**Noname manuscript No.**
(will be inserted by the editor)

# Redistributing time-based rights between consumer devices for content sharing in DRM system

**Sangho Lee · Jong Kim · Sung Je Hong**

the date of receipt and acceptance should be inserted later

**Abstract** Device-based DRM systems tightly bind rights for content to a device. However, it can decrease the consumers' convenience because it disturbs consumers who want to use the already purchased content with their other devices freely. Previous research into solving this problem still have burdens such as restricting the number of devices that a consumer can use and requiring a special device that manages content sharing. In this paper, we propose a new rights sharing scheme which does not restrict the number of devices that a consumer can use and does not require a specialized device. In our scheme, the right to use content is represented as the right to use the content for a certain amount of time. Consumers can use the content with any of their devices by redistributing the usage amount of time between devices. The redistribution process only requires local synchronization among participating devices. To prevent illegal content sharing and to detect illegally increased content usage time, the amount of time that a consumer can have is limited and the rights for each unit of time has a unique number to prevent illegal duplications. We present data structures and protocols, analyze security properties of our scheme, compare our scheme with related work, and evaluate our scheme through implementation.

**Keywords** Digital rights management · Rights sharing · Content sharing · Domain model

## 1 Introduction

Digital rights management (DRM) technology was introduced to protect digital content from illegal consumers in digital environments. Nowadays, the importance of DRM is increasing due to the growth of the digital content market. However, consumers dislike current DRM systems because they do not consider consumers' convenience [11]. One of the significant problems that bring consumers' inconvenience is the rights sharing problem which is the focus of this paper.

The rights sharing problem revolves around how to share the consumers' purchased content among their devices for their convenience. Consumers may have many devices and they want to use purchased content with any of their device freely. But previous device-based DRM systems cannot support this requirement because they give the rights to a device and permit only that device to use the content. Hence, consumers need to buy the same content several times if they want to enjoy the same content on their other devices. Consequently, the device-based DRM system is not suitable in the situation where consumers have several devices; therefore, we need another method [4,11].

The rights sharing problem has conflicting requirements from consumers and service providers. Consumers require loosely restricted sharing because they want convenience. However, service providers require tightly restricted sharing because they want to prevent illegal

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute for Information Technology Advancement) (IITA-2009-C1090-0901-0045).

Sangho Lee · Jong Kim · Sung Je Hong
Department of Computer Science and Engineering,
Pohang University of Science and Technology (POSTECH),
Hyoja-dong, Nam-gu, Pohang, Korea
E-mail: sangho2@postech.ac.kr

Jong Kim
E-mail: jkim@postesch.ac.kr

Sung Je Hong
E-mail: sjhong@postech.ac.kr

sharing. Therefore we should find a tradeoff between loosely restricted and tightly restricted sharing to increase the satisfaction from both consumers and service providers.

Many researchers and industry participants have suggested various schemes for the rights sharing problem [7, 9,10,13,15–17]. One of them is the authorized domain model that permits a consumer to create a content sharing group, which is composed of a limited number of devices, named as a domain [7,10,13,15,16]. All devices belonging to the same domain can share rights objects in that domain. However, the restriction on the number of devices is one of the problems of the domain model because it is difficult to find the optimal number of devices that satisfies both consumers and service providers. Also, the domain model requires globally synchronized information such as the number of devices that belong to the domain and the information about each device such as a device ID.

Other schemes based on preventing a simultaneous usage of content are also introduced [9,17]. One of them is a smart card-based scheme [9]. This scheme can support an unlimited number of devices by transferring rights objects that are stored in the smart card between devices. However, this scheme requires every device to support the smart card. Moreover, consumers must always bring their smart cards with their devices. Another scheme is a log-based simultaneous usage detection scheme [17]. In this scheme, every device logs its content usage time and if devices are connected with each other then they exchange their logging data to detect time overlap. This scheme assumes that a secure and globally synchronized clock is installed in every device, but it is difficult to realize. In addition, sometimes a consumer may want to use content with his/her device at the same time or a consumer may accidentally start devices by mistake. The log-based scheme can decide those cases as illegal activities but it is a doubtful decision.

### 1.1 Research goals

In this paper, we propose a novel rights sharing scheme based on redistributing time-based rights among consumer devices. We divide the allowed time into sum of unit-times and allow consumers to redistribute these unit-times to their devices freely. Therefore the consumers can decide the number of devices that share content, unlike the domain model [7,10,13,15,16], and can decide the time-quota of each device. To prevent illegal content sharing, our scheme limits the number of time-units that a consumer has. Moreover, we assign a unique number to each time-unit; hence the illegal time-units which are created by a compromised device can be found easily. Our scheme neither needs additional hardware [9] nor secure and globally synchronized clocks, and does not cause doubtful decisions for illegal sharing of content [17].

### 1.2 Our contributions

To the best of our knowledge, the proposed time-based rights redistribution scheme for personal sharing of digital content is the first attempt to use time as a limit on content sharing among consumer devices. Also, our scheme can increase consumers' convenience because it is a fully distributed scheme. Our scheme does not need a central device to manage content sharing and devices do not need to be connected with each other to manage the content sharing. Only two devices, a requesting device and a responding device, need to be connected to redistribute the remaining time-based rights. Other devices of the consumer may attend the redistribution process as monitoring devices. We have also shown that our scheme can be realized by explaining data structures, protocols, and algorithms of our scheme and implementing the fundamental protocols.

### 1.3 Paper organization

The remainder of this paper is as follows. In Section 2, we briefly review the related work. In Section 3, we discuss the assumptions of this paper. In Section 4, we introduce our scheme in detail. And in Section 5 we analyze the security features of our scheme and compare our scheme with the related work. In Section 6, we explain our implementation. Finally, we conclude our paper and discuss future work in Section 7.

## 2 Related work

In this section, we discuss the related work in detail. We classify the related work according to the methods of preventing illegal content sharing: restricting the number of devices that share content [7,10,13,15,16] and restricting the simultaneous usage of content [9,17].

### 2.1 Domain model

In the domain model, a consumer can create a group that include his/her devices, called a domain [7,10,13, 15,16]. The consumer can use his/her content with any

devices that belong to the domain. The number of devices that belong to a domain is restricted to prevent illegal content sharing. If not, then a consumer can create a domain which includes thousands of his/her devices or even other peoples.

Every domain model manages information such as the number of devices that belong to the domain and the specific information of each device (e.g., a device ID). This information should be synchronized globally for data consistency. We can classify domain models according to how they manage such information: a centralized domain model, a local domain model, and a distributed domain model. The centralized domain model manages the domain information in a centralized server; thus, global synchronizations of the domain information can be kept [13]. However, it requires every device to be connected to the remote domain manager to join or to leave from a domain. The local domain model permits one of the consumer devices to be a local domain manager [7,10,16]. Hence, devices do not need to communicate with the remote server when joining to or leaving from a domain. A compromised local domain manager is more dangerous than a compromised device because it can create a domain including infinite devices. Hence, the local domain manager should be resistant against attacks. In the distributed domain model, all of a consumer's devices manage the domain information in cooperation; thus, a domain manager is not required [15]. However, this model requires every device to be connected with each other continuously for data consistency.

### 2.2 Simultaneous usage restriction schemes

Simultaneous usage restriction schemes restrict the multiple usage of content at the same time to prevent illegal content sharing [9,17]. These schemes assume that the simultaneous usage only occurs when consumers share their content with others illegally. One of these schemes is the smart card-based scheme [9]. It requires consumers to have smart cards which store consumers' keys and rights objects. Consumers can shift the smart cards among their devices to use content; thus, the number of devices that can use their content is unrestricted. But it means every device must support the smart card.

Another scheme is the log-based simultaneous usage detection scheme [17]. In this scheme, every device logs the start-time and end-time of the content use, and later if devices are connected to other devices, then they check whether a time overlap has occurred to find the simultaneous usage. This scheme requires every device to have a secure and globally synchronized clock. In addition, a service provider should analyze more precisely to determine whether the simultaneous usage is illegal sharing or not because a simultaneous usage can occur by a consumer's mistake.

## 3 Assumptions

We have the following assumptions to validate our scheme:

- A device has a tamper-proof DRM agent that enforces a DRM system and a secure storage that stores secrets to maintain the DRM system. The DRM agent and the secure storage can prevent and detect several attacks [7,13,16,17,19].
- A service provider has some schemes to find compromised devices such as the fingerprinting scheme [8] or the traitor tracing scheme [2].
- A device has a device certificate for device authentication and revocation. [7,13,17]
- A device certificate revocation list is delivered transparently when devices are connected or it can be attached to the protected content [7,13,15–17].
- Devices share a user certificate of their owner for the consumer and owner authentication processes.
- A device has the same-type of DRM agent. We do not consider the interoperability problem [6].

Almost every DRM system depends on the robustness of the DRM agent in a device. However, it is hard to implement a perfectly secure DRM agent against device compromise. Thus, without the detection scheme of compromised devices, DRM systems cannot guarantee their security. Therefore the first and second assumptions are needed and other researchers also assume them [7,13,16,17,19]. Many DRM systems such as OMA DRM also require a device certificate for device authentication and validation [7,13,17]. Also, to revoke a compromised device, systems based on PKI usually use certificate revocation lists [7,13,17]. Thus, assuming device certificate and certificate revocation process is reasonable. Because our DRM system is a consumer-based DRM system, a consumer authentication scheme is needed to protect illegal content sharing among different consumers. We assume a user certificate for the consumer authentication because it is a strong authentication scheme and has consistency with the device certificate. But other schemes such as a shared password can also be used for consumer authentication. Since DRM interoperability is another big issue and it complicates the explanation of the proposed content sharing scheme, we assume every device has the same-type of DRM agent to limit the problem scope.

## 4 Proposed scheme

### 4.1 Overview

Restricting the number of a consumer's devices that can share the consumer's content may decrease the consumer's convenience. However, to prevent illegal content sharing, we need some restrictions on content sharing. We suggest a new scheme that allows some amounts of time to use content in a specific period to a consumer and then allows the consumer to redistribute his/her content-use-time to his/her devices freely. It is similar to the accumulated time constraint [5,12,18]. For example, a consumer can use his/her content in a $\text{Device}_1$ when the device has a rights object for the content and sufficient content-use-time. If not, then the consumer can transfer other device's (say $\text{Device}_2$) remaining content-use-time to $\text{Device}_1$. Other devices of the consumer or a rights issuer do not need to participate in the content-use-time transfer process; thus, the proposed scheme is a distributed scheme. The restriction to the content-use-time for content sharing does not bring the consumer's inconvenience because the upper bound of content-use-time in a specific period is determined (e.g., a person cannot use content more than 24 hours in a day.). Thus, if a service provider gives sufficient time (e.g., more than 24 hours for a day) to a consumer, then the consumer can utilize his/her content-use-time sufficiently. We divide the content-use-time into several same-length-units for easy distribution and management. We name the divided time `dt-token`. Each `dt-token` has a unique number to prevent illegal usage and sharing (see Fig. 1). The `dt-token` redistribution process can be performed manually or automatically. Each device has a threshold value and redistribution ratio for automatic `dt-token` redistribution.

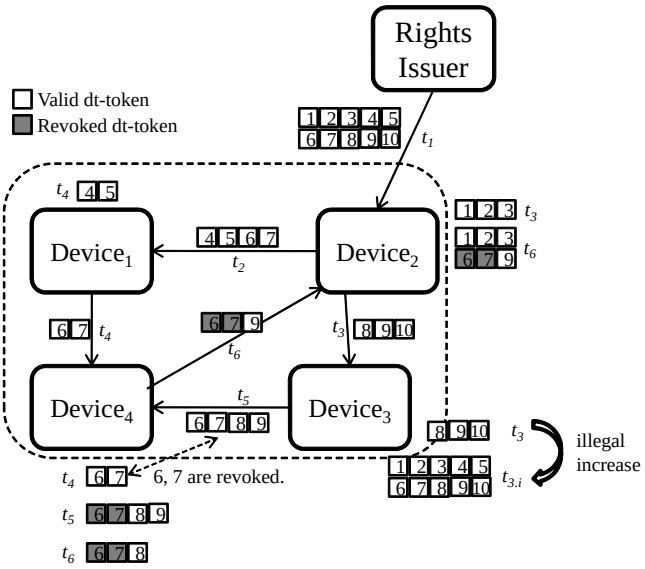



**Fig. 1** Conceptual procedures of the proposed scheme. When a consumer obtains $n(n = 10)$ `dt-token`s from a rights issuer using $\text{Device}_2$ ($t_1$), he/she can distribute them to other devices ($t_2$,$t_3$,$t_4$). Because each `dt-token` has a unique number, even if a compromised device $\text{Device}_3$ creates and distributes illegal `dt-token`s to $\text{Device}_4$, $\text{Device}_4$ can revoke the illegal `dt-token`s by checking duplications ($t_5$). The revocation information is delivered to other devices when they are connected ($t_6$).

### 4.2 Representations

This section presents basic representations for exchanging and managing the rights object and `dt-token` state information. Table 1 shows the notations of this paper.

#### *4.2.1 Rights object structure*

An $RO$ has the following structures (see Fig. 2). The $RO$ embeds a signature of $RI$ and $CEK$ is protected by the user public key $u_C$. Thus, the $RO$ can be stored in a normal storage and a consumer can backup the $RO$ to secondary storage.

$$RO_C = Sig_{v_{RI}}(ID_{RO}, ID_{Content}, rights, SE_{REK}(CEK), PE_{u_C}(REK), \texttt{dt-token}_{len}, \#\texttt{dt-token}\text{s})$$

| | |
|---|---|
| $ID_{RO}$ | Identifier of $RO$ |
| $ID_{Content}$ | Identifier of content related to $RO$ |
| $rights$ | Consumer's rights to content |
| $SE_{REK}(CEK)$ | Encrypted $CEK$ with $REK$ |
| $PE_{u_C}(REK)$ | Encrypted $REK$ with $u_C$ |
| $\texttt{dt-token}_{len}$ | Unit-length of a `dt-token` |
| $\#\texttt{dt-token}$s | The number of `dt-token`s |

**Fig. 2** Rights object structure

#### *4.2.2* `dt-token` *state information structure*

`dt-token` state information has the following structures (see Fig. 3). Each $s_i$ requires 2 bits to represent its state.

$$S_D = (ID_D, ID_{RO}, SE_{REK}(s_1, \ldots, s_{\#\texttt{dt-token}\text{s}}))$$

| | |
|---|---|
| $ID_D$ | Identifier of a device managing $S_D$ |
| $ID_{RO}$ | Identifier of $RO$ |
| $s_i$ | State of an individual `dt-token` |
| | 00: A device $D$ owns $s_i$ which is *unused* |
| | 01: $s_i$ is *already used* |
| | 10: $D$ has *no ownership* to $s_i$ |
| | 11: $s_i$ is *revoked* |

**Fig. 3** `dt-token` state information structure

**Table 1** Notations

| Symbol | Meaning |
|---|---|
| $SE_K()$ | Symmetric key encryption with key $K$ |
| $PE_K()$ | Public key encryption with key $K$ |
| $Sig_K()$ | Private key signing with key $K$ |
| $ID_E$ | Unique identifier of $E$ |
| $u_E, v_E$ | Public/private key pair of $E$ |
| $Cert_E$ | Certificate of $E$ |
| $N_E$ | Random nonce generated by $E$ |
| $REK$ | Rights object encryption key |
| $CEK$ | Content encryption key |
| $SK, RK$ | Session key and random key |
| $RI, RO$ | Rights issuer and rights object |
| $C$ | Consumer |
| $\texttt{dt-token}_{rem_E}$ | The number of remaining `dt-token`s of $E$ |
| $\texttt{dt-token}_{thr_E}$ | Threshold vale of `dt-token`s of $E$ |
| $\texttt{dt-token}_{len_E}$ | Unit-length of a `dt-token` of $E$ |
| $S_E$ | `dt-token` state information of $E$ |
| $R_E$ | Relative redistribution ratio of $E$ |
| `Verify`$(M)$ | Verify a signature on $M$ |
| `StateSetup`$(RO, a)$ | Set up `dt-token` state information with an $RO$, $a \in \{full, empty\}$ |
| `StateUpdate`$(S_i, S_j; R_i, R_j)$ | Update `dt-token` state information |
| `StateChecking`$(\mathbf{S})$ | Check a set of `dt-token` state info. |

This information should be stored in secure storage to prevent illegal modifications.

#### *4.2.3 Translating constraints to* `dt-token`s

We should translate time-based or time-translatable constraints to `dt-token`s to generalize our scheme. We focused on four types of constraints: unlimited, period-limited, accumulated-time-limited, and use-count-limited constraints. If a consumer has an $RO$ that has unlimited constraints, then he/she can use content belonging to the $RO$ permanently. A simple solution to translate the unlimited constraint to the `dt-token`s is giving infinite `dt-token`s. However, this is undesirable because an illegal consumer can distribute the infinite `dt-token`s to other consumers' devices infinitely. To solve this problem, we consider the unlimited constraint as the infinite period-limited constraint. For example, we can translate the unlimited constraint to the constraint that gives 720 hours per month where each month is repeated continuously. Because each month's `dt-token`s are limited, an illegal consumer cannot distribute them, infinitely. The period-limited and accumulated-time-limited constraints are almost the same except that the period-limited constraints have a starting point (e.g., from 12/01/200X to 12/31/200X). These two constraints are easily translated to the `dt-token`s. The use-count-limited constraint restricts the number of content usage. If the content is a multimedia file that has a running time (e.g., audio or video), then we can represent the use-count as time by multiplying the use-count with the content-length. However, the running time of other data such as an image and an application is difficult to guess. Therefore a service provider has to decide a policy to translate the use-count-limited constraint onto such type of data.

### 4.3 Protocols and algorithms

#### *4.3.1 Device initialization*

When a consumer obtains a new device, he/she should set three values in the device. The first one is a user certificate. The consumer can copy the user certificate in another device using a secure channel or obtain the user certificate from the certificate authority online or offline. The user certificate is protected by a device private key and stored in a secure storage. The other two values are the threshold value of the number of `dt-token`s and the redistribution ratio of `dt-token`s. These two values are determined by the consumer and are used in the automatic `dt-token` redistribution protocol. When a device's remaining `dt-token`s are lower than its threshold value then it runs the `dt-token` redistribution protocol to obtain `dt-token`s from other devices. The reassignment of the remaining `dt-token`s of the two devices is determined by two devices' redistribution ratios.

#### *4.3.2 Rights object acquisition protocol (ROAP)*

A consumer can obtain an $RO$ from $RI$ by the rights object acquisition protocol (see Fig. 4). In our scheme, we assume that SSL is used to authenticate a device and establish a secure session. Thus, the device authentication process is simplified. We also simplify the billing process to reduce the complexity of the protocol. When the protocol is succeeded and a device acquires an $RO$, the device creates the *full* `dt-token` state information which is composed of *unused* `dt-token`s. A consumer can transfer the $RO$ to other devices of him/her to create `dt-token` state information. Devices which do not receive an $RO$ from $RI$ directly create the empty `dt-token` state information which is composed of *no ownership* `dt-token`s. If they create the *full* `dt-token` state information illegally, then it can be detected by the `dt-token` redistribution protocol.

#### *4.3.3* `dt-token` *redistribution protocol (DTRP)*

When a device's remaining `dt-token`s are too small (i.e., lower than its threshold value), a consumer can

| Device authentication |
|---|
| $D \leftrightarrow RI$:*Authenticate each other and establish a secure session using device certificates (e.g., SSL)* |
| Consumer authentication |
| $D \leftarrow RI$:$SE_{SK}(N_{RI})$ |
| $D \rightarrow RI$:$SE_{SK}(Cert_C, Sig_{v_C}(N_{RI}))$ |
| *RO* issuing and `dt-token` state setup |
| $D \rightarrow RI$:$SE_{SK}(ID_{Content}, billing)$ |
| $RI$ :$RO \leftarrow Sig_{v_{RI}}(ID_{RO}, ID_{Content}, \ldots, PE_{u_C}($ $REK), \ldots)$ |
| $D \leftarrow RI$:$SE_{SK}(RO)$ |
| $D$ :`Verify`$(RO)$ |
| $D$ :$B_D \leftarrow$ `StateSetup`$(RO, full)$ |

**Fig. 4** Rights object acquisition protocol

initiate the `dt-token` redistribution protocol manually or the device can initiate it automatically (see Fig. 5). A requesting device broadcasts a device finding message to recognize connectible devices and determine their remaining `dt-token`s. The requesting device receives reply messages and then selects a device that has sufficient `dt-token`s. Then two devices perform the device authentication process followed by the owner authentication process. If the authentication processes are successful, then they exchange the `dt-token` state information and redistribution ratios to update their `dt-token` state information (see Fig. 6). A device checks the duplication of *unused* `dt-token`s with other's `dt-token` state information and revokes the duplicating `dt-token`s. Next, the device reassigns remaining *unused* `dt-token`s according to the redistribution ratios of its and the others. The accumulated `dt-token` state information is used to verify another device's validity by checking the uniqueness of `dt-token`s and distribute the updated `dt-token` state information to other listening devices.

At last, the two devices broadcast their updated `dt-token` state information to check the validity of the updated `dt-token` state information together (see Fig. 7). Devices that have received the updated `dt-token` state information run the `dt-token` state check algorithm with its and the others' `dt-token` state information to check the duplication of *unused* `dt-token`s and revoke the duplicated `dt-token`s. Note that there is no need all devices to be connected at the same time. Devices have not received the updated state information can check it later when they are connected with the devices have received that information. If a device is directly linked to another device to perform DTRP, then the device finding part is omitted.

#### *4.3.4* `dt-token` *checking protocol (DTCP)*

Devices periodically perform DTCP to check the validity of their `dt-token` state information (see Fig. 8).

| Device finding and authentication |
|---|
| $D_i \Rightarrow *$ :$Cert_{D_i}, Sig_{v_C}(Sig_{v_{D_i}}(N_{D_i}, ID_{RO}))$ |
| $D_k \Rightarrow *$ :$Cert_{D_k}, Sig_{v_C}(Sig_{v_{D_k}}(N_{D_i}, N_{D_k}, ID_{RO},$ $PE_{u_C}($`dt-token`$_{rem_{D_k}}$, `dt-token`$_{thr_{D_k}}$$)))$ |
| ($D_k \in$ *a set of all listening devices excepts* $D_i$) |
| $D_i$ :*Selects* $D_j$ *that has sufficient* `dt-token`s |
| $D_i \leftrightarrow D_j$:*Establish a secure session using device certificates (e.g., SSL)* |
| `dt-token` state exchange and update |
| $D_i \rightarrow D_j$:$SE_{SK}(ID_{RO}, S_{D_i}, R_{D_i})$ |
| $D_i \leftarrow D_j$:$SE_{SK}(ID_{RO}, S_{D_j}, R_{D_j})$ |
| $D_i$ :$S'_{D_i}, S'_{D_{ij}} \leftarrow$ `StateUpdate`$(S_{D_i}, S_{D_j}; R_{D_i}, R_{D_j})$ |
| $D_j$ :$S'_{D_j}, S'_{D_{ij}} \leftarrow$ `StateUpdate`$(S_{D_i}, S_{D_j}; R_{D_i}, R_{D_j})$ |
| Updated `dt-token` state broadcasting and checking |
| $D_i \Rightarrow *$ :$Cert_{D_i}, Sig_{v_C}(Sig_{v_{D_i}}(N_{D_i}, \mathbf{N}_{D_k}, ID_{RO},$ $PE_{u_C}(S'_{D_{ij}})))$ |
| $D_j \Rightarrow *$ :$Cert_{D_j}, Sig_{v_C}(Sig_{v_{D_i}}(N_{D_i}, ID_{RO}, PE_{u_C}($ $S'_{D_{ij}})))$ |
| $D_k$ :$D_i$'s $S'_{D_{ij}} \stackrel{?}{=} D_j$'s $S'_{D_{ij}}$ |
| $D_k$ :`StateChecking`$(S_{D_k}, S'_{D_{ij}})$ |
| ($D_k \in$ *a set of all listening devices including* $D_i$ *and* $D_j$) |

**Fig. 5** `dt-token` redistribution protocol

A device requests other devices' `dt-token` state information and then performs the `dt-token` state check algorithm with its own and the received `dt-token` state information. If a device is directly linked to the other device (to transfer content, *RO*s, and etc), then the two devices also perform DTCP.

#### *4.3.5 Device leave protocol (DLP)*

When a consumer wants to remove all remaining `dt-token`s from a device, he/she can set zero redistribution ratios to the device and run DTRP with another device to transfer the remaining `dt-token`s to the other device.

## 5 Analysis

### 5.1 Attack classifications and countermeasures

To analyze the security features of our DRM system, we classify the possible attacks to our system into four types: content attack, secret key attack, rights object attack, and system attack.

#### *5.1.1 Content attack*

Content attack is an attempt to reveal raw content. It includes device attacks to obtain decrypted digital content from internal channels (e.g., memory or bus) and the analog-hole attacks [3] to record analog signals

INPUT : dt-token states and redistribution ratios of two devices: $S_{D_i}, S_{D_j}; R_{D_i}, R_{D_j}$
OUTPUT : an updated dt-token state of device $h$ and accumulated state: $S'_{D_h}, S'_{D_{ij}}$
$N \leftarrow$ total number of dt-tokens
Create an empty (*no ownership*) dt-token state: $S'_{D_h}, S'_{D_{ij}}$
for $l \leftarrow 1$ to $N$ do
  if $S_{D_j,l} =$ *unused* then ; $S_{D_i,l}$ is $l$th dt-token of $S_{D_i}$
    if $S_{D_j,l} =$ *no ownership* then $S'_{D_{ij},l} \leftarrow$ *unused*
    else $S'_{D_{ij},l} \leftarrow$ *revoked*
  else if $S_{D_j,l} =$ *already used* then
    if $S_{D_j,l} =$ *no ownership* or *already used* then
      $S'_{D_{ij},l} \leftarrow$ *already used*
    else $S'_{D_{ij},l} \leftarrow$ *revoked*
  else if $S_{D_i,l} =$ *no ownership* then $S'_{D_{ij},l} \leftarrow S_{D_j,l}$
  else $S'_{D_{ij},l} \leftarrow$ *revoked*
$S'_{D_h} \leftarrow S'_{D_{ij}}$
$U \leftarrow$ the number of *unused* dt-tokens in $S'_{D_{ij}}$
$assign_i \leftarrow (R_{D_i}/(R_{D_i} + R_{D_j}))U$
$assign_j \leftarrow U - assign_i$
$m \leftarrow 1$ // $m$ should not exceed $N$
while $assign_i \neq 0$ do  if $S'_{D_{ij},m} =$ *unused* then
    if $h = j$ then $S'_{D_h,m} \leftarrow$ *no ownership*
    $assign_i \leftarrow assign_i - 1$
  $m \leftarrow m + 1$
while $assign_j \neq 0$ do
  if $S'_{D_{ij},m} =$ *unused* then
    if $h = i$ then $S'_{D_h,m} \leftarrow$ *no ownership*
    $assign_j \leftarrow assign_j - 1$
  $m \leftarrow m + 1$
return $S'_{D_h}, S'_{D_{ji}}$

**Fig. 6** dt-token state update algorithm

INPUT : dt-token states of $M$ devices: $S_{D_1}, \ldots, S_{D_M}$
OUTPUT : an updated dt-token state of device $h$: $S_{D_h}$
$M \leftarrow$ total number of devices
$N \leftarrow$ total number of dt-tokens
for $l \leftarrow 1$ to $M$ do
  if $i \neq h$ then
    for $j \leftarrow 1$ to $N$ do
      if $S_{D_i,j} =$ *unused* or *already used* then
        if $S_{D_h,j} \neq$ *no ownership* then $S_{D_h,j} \leftarrow$ *revoked*
      else if $S_{D_i,j}$ is *revoked* then $S_{D_h,j} \leftarrow$ *revoked*
return $S_{D_h}$

**Fig. 7** dt-token state check algorithm

from output channels (e.g., output ports to a speaker or a display). We can use more robust devices that include hardware and software based tamper-resistance properties to prevent the device attack [19]. We also use detection schemes such as the fingerprinting scheme and traitor tracing scheme to detect the device and analog-hole attack [2,8] and then revoke the detected devices with the device certificate revocation [13,16,17].

| dt-token state checking |
|---|
| $D_i \Rightarrow *$ :$Cert_{D_i}, Sig_{v_C}(Sig_{v_{D_i}}(N_{D_i}, ID_{RO},$ $PE_{u_C}(S'_{D_i})))$ |
| $D_j \Rightarrow D_i$:$Cert_{D_j}, Sig_{v_C}(Sig_{v_{D_j}}(N_{D_i}, N_{D_j}, ID_{RO},$ $PE_{u_C}(S'_{D_j})))$ ($D_j \in$ *a set of all listening devices*) |
| $D_i$ :StateChecking$(S_{D_i}, S_{D_j})$ |

**Fig. 8** dt-token checking protocol

### *5.1.2 Secret key attack*

Secret key attack is an attempt to reveal the secret key that can decrypt the protected content. In our scheme, all secret keys that can reveal the protected content are protected by a device private key. The device private key is secure until the device is compromised. Moreover, any schemes against the content attack such as the tamper-resistance hardware and software [19], the fingerprinting scheme [8], and the traitor tracing scheme [2] are also applicable for the prevention of secret key attack.

### *5.1.3 Rights object attack*

Rights object ($RO$) attack is an attempt to ignore, modify, replay, and forge an $RO$ to use content illegally. Ignoring $RO$ requires a compromised device, which can be prevented and detected by the device securing schemes [2, 8,19]. Moreover, the modification and forgery of an $RO$ are difficult because the $RO$ has an embedded signature of the rights issuer [13]. Lastly, the replay attack of the stateful $RO$ such as the accumulated-limited or use-count limited $RO$ can be prevented by using the $RO$ revocation list or discarding the backup of the stateful RO [13].

### *5.1.4 System attack*

System attack is an attempt to compromise an entire DRM system in collusion with a few compromised devices of that system. This attack is one of important attacks to every rights sharing scheme. If there are $n$ devices in a system, the desired robustness against the system attack is $n$ times more than the robustness against the device attack. Namely, if the expected effort to compromise a device is $e$ then we expect that the effort to compromise $n$ devices is $ne$. However, every rights sharing scheme cannot assure this expectation because devices of the system have dependencies. For example, if an illegal consumer compromises the local domain manager [7,10,16] or smart card [9] then he/she can use other uncompromised devices illegally.

Our scheme also suffers from system attacks. If a compromised device creates fake dt-tokens and distributes them to other uncompromised devices, then a consumer can use uncompromised devices illegally. If our scheme does not have countermeasures against a system attack, then our scheme's robustness against the system attack is almost $e$ because each device has a same role.

We can use countermeasures against the *RO* attack to deal with the system attack such as embedding signatures and maintaining revocation lists because possible attacks to the dt-tokens are similar to the attacks to *RO*. In our scheme, the number of dt-tokens and a unit-length of a dt-token are signed by *RI*; thus, a consumer cannot illegally increase them. Moreover, we assign a unique number, which is similar to the rights object identifier, to each dt-token. If a compromised device has illegal dt-tokens which are duplicates of other uncompromised devices' dt-tokens, then the illegal dt-tokens are detected and revoked by other uncompromised devices when it interacts with them. Because a device's dt-token state information is stored in secure storage, a consumer cannot know the unique numbers assigned to the device's dt-tokens until he/she compromises the device. Thus, even if the consumer compromises a device and creates illegal dt-tokens, it is hard to distribute the illegal dt-tokens to other uncompromised devices because he/she cannot determine whether the illegal dt-tokens are duplicates of other uncompromised devices' dt-tokens. Hence, although there is no centralized device or server that is in charge of the illegal dt-token problem, illegal dt-tokens and compromised devices are eliminated from content sharing. Therefore system attacks are hard to succeed in our scheme.

## 5.2 Storage requirements of dt-token state information

dt-token state information should be stored in secure storage to prevent modifications and replay attacks. However, secure storage is more expensive than normal storage. To minimize the storage requirement, we assign 2 bits to represent the state of each dt-token. Table 2 shows the storage requirements of 30 days (720 hours) rights according to the unit-length of dt-tokens. The one second-length dt-token requires a large amount of storage but still less than 1 MiB.

**Table 2** Storage requirements of dt-token state information

| $\texttt{dt-token}_{len}$ | 1 minute | 10 seconds | 1 second |
|---|---|---|---|
| Storage requirement for 30 days rights | 10,800B | 64,800B | 648,000B |

## 5.3 Comparisons with related work

We compare our scheme with the related work according to the following criterion: the type of restrictions used to prevent illegal sharing, whether a central manager is needed or not, how to register a device to a content sharing group, how to restrict the simultaneous usage, and how to register a $(n+1)$th device (see Table 3).

### *5.3.1 Restriction*

To restrict heavy content sharing, all schemes restrict the number of devices, the simultaneous usage, and time for content usage, respectively.

### *5.3.2 Central manager and device registration*

Some domain models and the log-based scheme require a central manager to manage device and/or clock information but other schemes including our scheme do not require a central manager.

### *5.3.3 Simultaneous usage*

The smart card-based scheme and the log-based scheme restrict the simultaneous usage. However, the domain model and our scheme permit the simultaneous usage with some restrictions: the domain model restricts the simultaneous usage up to $n$ devices and our scheme restricts the simultaneous usage according to the amount of time of content usage.

### *5.3.4 $(n+1)$th device registration*

If consumers want to register their new devices to their full domains, then they must erase some old devices to register the new ones. However, other schemes including ours do not need this process for a new registration.

### *5.3.5 System attack prevention*

Robustness against system attacks of a domain with a domain manager and smart card-based scheme depends on the robustness of the domain manager and smart card, respectively (centralized approaches). In the distributed domain model, all device monitors each other

continuously. Similarly, in the log-based scheme and our scheme, devices check each other sometimes by checking time overlap and duplications of `dt-token`s, respectively (distributed approaches).

It is hard to decide which methods are more robust against system attacks. If the robustness of a special device such as a domain manager and smart card is higher than the sum of robustness of normal devices, then the centralized approaches are more robust. If not, then the centralized approaches can suffer a single-point-of-failure problem. The distributed approaches also suffer a problem when some devices are not connected with each other. In that case, some parts of their information cannot be verified.

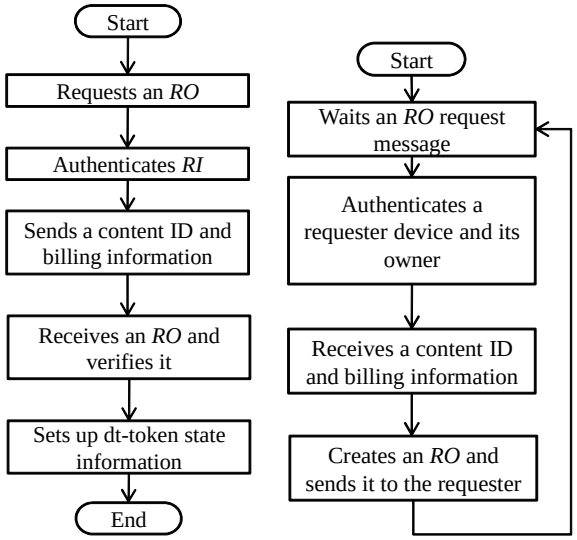
## 6 Implementation

We implement the two protocols, the rights object acquisition protocol and the `dt-token` redistribution protocol, on a Linux system with OpenSSL library [14, 20]. These two protocols are implemented in two programs: the rights issuer program and the DRM agent program (a device). In our implementation, the rights issuer program is a simple program that authenticates a device and the owner of the device, receives a content identifier from the device, and then issues a rights object bind to the owner.

To authenticate a device and a rights issuer mutually and to establish a secure session, we use SSL directly to establish secure channels between the device and the rights issuer and between devices. We also use the crypt library of OpenSSL for data encryptions and digital signatures according to our protocols. Devices use TCP and UDP for point-to-point communications and broadcast communications, respectively.

### 6.1 Rights object acquisition protocol

The rights issuer waits for an *RO* request message continuously. When a device request an *RO*, the device and the rights issuer authenticate them mutually and authenticate the owner of the device. Finally, the device requests an *RO* corresponding to a content ID and then the rights issuer issues the *RO* to the owner of the device (see Fig. 9).




(a) Device side (b) Rights issuer side

**Fig. 9** Flow charts of the rights object acquisition protocol

### 6.2 `dt-token` redistribution protocol

When every device is started, the respondent process is run in the background. Once a device's `dt-token` becomes lower than its threshold, the requester process is run to obtain `dt-token`s from other devices. The requester broadcasts a device finding message and waits for reply messages in several $t$ seconds. If the request cannot receive a reply message then the requester process is terminated. Once the waiting time is over, the requester selects a device that has sufficient `dt-token`s in the message storage that stores received replies and performs the device and owner authentication processes. Next, they exchange the `dt-token` state information and redistribution ratios and update their `dt-token` state information. Finally, they broadcast the updated `dt-token` state information and wait for the other's updated `dt-token` state information. If a requester or respondent cannot receive the other's updated `dt-token` state information in $s$ seconds, then they rollback the `dt-token` state; otherwise, they perform the `dt-token` state check process (see Fig. 10). The respondent process waits for three types of messages: the device finding message, `dt-token` exchange request message, and updated `dt-token` state information message. When it receives the device finding message, it replies the number of its remaining `dt-token`s, and when it receives the `dt-token` exchange request message, it performs the `dt-token` exchange and update process with a requester. Finally, when it receives the updated `dt-token` state information message, it waits for another updated `dt-token` state information message and performs the `dt-token` state check process with its own and the other two `dt-token` state information (see Fig. 11).

**Table 3** Comparison results

| | Domain with manager [7,10,13,16] | Distributed domain [15] | Smart card-based [9] | Log-based [17] | Our scheme |
|---|---|---|---|---|---|
| Restriction | The number of devices | | Simultaneous usage | | The amount of time |
| Central manager | O | X | X | O | X |
| Device registration | Register to a manager | Register to all member | none | Register to a manager | Set a user certificate |
| Simultaneous usage | Up to $n$ devices | | A fixed number of devices | | Restricted according to the amount of time |
| $(n+1)$th device registration | Erase old one and then register new one | | Transfer a smart card | Register new one | Transfer `dt-token`s |
| System attack prevention | Robust domain manager | Monitor each other | Robust smart card | Check time overlap | Check uniqueness of `dt-token`s |

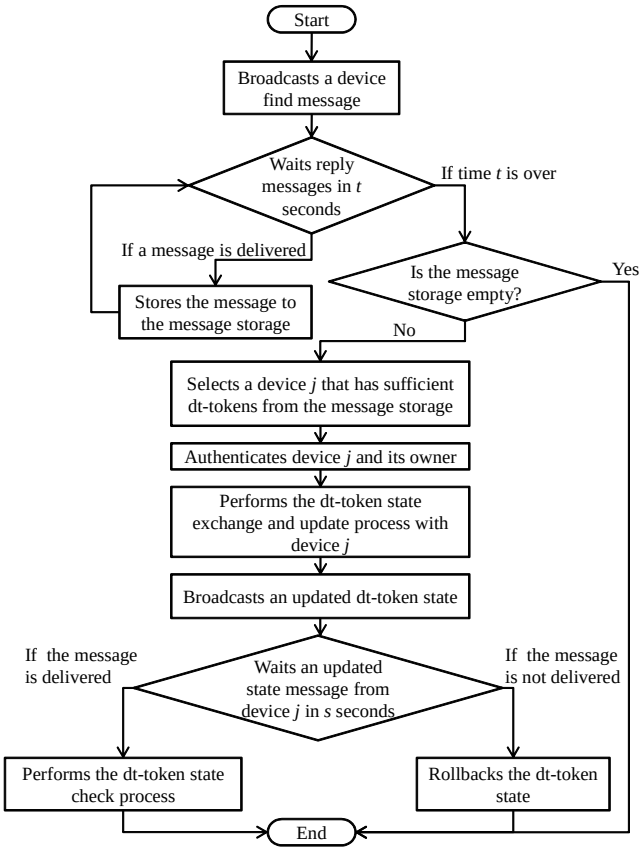



**Fig. 10** Flow chart of requester side of `dt-token` redistribution protocol

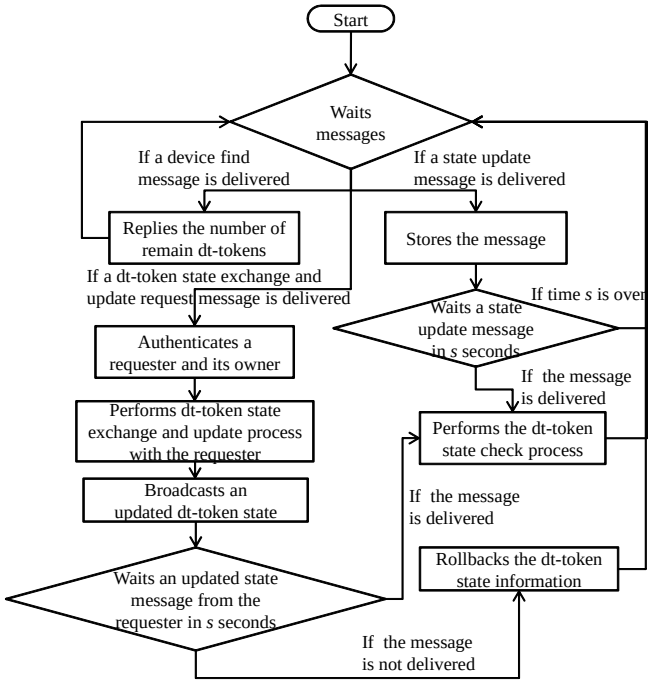



**Fig. 11** Flow chart of respondent side of `dt-token` redistribution protocol

### 6.3 Performance consideration for `dt-token` redistribution protocol

To find neighboring connectible devices, `dt-token` redistribution protocol (DTRP) uses broadcast communications. However, in a real environment, broadcast communications are hard to apply because of the unguaranteed response time. Especially, in wireless networks, transmission delay is longer than that of the wired network due to the signal collision. Thus, the waiting time $t$ for reply messages of the device finding message should be decided carefully for gathering information of connectible devices sufficiently and reducing the overall time of DTRP. $t$ can be estimated as $t = n(w + s) + p$. Here, $n$ is the expected number of neighboring connectible devices, $w$ is the expected waiting time for transmission, $s$ is the transmission time for a single message in DTRP, and $p$ is the processing time for a single message in DTRP. To determine $n$, a device which will initiate DTRP may broadcast probing messages to recognize the number of neighboring devices before it performs DTRP. $w$ depends on the number of devices that may use a shared channel and the traffic rate of the shared channel [1]. Thus, using $n$ and the

traffic rate, the requesting device can estimate $w$. $s$ depends on the size of a message of DTRP. We assume the size of each message of DTRP is same to simplify formulae. Because the size of each message of DTRP is less than several megabytes, even if we set $s$ as the transmission time of the largest message, the waiting time is not extremely increased. Similarly, we assume the processing time $p$ for each message is same. According to the formula to estimate $t$, the length of $t$ heavily depends on $n$. Because our scheme is designed for rights sharing among consumer's devices, in general, $n$ is not a huge number. Thus, the length of $t$ may not be extended to longer time. Finally, we can estimate the overall time $T$ for performing DTRP as $T = 3(w + s) + t + 7p + \sigma = (n + 3)(w + s) + 8p + \sigma$. Here, $\sigma$ is the required time for establishing an SSL session.

## 7 Concluding remarks

In this paper, we presented a novel rights sharing scheme based on the redistribution of usage amounts of time. Consumers can freely use content in their devices by redistributing the usage amounts of time between the devices. Moreover, our scheme only needs local synchronizations among participating devices in the redistribution process of usage amounts of time. In addition, our scheme does not require additional hardware or a globally synchronized secure clock. We evaluated our scheme by implementing essential protocols. In the future we will adapt our scheme to a working DRM system to evaluate it in a real environment.

## References


1. Cena, G., Bertolotti, I.C., Valenzano, A., Zunino, C.: Analysis of response times in 802.11 industrial networks. In: Proceedings of 2007 5th IEEE International Conference on Industrial Informatics, pp. 195–200 (2007)
2. Chor, B., Fiat, A., Naor, M., Pinkas, B.: Tracing traitors. IEEE Transactions on Information Theory (3), 893–910 (2002)
3. Diehl, E., Furon, T.: ©watermarking: closing the analog hole. In: Proceedings of the 2003 IEEE International Conference on Consumer Electronics, pp. 52–53 (2003)
4. van den Heuvel, S., Jonker, W., Kamerman, F., Lenoir, P.: Secure content management in authorised domains. In: Proceedings of International Broadcasting Conference 2002, pp. 467–474 (2002)
5. ISO/IEC FDIS 21000-5:2003(E): Information technology - multimedia framework - part 5: rights expression language
6. Koenen, R.H., Lacy, J., Mackay, M., Mitchell, S.: The long march to interoperable digital rights management. Proceedings of the IEEE (6), 883–897 (2004)
7. Koster, P., Kamperman, F., Lenoir, P., Vrielink, K.: Identity based DRM: personal entertainment domain. In: Proceedings of Communications and Multimedia Security, vol. LNCS 3677, pp. 42–54 (2005)
8. Kundur, D., Karthik, K.: Video fingerprinting and encryption principles for digital rights management. Proceedings of the IEEE (6), 918–932 (2004)
9. Lee, W., Wu, W., Chang, C.: A portable DRM scheme using smart cards. Journal of Organizational Computing and Electronic Commerce (3), 247–258 (2007)
10. Marlin Developer Community: Marlin architecture overview (2006). `http://www.marlin-community.com`
11. Mulligan, D.K., Han, J., Burstein, A.J.: How DRM-based content delivery systems disrupt expectations of "personal use". In: Proceedings of the 3rd ACM Workshop on Digital Rights Management, pp. 27-38, Oct. 2003., pp. 27–38 (2003)
12. Open Mobile Alliance: DRM rights expression language - candidate version 2.1 (2007). `http://www.openmobilealliance.org`
13. Open Mobile Alliance: DRM specification - candidate version 2.1 (2007). `http://www.openmobilealliance.org`
14. OpenSSL Project: `http://www.openssl.org`
15. Pestoni, F., Lotspiech, J.B., Nusser, S.: xCP: peer-to-peer content protection. IEEE Signal Processing Magazine (2), 71–81 (2004)
16. Popescu, B.C., Crispo, B., Tanenbaum, A.S., Kamperman, F.: A DRM security architecture for home networks. In: Proceedings of the 4th ACM Workshop on Digital Rights Management, pp. 1–10 (2004)
17. The Digital Media Project: Approved document No. 3 - Technical specification: interoperable DRM platform, version 3.0 (2007). `http://www.dmpf.org`
18. The Open Digital Rights Language (ODRL) Initiative: `http://www.odrl.net`
19. Trusted Computing Group: `http://www.trustedcomputinggroup.org/home`
20. Viega, J., Messier, M., Chandra, P.: Network security with OpenSSL. O'REILLY (2002)



**Sangho Lee** received the B.S. in computer engineering from Hongik University, Korea, in 2006 and received the M.S. degree in computer science and engineering from Pohang University of Science and Technology (POSTECH), Korea, in 2008. He is currently working toward the Ph.D at POSTECH. His current research interests include information security, digital rights management, secure group communication, privacy protection, and applied cryptography.

**Jong Kim** received the B.S. in electronic engineering from Hanyang University, Korea, in 1981, the M.S. degree in computer science from the Korean Advanced Institute of Science and Technology, Korea, in 1983, and the Ph.D. degree in computer engineering from Pennsylvania State University in 1991. He is currently a professor in the Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Korea. From 1991 to 1992, he was a research fellow in the Real-Time Computing Laboratory of the Department of Electrical Engineering and Computer Science, University of Michigan. His major areas of interest are fault-tolerant computing, parallel and distributed computing, and security.

**Sung Je Hong** received the B.S. in electronics engineering from Seoul National University, Korea, in 1973, the M.S. degree in computer science from Iowa State University, Ames, in 1979, and the Ph.D. degree in computer science from the University of Illinois, Urbana-Champaign, Illinois, in 1983. Since 1989, he has been a professor in the Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Korea. From 1983 to 1989, he was a staff member of Corporate Research and Development, General Electric Company, Schenectady, NY, USA. His current research interests include VLSI design, CAD algorithms, testing, parallel processing, and RFID security.