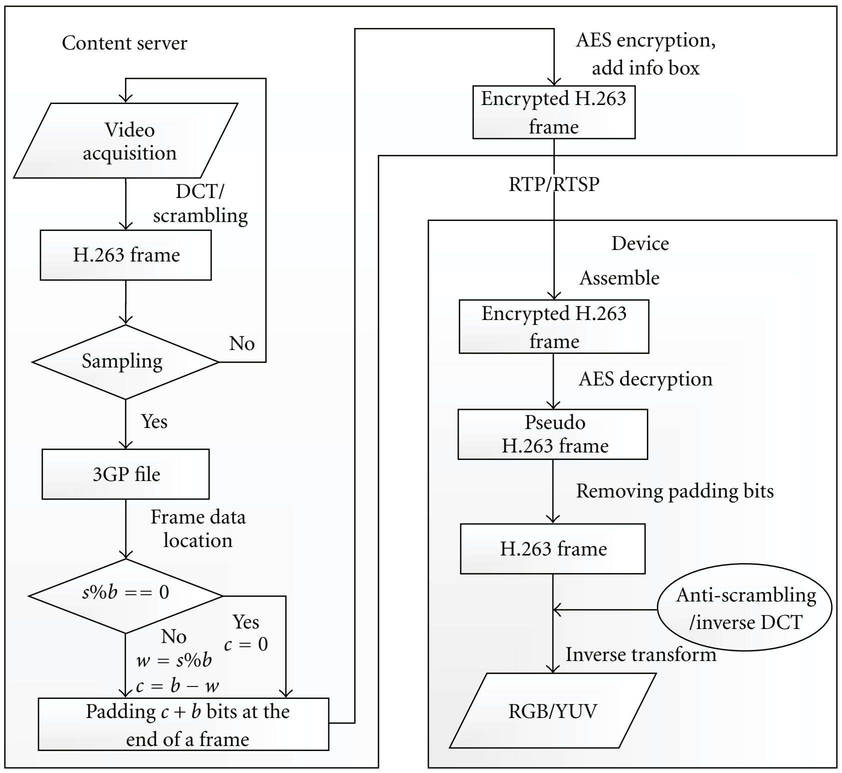


FIGURE 6: The implementation process of the improved H.263 video encoding/decoding method in Phosphor.

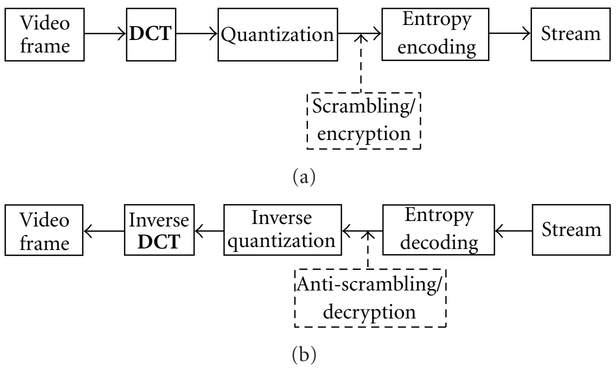


FIGURE 7: The process of the improved H.263 video encoding/decoding: (a) is the process of the improved H.263 video encoding; (b) is the process of the improved H.263 video decoding.

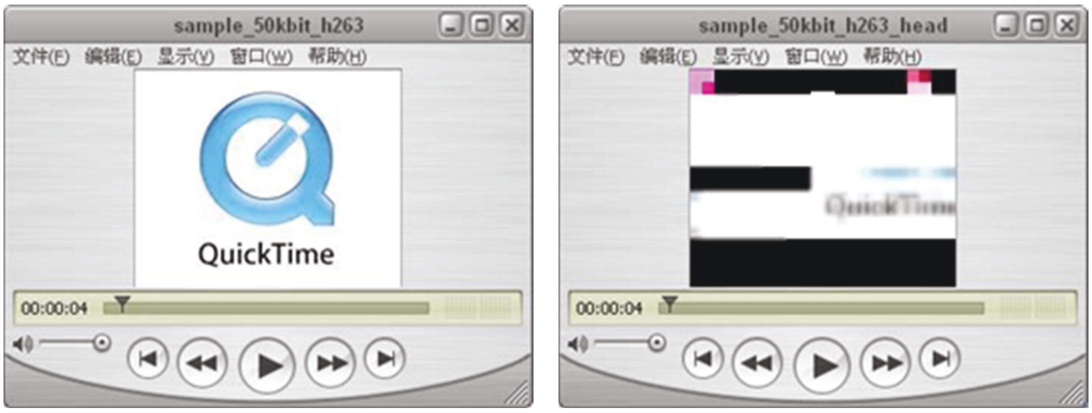
6.3. Encoding/Decoding. We adopt an improved H.263 video encoding/decoding method in Phosphor. The implementation process is shown in Figure 6. The encoding process is carried out in the content server. First, we acquire the original frames from the video. Through the DCT transform and scrambling operations, the original video frame becomes a H.263 frame. Then, the frame is encrypted by AES with counter (CTR) mode [23] using K_{cek} and encoded by the entropy encoding operation. After that, these frames constitute a video streaming which is transmitted to the device through RTP and RTSP. More details can be found in Figure 7(a). The decoding process in the device, as shown in Figure 7(b), is the inverse process of encoding.

Different from traditional encoding/decoding methods, there are some improved ones, such as [24], in which the scrambling operation is added into the encoding/decoding process. In Phosphor, similarly, the scrambling operation is after the quantization and before the entropy encoding in the content service. Meanwhile, the inverse scrambling operation is between the inverse quantization and the entropy decoding, after the streaming is received by the device. More details are shown in Figure 7.

We compare the original video data with the encoded video data. The result of the comparison, shown in Figure 8, demonstrates that the video is well protected in Phosphor. Furthermore, the improved H.263 video encryption/decryption method provides higher security and reduces the computational cost of encryption/decryption operations.

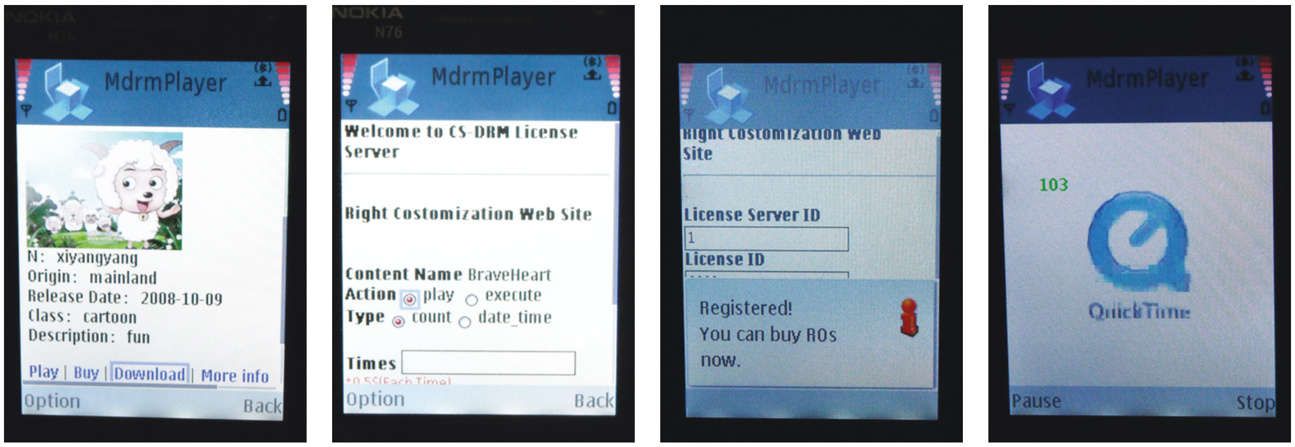
6.4. User Experience. We implemented and deployed the client of Phosphor on a real device, Nokia N76. In this subsection, a perceptual understanding of Phosphor through user operations is given. The following pictures shown in Figure 9 are taken by a camera to record some typical scenarios of Phosphor on the real device.

As shown in Figure 9(a), a user is browsing the content server web site and selecting her/his favorite. The user customizes the rights of the content in Figure 9(b) if there is


sample_50kbit_h263
文件(F) 编辑(E) 显示(V) 窗口(W) 帮助(H)
QuickTime
00:00:04
sample_50kbit_h263_head
文件(F) 编辑(E) 显示(V) 窗口(W) 帮助(H)
00:00:04

(a) (b)

Figure 8: The comparison between the original video data and the encoded video data. (a) is the original video data; (b) is the video data after the improved H.263 encoding method.



N: xiyangyang
Origin: mainland
Release Date: 2008-10-09
Class: cartoon
Description: fun
Play | Buy | Download | More info
Option Back

Welcome to CS-DRM License Server
Right Costomization Web Site
Content Name BraveHeart
Action [✓ play] execute
Type [✓ count] date_time
Times
Option Back

Right Costomization Web Site
License Server ID
1
License ID
Registered!
You can buy ROs now.

103
QuickTime
Pause Stop

(a) (b) (c) (d)

Figure 9: The implementation diagram of Phosphor: (a) describes the scene searching content on the content server web site; (b) describes the scene customizing the rights on the license server; (c) describes the scene where the DRM agent is acquiring a license after successful registration; (d) describes the process of rendering the media in the compliant player.

no license in the device. In detail, the user can customize the rights in actions or types. For example, (s)he can customize rights to play a video segment 3 times or in a time period. After the user finishes customizing the rights, the trigger message is sent to her/him. At present, the users can acquire the license (see Figure 9(c)). Finally, as shown in Figure 9(d), the user can watch the streaming media content, after (s)he acquires the license and pays for it.

7. Experiments

We conduct extensive experiments on the performance of our proposed CS-DRM scheme and Phosphor. In this section, the experimental setting is described at first. Then, extensive experimental results are reported to show the effectiveness and the efficiency of the CS-DRM scheme.

For the aspect of the efficiency, we only focus on the license service, because the results of the experiments on the license service are representative and the similar conclusion can also be reached to the content service.

It is worth noting that DRM is a special area for security. We do not use any benchmark in our experiments. A DRM system involves too many domains, for example, encryption/decryption, network transmission, the implementation technology of the compliant client and servers; therefore, there is no public benchmark so far.

7.1. Experimental Setting. Experiments run on both a cluster of machines locally as a private cloud and the GAE platform as a public cloud. In the private cloud, each local machine has 4 single-core processors (2.1 GHz), 4 GB main memory, and 1.5 TB hard disk. Each physical machine runs Ubuntu 9.04 with Java SE 6.0. In the experimental environment, Hadoop 0.19.2 and HBase 0.19.3 are used as the datastore, Apache Tomcat 6 as the http server, Axis2 as the web service tool. In order to compare with private cloud environment,

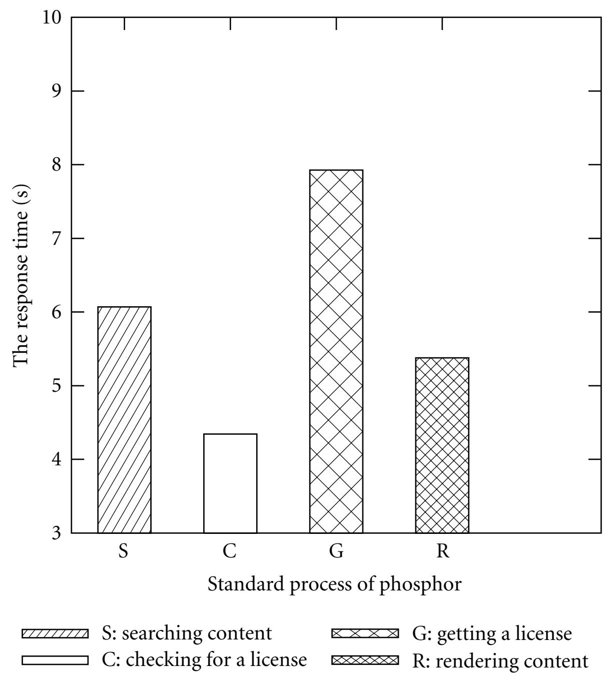


Figure 10: Running time.

the license service is also deployed on the GAE. However, since we only have a free account of GAE, there are a lot of limitations. For example, the maximum number of simultaneous dynamic requests (users) is 30. It means that we cannot simulate hundreds of users to connect the license server concurrently. But it is enough for us to study the performance of application services in both the private and public clouds.

7.2. Effectiveness. We run our program in the N76 to illustrate the effectiveness of CS-DRM according to a general process. We divided the general process into four main steps under different scenarios. The first scenario starts when a user browses the content portal web site and ends when (s)he selects her/his favorite. The second scenario starts when the user clicks the "play" button to check for a license for watching the content and ends when the rights customization web page appears in the browser. The third scenario starts when the user customizes the rights and ends when (s)he acquires the license. The fourth scenario starts when the user gets the license and ends when the user can watch the media. As shown in Figure 10, we measure the approximate running time of the whole general process, which includes the time of artificial operations, such as filling out forms.

Analyzing the running time of each scenario of CS-DRM, we conclude that the third scenario takes more time than other scenarios. It takes about 8 seconds in this scenario to customize the rights, authenticate, and transmit the license. Users need to do some artificial operations during the entire process. Just because of artificial operations, users would not feel intolerable or boring. Removing the time cost of these artificial operations, our scheme has excellent effectiveness.

7.3. Efficiency. As mentioned in Section 5.3, the low access speed and long latency are two major factors limiting the development of the mobile Internet. In the CS-DRM scheme, the response time of the communication between cloud clients and cloud services becomes the most important indicator of the performance. Both the license service and content service are deployed in the cloud. We consider that the performance of these two services is the key for CS-DRM to provide a better user experience. Naturally, the response time for user requests is an appropriate indicator for the performance. Here we focus on the life cycle of a license, which is involved in the LAP.

In this experiment, we simulate simultaneous users sending requests to license service for acquiring licenses and calculate the response time—$Total$ and $Average$. $Total$ is the running time from the first user sending a request to the last user receiving a response and represents the longest delay the user may incur. $Average$ is the average response time of each user and is calculated as

$$Average = \frac{\sum_{k=1}^{n} Time_k}{n}, \tag{4}$$

where n is the number of simultaneous users and $Time_k$ is the response time of the kth user. In this experiment, lots of mobile phone simulators are utilized to run on computers, because there are not enough physical mobile devices to simulate hundreds of simultaneous users. The number of simultaneous users for the private cloud increases from 50 to 500 with a step 50. Meanwhile, the number for GAE increases from 5 to 30 with a step 5. If the number of simultaneous users in GAE is bigger than 30, the error rate of requests for GAE will raise extremely fast due to the limitation on the maximum number of simultaneous requests for free. For example, when the number of simultaneous users is 100, the error rate is 78.4%.

As shown in Figures 11 and 12, both $Total$ and $Average$ increase when the number of simultaneous users scales up. Meanwhile, the total running time is a little longer than the response time of the individual user. The results on both GAE and the private cloud are reasonable. As shown in Figure 11, for GAE, when the number of simultaneous users is lower than the quota, the license service can get enough resources to deal with requests without much latency. Also, there are several interesting features. Firstly, $Average$ of GAE increases abruptly when the number of simultaneous users increases from 5 to 10. But $Average$ remains with very little increase when the number increases from 15 to 30. For the free account of GAE, we guess a watershed of the number of simultaneous users is a number between 5 and 10. When the number of simultaneous users is less than the watershed, one cluster of machines can deal with the requests. If the number of simultaneous users is larger than the watershed and lower than the quota, another cluster of machines joins. The result is that the simultaneous requests are dealt with without much latency. Secondly, when the number of simultaneous users reaches 30, less than 5% requests receive error responses,

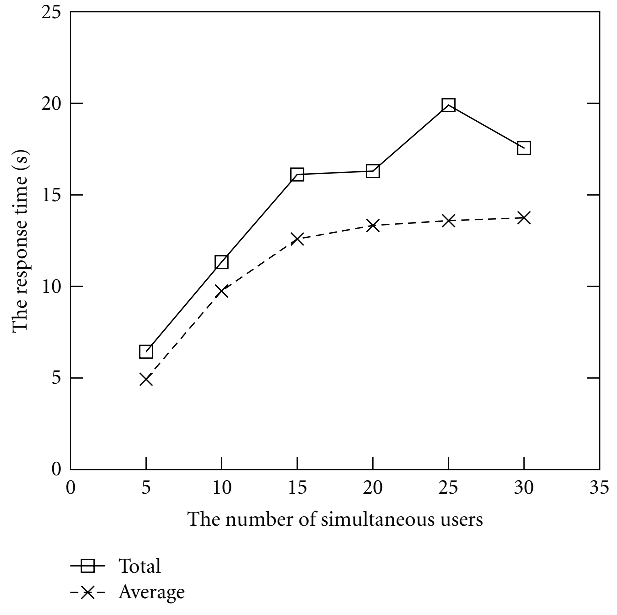


Figure 11: Total response time in the public cloud.

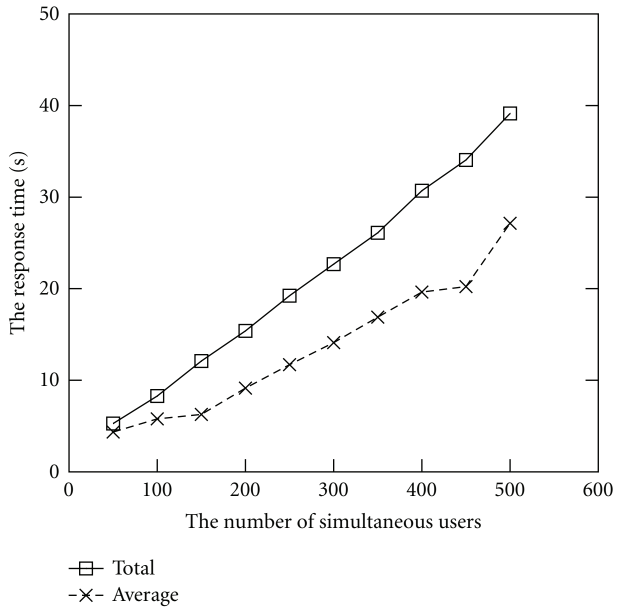


Figure 12: Total response time in the private cloud.

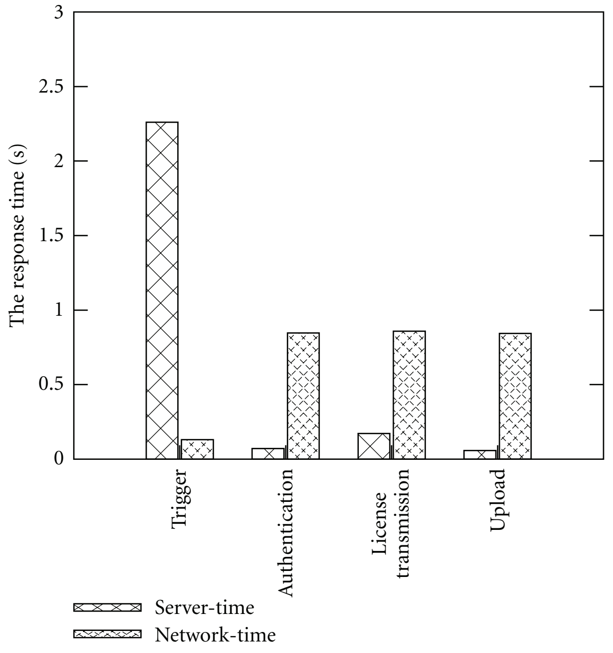


Figure 13: Response time of LAP in the private cloud with 100 simultaneous users.

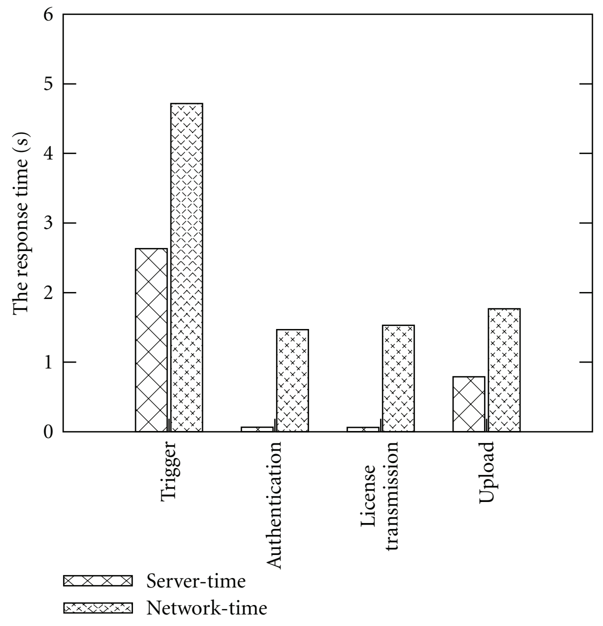


Figure 14: Response time of LAP in the public cloud with 20 simultaneous users.

which reduces the *Total*. The last point of *Total* in Figure 11 fits well with the prediction.

In the private cloud, as shown in Figure 12, both *Total* and *Average* increase proportionally to the number of simultaneous users. It means that our private cloud does not do well in the distributed computing, which can be improved in the future. When there are 500 simultaneous requests, the license server reaches a limitation of the resources of the private cloud. It has to slow down and wait for new resources to deal with rest requests. However, the total running time of license service on GAE is much longer than that on the private cloud when the number of simultaneous users is the same. The main reason for this situation is the network latency of GAE which is the main factor influencing the performance of license service in the public cloud. The results of the next experiment just explain it well.

In order to investigate factors influencing the performance, we analyze the response time of subprotocols of LAP according to *Server-Time* and *Network-Time*, where *Server-Time* is the response time for dealing with user

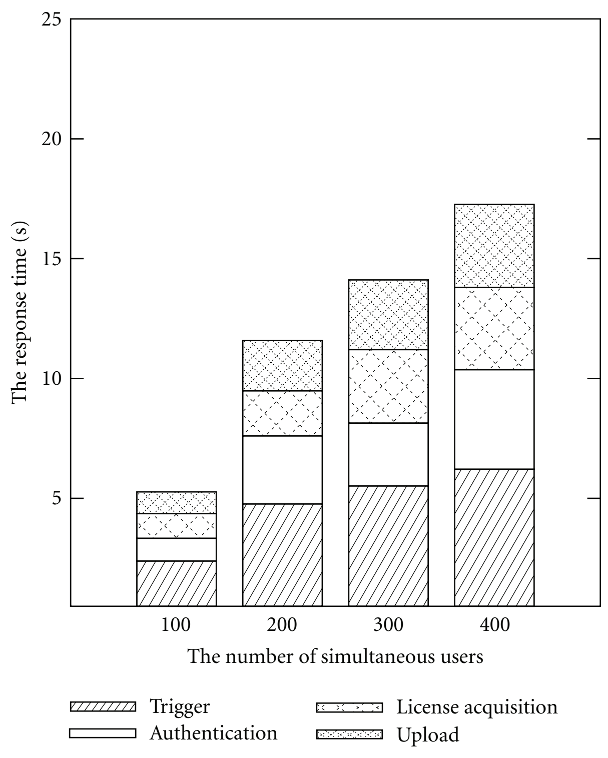


FIGURE 15: Response time of LAP in the private cloud with increasing simultaneous users.

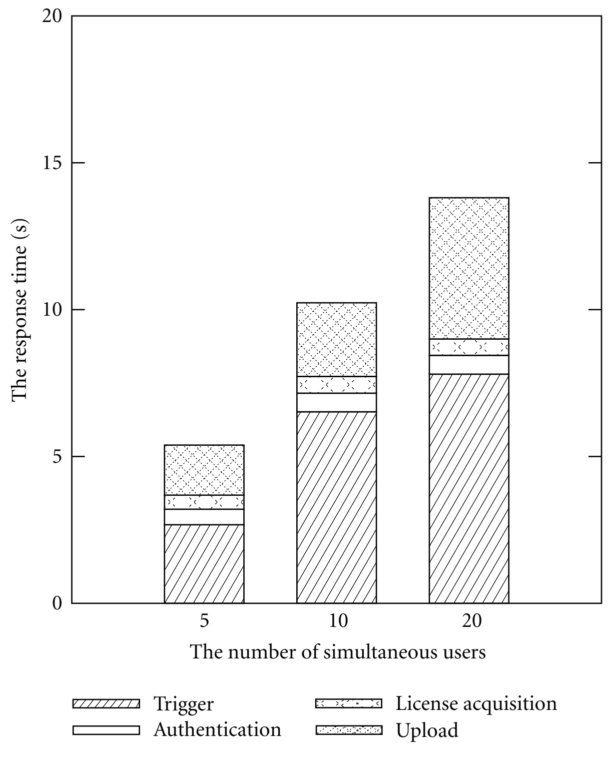


FIGURE 16: Response time of LAP in the public cloud with increasing simultaneous users.

requests and generating responses in the license service, and *Network-Time* is the network latency of the message transmission. Here we fix the number of simultaneous users. The simultaneous number of GAE is 20, and that of the private cloud is 100. The results of two experiments are shown in Figures 13 and 14. Obviously, the network latency is the main factor influencing the performance of cloud. The differences on *Server-Time* between GAE and the private cloud are not too much, which means that the efficiency can be satisfied by both the private and public clouds. Comparing with each subprotocol, the trigger subprotocol needs the longest *Server-Time*. In this subprotocol, the cloud analyzes the requests, generates, and stores usage rights with a large number of data operations for the corresponding licenses. The upload subprotocol takes more time on network due to uploading licenses and LSW. The other subprotocols do not involve so much data operations; therefore, they take less *Server-Time*.

From the analysis of the experiments on subprotocols of LAP both in the private and the public clouds (see Figures 13 and 14), we conclude that, for the public cloud, *Server-Time* changes a little, while *Network-Time* becomes the main factor for GAE to provide high quality services. Since that, we argue that there are some optimization opportunities for reducing *Network-Time* in the public cloud, such as XML compression. We can encode the license and transfer the messages, which are both in the XML format, into binary code. These binary encoded streams are used for delivery between a client and a backend. However, the encoding and decoding process may take more computational cost.

We show the relationship between the response time of LAP including its subprotocols and the number of simultaneous users. The results are shown in Figures 15 and 16. The response time of each LAP subprotocol increases with the increasing of the number of simultaneous users in the private cloud. Moreover, the response time of the trigger subprotocol and that of the upload subprotocol increase faster than those of the other two subprotocols in public cloud.

8. Related Work

Digital Rights Management is an active area of research with a lot of prior works. Device-based DRM schemes and smart card based DRM schemes are two main categories for the existing DRM schemes. OMA DRM [1] and Microsoft DRM [2] are device-based DRM schemes which have a common problem—inflexibility. In order to solve this problem, smart card-based DRM schemes are proposed [4–6].

A significant portion of research on DRM systems is the mobile DRM which has some special features such as the wireless environment and the constraints on computation as well as storage. The OMA completes an open standard for mobile DRM. But it is still not detailed enough. Comparing with the present situation of the usage of SIM card in the DRM scheme, we design the whole scheme based on a SIM card, including a set of algorithms and protocols. However, OMA has standardized the usage of a SIM card in the DRM system in its last document of DRM specification. In the specification, a SIM card is only used to bind Rights Objects (RO) with user identity. Considering the authority of OMA, both the scientific research and engineering practice on the SIM card-based DRM should pay close attention to the OMA DRM specification in the future. Smart card-based DRM systems are more suitable for the mobile environment. However, there are security issues, such as imposter attacks [4, 5], in some existing smart card-based DRM schemes. Although some solutions (e.g., [6]) are proposed to solve these security issues, the complexity and cost of the system dramatically increase.

Reference [17] is a relevant prior work about the SIM card. It proposes a TMP-based DRM system which combines the trust mobile platform (TMP) and OMA DRM. The TMP-based DRM introduces SIM card to the DRM scheme, but the usage of SIM card is very limited. The SIM card is just for the authentication in [17], such as the confirmation of the user identity for TMP, network as well as Rights Issuer (RI). Moreover, the concrete authentication steps and authentication algorithms are not mentioned anymore. The security issue about the communication between TMP and SIM card is also ignored. Reference [25] is another relevant work on the multimedia content distribution on the mobile phone. It introduces smart card in the DRM scheme. Meanwhile, watermarking is applied to detect piracy in the network. Reference [26] applies a smart card to the condition access system and designs a communication method and the interface between a set-top box and a smart card. Reference [27] designs a fair exchange protocol for trading electronic rights to claim goods and services. Reference [28] proposes a smart card-based digital content protection scheme for the workflow and implements the system.

There are also some related works on the performance issue of mobile DRM schemes. References [29, 30] are two schemes on how to improve the performance of OMA DRM scheme. The device identifier is introduced to the OMA DRM in [29]. The RI server encrypts RO by using the symmetric encryption instead of asymmetric encryption. The device identifier is the symmetric key for the symmetric encryption. However, the device identifier has to be transmitted by asymmetric encryption using public key of RI. Since that, this scheme not only lacks flexibility, like device-based DRM system, but also increases the cost and complexity by transmitting the device identifier to RI. Reference [30] improves the performance of the RO generation algorithm by reducing generation and usage times of the random number. However, the performance improvement is very limited.

9. Conclusions

This paper presented a new DRM scheme—CS-DRM which is a cloud-based SIM DRM scheme for the mobile Internet. CS-DRM introduces a SIM card and cloud computing to save the cost and provide higher security. Meanwhile, a prototype of CS-DRM called Phosphor has been implemented. Furthermore, our experimental results on the performance of CS-DRM show that CS-DRM can satisfy performance requirements of users well. All of these demonstrate that CS-DRM is low cost, high efficient, secure, and practicable. However, there are still some optimization opportunities for CS-DRM, such as XML compression, which are our future works.

Acknowledgments

The authors would like to thank the editor and the two anonymous reviewers, who have made very valuable and constructive comments that resulted in an improved paper. Their thanks go to Wanpeng Tian and Dalei Bao for their help in the deployment of Phosphor. Their thanks also go to Shen He and Xianbo He for their hard work in authors' mobile DRM group. This work was partially supported by the National Natural Science Foundation of China (no. 60803016 and no. 90718010), the National Basic Research Program of China (no. 2007CB310802), the National HeGaoJi Key Project (no. 2010ZX01042-002-002-01), and Tsinghua National Laboratory for Information Science and Technology (TNLIST) Cross-discipline Foundation.

References

[1] "OMA DRM Specification V 2.1," 2009, http://www.openmobilealliance.org/.

[2] "Microsoft DRM," 2009, http://www.microsoft.com/windows/windowsmedia/forpros/drm/default.mspx.

[3] "ISO 7816 Smart Card Standard," http://www.iso.org/iso/search.htm?qt=7816&sort=rel&type=simple&published=true.

[4] C. Conrado, F. Kamperman, G. J. Schrijen, and W. Jonker, "Privacy in an identity-based DRM system," in *Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, pp. 389–395, 2003.

[5] H. M. Sun, C. F. Hung, and C. M. Chen, "An improved digital rights management system based on smart cards," in *Proceedings of the Inaugural IEEE-IES Digital EcoSystems and Technologies Conference (DEST '07)*, pp. 308–313, February 2007.

[6] S. Palavalli, U. S. Srinivas, and A. R. Pais, "Identity based DRM system with total anonymity and device flexibility using IBES," in *Proceedings of the 22nd EUROPEAN Conference on Modelling and Simulation*, June 2008, http://www.scs-europe.net/conf/ecms2008/ecms2008%20CD/hpcs2008%20pdf/hpcs08w1-1.pdf.

[7] "Statistical Report on Internet Development in China," July 2009, http://www.cnnic.cn/uploadfiles/pdf/2010/8/24/93145.pdf.

[8] "GSM 11.11," http://www.ttfn.net/techno/smartcards/gsm11-11.pdf.

[9] "GSM 11.14," http://www.ttfn.net/techno/smartcards/GSM11-14V5-2-0.pdf.

[10] M. Armbrust, A. Fox, R. Griffith et al., "Above the clouds: a berkeley view of cloud computing," Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley, February 2009.

[11] M. Creeger, "Cloud computing: an overview," *Queue*, vol. 7, no. 5, pp. 3–4, 2009.

[12] P. Zou, C. Wang, Z. Liu, J. Wang, and J. Sun, "A cloud based SIM DRM scheme for the mobile internet," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, pp. 759–761, October 2010.

[13] "RTP: A Transport Protocol for Real-Time Applications. RFC 3550," http://www.ietf.org/rfc/rfc3550.

[14] "Real Time Streaming Protocol (RTSP). RFC 2326," 1998, http://tools.ietf.org/html/rfc2326.

[15] A. Savoldi and P. Gubian, "Data hiding in SIM/USIM cards: a steganographic approach," in *Proceedings of the 2nd International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE '07)*, pp. 86–97, Bell Harbor, Wash, USA, April 2007.

[16] K. Wrona, M. Schuba, and G. Zavagli, "Mobile payments—state of the art and open problems," in *Proceedings of the Second International Workshop on Electronic Commerce*, pp. 88–100, November 2001.

[17] Y. Zheng, D. He, H. Wang, X. Tang, and L. Fiege, "Secure DRM scheme for future mobile networks based on trusted mobile platform," in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing*, vol. 2, pp. 1164–1167, 2005.

[18] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*, pp. 199–212, November 2009.

[19] P. Zou, C. Wang, Z. Liu, and D. Bao, "Phosphor: a cloud based DRM scheme with sim card," in *Proceedings of the 12th Asia-Pacific Web Conference on Advances in Web Technologies and Applications (APWeb '10)*, pp. 459–463, April 2010.

[20] "Symbian Developer," http://developer.symbian.org/.

[21] "Google App Engine," http://code.google.com/appengine/.

[22] "Darwin Streaming Server," http://dss.macosforge.org/.

[23] "Using Padding in Encryption," http://www.di-mgt.com.au/cryptopad.html.

[24] F. Dufaux and T. Ebrahimi, "Scrambling for video surveillance with privacy," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2006, June 2006.

[25] C. Fontaine, C. Delpha, P. Duhamel et al., "An end-to-end security architecture for multimedia content distribution on mobile phones," *Journal of ISAST Transactions on Communications and Networking*, vol. 2, no. 1, pp. 81–91, 2008.

[26] K. Gi-seop, "Communication method between set-top box and smart card and interface module used for the same," US Patent 2006/0 174 259 A1, August 2006.

[27] M. Terada, M. Iguchi, M. Hanadate, and K. Fujimura, "An optimistic fair exchange protocol for trading electronic rights," in *Proceedings of the 6th Smart Card Research and Advanced Application IFIP Conference (CARDIS '04)*, pp. 255–270, 2004.

[28] A. Durand, M. Éluard, S. Lelievre, and C. Vincent, "SmartPro: a smart card based digital content protection for professional workflow," in *Procedings of the 8th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Applications (CARDIS '08)*, vol. 5189 of *Lecture Notes in Computer Science*, pp. 255–266, September 2008.

[29] Z. Liu, G. Liu, and B. Lee, "An efficient key distribution method applying to OMA DRM 2.0 with device identifier," in *Proceedings of the 9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 3–7, 2008.

[30] J. Han, H. -Y. Chang, S. Cho, and M. Park, "EMCEM: an efficient multimedia content encryption scheme for mobile handheld devices," in *Proceedings of the International Conference on Information Science and Security (ICISS '08)*, pp. 108–114, January 2007.