Journal of Computer Security 0 (0) 1
IOS Press

Content Sharing Schemes in DRM Systems with Enhanced Performance and Privacy Preservation

Michal Davidson [a], Tamir Tassa [a,*] and Ehud Gudes [a,b]
[a] *Department of Mathematics and Computer Science, The Open University, Ra'anana, Israel.*
[b] *Computer Science Department, Ben-Gurion University of the Negev, Israel.*

Abstract. We present a solution to the problem of content sharing in digital rights management (DRM) systems. Users in DRM systems purchase content from content providers and then wish to distribute it between their own devices or to other users. The goal is to allow the sharing of such content, with the control of the content provider, while ensuring that it complies with the content's usage rules. We also address in this paper the subject of protecting users' privacy during the content sharing; to the best of our knowledge no study thus far addressed this topic. While most of the previous studies on content sharing in DRM systems assume the existence of authorized domains, ours does not make that assumption. The solutions that we present here are based on Certified Sharing Requests which are used when devices request from the content provider to share content with other devices. Our solutions enhance the usability of DRM, from both the users' and content provider's perspective, by supporting on-the-fly sharing, sharing and re-sharing of controlled content, a pay-per-share business model, and privacy preservation.

Keywords: Digital Rights Management, Content Sharing and Re-Sharing, Authorized Domain, Proxy Re-encryption, Certified Sharing Requests

1. Introduction

In recent years there has been an explosion in the usage of smartphones and other devices for downloading and viewing digital media. For the providers of such data it creates a huge commercial opportunity as well as a great risk. The risk stems from the difficulty of controlling the dissemination of the provider-owned content and its sharing between users and devices. Digital Rights Management (DRM) is at the core of systems and methods for controlled sharing.

The usage of DRM in the industry is controversial, since it limits the use of legally purchased content, and it does not allow certain scenarios that were previously possible. One of the main controversies with DRM systems is with regard to content sharing. When physical content is purchased it can be shared, copied, and re-sold. In DRM-free systems, digital content can also be freely shared and copied. On the other hand, in DRM systems, the content provider (CP) wants to control such content sharing, and ideally would like to get paid whenever such content is further shared with other users.

*Corresponding author. E-mail: tamirta@openu.ac.il

The digital media industry finds itself in a situation where forbidding content sharing antagonizes existing customers and pushes them to leave; on the other hand, freely permitting content sharing reduces the value of the content and leads to a decrease in revenue. Introducing the pay-per-share business model brings a win-win situation where customers who otherwise might have not purchased the content can now do so at a reduced cost, while the CP's revenue increases with a minimum overhead. It should be noted that since the content is being shared, the CP merely needs to provide a permission for this sharing transaction without having the communication overhead of actually transferring the data-heavy content.

Most current solutions for content sharing propose and expand upon the use of an "Authorized Domain" — a group of devices that can freely share content amongst them [18]. However, the Authorized Domain model does not support current business needs. Such models have two main drawbacks: they do not support "on the fly" sharing, namely, sharing between devices that do not belong to the same domain; and they do not offer the means to control which content can be shared between two devices. In addition, the problem of formally defining a domain of devices remains open, since formal logic is limited when it comes to human "intuition", and many different use cases exist where users feel entitled to join a domain (e.g users with multiple devices, family members residing in different households, roommates who are not related but living in the same household, etc.). For these reasons, the Authorized Domain solution does not address today's users' needs. In particular, users are demanding that DRM systems support content sharing between friends, without requiring that their devices belong to the same domain. This functionality cannot be supported by the Authorized Domain model, and requires "on-the-fly" sharing such as that which is supported by our model. With the constant threat of users resenting the DRM system limitations, and deciding to avoid becoming customers of DRM systems altogether, acquiring content through illegal methods which do not generate CP revenue, it is crucial that DRM systems become more flexible, while obviously not compromising on security. Our paper proposes a secure yet flexible system for content sharing to replace the Authorized Domain model.

In this paper we propose solutions for content sharing that do not rely upon authorized domains.

A recent scheme that solves the content sharing problem without assuming authorized domains was proposed by Ma et al. [13]. Their scheme uses a proxy re-encryption method [3] which allows re-encryption of a message without decrypting it first. Although this method is elegant and secure (see its detailed description in the next section), it involves a considerable overhead in terms of storage, and it relies on the complex cryptographic primitive of bilinear pairing. Moreover, the implementation of the pairing in [13] dictates using the El-Gamal public key cryptosystem and prevents using other public key methods like the prevalent RSA cryptosystem. Finally, the solution in [13] does not support re-sharing of purchased content with other users, or a flexible payment scheme.

Another problem is preserving the privacy of devices that request shared content. While there are a number of works on privacy preservation during content purchasing in DRM systems, we are not aware of any solutions that address the problem of privacy preservation during content sharing. We propose here a scheme that protects users' privacy during the process of content sharing.

Here we address the above problems and present a simpler scheme for controlled sharing in DRM systems. Our scheme is called: the *CSR (Certified Sharing Request) Scheme*. The specific goals of our work are:

- "On the fly sharing" — sharing between devices that were not previously connected in any fashion.
- Re-sharing to any preset depth — devices that received the content through sharing can also re-share the content with other devices.

- Content-dependent sharing privileges — each content is tagged by its own sharing privileges and limitations. (In contrast to authorized domain solutions where two peer devices in the same domain may share all content between them.)
- CP knowledge — the CP must be notified of and permit sharing before it can take place. We present also a solution that respects a relaxed form of this goal, in which the CP controls only the number of sharing requests per device in a given time window; such a relaxed solution is more efficient.
- Pay-per-sharing — support for a pay-per-sharing business model where a device can be charged both for purchasing and sharing content. This business model allows the CP to charge a reduced rate for sharing content (say ten percent of the charge of directly purchasing the same content). This both provides the CP with additional revenues while benefiting users who enjoy reduced charges.
- Privacy preservation — preserving the anonymity of users who receive shared content.
- Security — achieving the aforementioned objectives while ensuring common security and privacy properties. An important contribution of this paper is a detailed security analysis of both the proxy re-encryption scheme [13] and our scheme, which highlights the security advantages which the latter offers.

To summarize, the CSR scheme provides a simple, efficient and flexible mechanism of DRM sharing, with strong security and privacy guarantees. The CSR scheme aims to replace the authorized domain model, since the latter fails to address current user expectations in the mobile era.

The rest of the paper is organized as follows. Section 2 provides the necessary background, an overview of the related work, and a detailed description of the proxy re-encryption scheme. In Section 3 we describe our sharing scheme. The version which we present there is the fully secure version of our scheme, which is designed for the full non-trust scenario, i.e., a scenario in which the CP performs all necessary verifications before approving any sharing or re-sharing. In Section 4 we present a more efficient scheme which is designed for the so-called partial trust scenario; in that scenario, the devices that purchased content directly from the CP are partially trusted and therefore the CP delegates to them some of the verifications that need to take place before sharing content. In Section 5 we expand our schemes to support privacy preservation. In Section 6 we discuss the advantages and disadvantages of both variants of our scheme and compare them to the proxy-rencryption scheme of [13]. We conclude in Section 7 and outline some future work directions.

Part of this work appeared in [8] as a short paper. The preliminary discussion and related work coverage in this manuscript (Sections 1 and 2) is much more comprehensive than their counterpart in [8]. So are the discussion and analysis of the CSR scheme (Section 3). Sections 4 (The CSR scheme in the partial trust scenario) and 5 (Privacy preservation applied to content sharing) are entirely new. Finally, the discussion of our results and comparison to the existing art (Section 6) is much more elaborated than the succinct discussion in [8].

2. Background and related work

2.1. Overview

Digital Rights Management ("DRM") is a method for controlling the viewing and distribution of digital content. A DRM system consists of a group of different entities:

- Content (C) - a purchasable item of digital content, typically a video, audio or text file. The content is distributed in an encrypted format, using a symmetric encryption, and can only be decrypted using the corresponding content key.

- Content License (CL)- a record that includes the content key and a set of usage rules. Content licenses are typically encrypted using public key encryption.
- Content Provider (CP) - The entity that owns the content items and wishes to control the distribution of the content to its client devices. It is sometimes known as the “Rights Issuer".
- Trusted Computing Base (TCB) - A trusted hardware component within the device which securely stores and processes keys.
- Device (will be denoted by A, B, A_0, A_1 etc.) - a tamper proof computer processing unit, such as a set top box or a smartphone, that is capable of parsing and decrypting the encrypted content and the encrypted content licenses. Each device holds a secret key and a corresponding certificate, which is signed by a Certificate Authority. We assume that the device’s secret key, as well as the content keys which the device extracts from content licenses, are securely stored and processed in a TCB and cannot be accessed by a third party.

It is important to note that DRM systems protect against distribution of cryptographic keys. However, an attack could take the form of an illegal distribution of the clear content, either in its digital format or in its analog format. An illegal distribution of the clear digital content is known as “the streaming problem" (see e.g. [14,26]). Various deployments of DRM systems offer different protections of the clear digital content by hardware and software techniques; the level of protection which is applied by the DRM system to the clear digital content depends on the specific DRM system implementation and on the TCB which is used within this system. See [14,26] for further discussion on that issue and suggested solutions for protecting the clear digital content within DRM systems from being illegally distributed.

The decryption and rendering of content in DRM systems is performed within the TCB so that the content keys are not directly available to the device. Once the encrypted digital content is decrypted and rendered by the device into analog format, the device could in theory ignore the content license and distribute the rendered analog content. This is known as the "analog hole" [24]. However, such a breach of the content license policy is considered a lesser threat for the industry because the conversion from analog content back to digital content, in order to distribute it, typically results in a loss of quality. There are various attempts to solve the "analog hole" vulnerability in DRM systems; the interested reader is referred to [24].

2.2. Related work

Most literature on the topic of content sharing within DRM systems focuses on the use of the Authorized Domain model [18]. This is the classic DRM solution for content sharing, in which a group of authorized devices are defined as belonging to a joint domain, and devices within the same domain can freely share content between them. A single device in the domain is used as the Domain Controller, giving permission to other devices to join the domain.

Sheppard and Safavi-Naini [23] propose enabling content sharing within DRM systems using the Authorized Domain model. They extend the Authorized Domain model with a context-aware method for defining domains; proposing a Domain Expression Language, where context-aware information is processed by the Domain Controller, and the act of joining the domain by a device is considered an environment role, in similarity to Role Based Access Control (RBAC) systems [21].

In [1] and [2] Abbadi proposes schemes for improving the security of the Authorized Domain model. In [1] he proposes using Location Based Services in order to verify that devices are indeed within a reasonable geographic proximity to the user’s registered address. In [2] he suggests using a central Master Control device which only adds a device to the authorized domain after the device is authenticated using

either biometric identification or a password. Both of those studies focus on improving the security of the Authorized Domain model rather than on the users' experience or the business model.

Sadeghi et al. [19] provide a secure platform on open systems which allows the usage of dynamic licenses. They consider the security issue where dynamic licenses are vulnerable to tampering on open systems. They solve this vulnerability by creating a secure virtual layer which runs below the existing open operating system and prevents both altering the licenses and using out-of-date licences. They present dynamic licenses as a solution to content sharing, and suggest a model which allows content transfer and lending. When a user transfers or shares his content, a new license is created for the receiving device and the current device's license is disabled or temporarily invalidated.

Lee, Kim and Hong [12] propose a system for content sharing which does not rely on the Authorized Domain model. Their system relies on time-based rights, where the purchasing device is given for any purchased content a certain amount of time for using it, and that time can be split between several devices.

A recent work on DRM and content sharing [13] uses the method of proxy re-encryption [3]. This method is described in detail in the next section and compared in detail to our scheme in Section 6.

Related work on privacy preservation in DRM systems will be discussed in Section 5.1.

2.3. Content sharing with proxy re-encryption

Ateniese et al. [3] presented a general purpose method for proxy re-encryption. This method allows users who received a message that was encrypted with their public key to re-encrypt it for other users without decrypting it first. In a nutshell, they describe two types of probabilistic public key encryption functions, which they call *first* and *second level encryptions*. If (sk_A, pk_A) denotes the private and public key pair of user A, then $E_\ell(m, pk_A)$, $\ell = 1, 2$, denote the first and second level encryptions of the plaintext m for user A; as the encryption function is probabilistic, $E_\ell(m, pk_A)$ is a large set of possible ciphertexts. User A may decrypt any ciphertext in $E_\ell(m, pk_A)$ using his private key sk_A. In addition, he may re-encrypt $E_2(m, pk_A)$ into a message in $E_1(m, pk_B)$ (namely, a first level encryption of m for user B) without decrypting it first. Their method uses bilinear pairings [5] that are based on the Tate pairing [9].

Ma et al. [13] describe a solution for content sharing in DRM systems which is based on the proxy re-encryption method of [3]. We proceed to describe it briefly.

Purchasing content. When device A requests from the CP to purchase content C, the CP sends to A a message $x \in E_1(m, pk_A)$, where $m = \mathrm{CL}$ is the content license of the requested content. In addition, the CP generates a random key pair (sk_R, pk_R) and sends to A a message $y \in E_2(m, pk_R)$. Finally, it adds to its records a new record that holds the identifiers of A and C, the generated random key pair, and a counter of the number of times in which A shared C so far. After the purchasing protocol is completed, A uses the message x to recover the content license CL, with which it can decrypt the encrypted content.

Sharing content. When device A wishes to share the purchased content C with another device B, it sends a corresponding request to the CP. The CP checks the details of the two devices A and B, and the number of times in which A had already shared that particular content. If that sharing request is approved, the CP computes a re-encryption key, rk, using B's public key pk_B and the random private key sk_R that was generated when A purchased that content, and sends it to A. A uses rk together with the message y which it received upon purchasing that content in order to compute a ciphertext $z \in E_2(m, pk_B)$, by means of bilinear pairing. Device A then sends z together with the encrypted content to device B. B proceeds to recover the content license m and decrypt the content.

There are several disadvantages to this solution: (a) Payment for sharing content can only be performed by the device A who is sharing the content; a better business model would be for the device B to pay to

the CP for the shared content. (b) This model requires that the CP stores a record for each device and each purchased content, where each record stores the corresponding counter and a pair of cryptographic keys. (c) The method is limited to only one level of sharing; it does not allow device B to re-share the content with another device. (d) The method relies upon the complex and costly bilinear pairing function. (e) The usage of bilinear pairings in [13] is based on El-Gamal public key cryptography and, thus, prevents using other public key cryptosystems, such as RSA. As we shall see, our proposed scheme overcomes those disadvantages.

3. Certified Sharing Requests (CSR) and the CSR scheme

Here we present our solution for content purchasing, sharing and re-sharing. We describe how a given device A_0 can purchase content C from the CP; how A_0 may share C with another device A_1; how A_1 can re-share C with A_2; and, in general, how to perform re-sharing of any depth (so that A_2 can re-share C with A_3, A_3 can further share it with A_4, and so forth).

The solution proposed by Ma et al. [13] described the operations of purchasing content and sharing it. Namely, it allows A_0 to purchase C from the CP and then share it with A_1, but it does not apply to cases where A_1 wishes to re-share C with a new device A_2. That is one prominent advantage that our solution offers with respect to the solution in [13]. Another difference between the two solutions is as follows. While in the former solution A_0 re-encrypts the content licence for A_1, in our solution it is the CP that encrypts the content license for A_1. We chose to transfer the task of encrypting the content license from the purchasing device A_0 to the CP for the following reasons:

- The CP must be involved in any such sharing or re-sharing operation since it needs to verify that the sharing or re-sharing is consistent with the usage rules for the content C. Hence, the CP can also encrypt the content license for the new device, and consequently, there is no need to resort to re-encryption solutions which rely on complex primitives such as bilinear pairings.
- The process of re-encryption can be performed only once per content and device, and thus does not support re-sharing. Since in the solution proposed in [13], A_1 receives only a first level encryption of the content license, it cannot perform re-encryption. In our scheme, since we transfer the task of encrypting the content license to the CP, who can perform a direct encryption rather than re-encryption, we can easily perform re-sharing of any depth.

Our solution is based on Certified Sharing Requests (CSRs). The CSRs include: information on the content C, the certificates of the devices that are involved in the sharing and re-sharing operations, and payment information. The CSRs are signed by all involved devices. The mechanism of CSRs is flexible enough to support interoperability between DRM systems; namely, a device in one DRM system can share content with a device that belongs to a different DRM system, under the above assumptions. This will allow CPs to charge devices for "pay per sharing", regardless of the DRM system to which they belong. For simplicity, no differentiation will henceforth be made between sharing in the same DRM system and sharing between devices in different DRM systems.

Note that sharing and re-sharing will require that the involved devices be online in order to sign the sharing requests. We assume that devices will be online within a reasonable time frame, and in cases where devices are offline, a delay can be tolerated. However, we do not anticipate large delays because of offline devices, due to the proliferation of 3G and 4G networks, where devices remain permanently online.

In this section we focus on the non-trust scenario. Namely, we do not grant the devices which are direct clients of the CP with any trust regarding the right to authorize a sharing request. The CP does not rely on A_0 for verifying that the sharing or re-sharing operations are consistent with the usage rules for the content in question. Instead, it performs all necessary checks before sending out the content license encrypted for the new device. The disadvantage in such a non-trust scenario is that the CP has to hold history information per content per device, what may result in too large storage demands, and it has to perform certificate and signature authentication against all devices in the sharing chain. In Section 4 we present a relaxed scenario of partial trust, in which the CP has a partial trust in its direct client devices. In such a model, the storage and computational costs for the CP are reduced significantly.

3.1. Purchasing content

Here we describe the process that takes place when a device, A_0, wishes to purchase a certain content, C. At the completion of this process, A_0 receives from the CP three items: (a) the content C, encrypted by a symmetric encryption using the content key k_C; (b) the content license (which includes k_C), encrypted by A_0's public key; and (c) a corresponding sharing license, denoted SL, which will be used only when A_0 chooses to share C with other devices. Note that in the entire paper the content license and the content itself can be decrypted only by the trusted hardware device (TCB), so no key in the clear (i.e., non-encrypted key) can be sent out by the device. Regarding the protection of the clear content on the device from being illegally distributed, it is reliant on the security of the TCB (see the related discussion on the streaming problem in Section 2).

The purchasing protocol is as follows:

1. When device A_0 wishes to purchase a content C, it sends to the CP a signed message with A_0's certificate and the ID of the requested content C.
2. The CP verifies the signature of A_0, the authenticity of A_0's certificate, and that it is not revoked.
3. The CP encrypts the content license $m = \text{CL}$ of C with A_0's public key.
4. The CP creates a corresponding sharing license SL (which we describe below) and certifies it by signing it. The signed sharing license will be denoted by $[SL, Sig_{CP}]$.
5. The CP sends to A_0 the encrypted content, the encrypted content license (see Step 3 above), and the signed sharing license (Step 4).
6. The CP creates and stores a record of the form (A_0, C, ST_{C,A_0}), where ST_{C,A_0} is a counter of the number of times in which A_0 shared the content C with other devices; it is initialized to zero.
7. A_0 decrypts the encrypted content license using its private key, in order to recover the content key. It then proceeds to decrypt the content using the content key.
8. A_0 also creates a counter ST_C for the number of times that it shared the content C with other devices.

The sharing license SL which the purchasing device A_0 receives upon purchasing the content C will be used when A_0 wishes to share that content with another device A_1. The structure of SL is as follows:

- C (the ID of the purchased content)
- GDI (the Global Device ID of A_0)
- MSL (Maximum Sharing Level)
- NoS (Number of Sharings).

MSL denotes the depth of sharing which is permitted for this particular content. The value of 0 indicates that no sharing is permitted for this content. The value of 1 permits only devices that purchased the

content directly from the CP to share the content with other devices, while the latter devices cannot do re-sharing. The value of $\ell+1$ enhances the permissions associated with the value ℓ by allowing one more level of re-sharing. The field *NoS*, on the other hand, bounds the total number of devices that can receive the shared content from A_0 by means of sharing or re-sharing. Viewing the sharing process as a tree over devices, where a parent-child relationship in the tree corresponds to an act of sharing, then *MSL* bounds the height of the sharing tree, while *NoS* bounds the number of edges in that tree. (Other constraints such as the maximum degree of a node can also be added.)

Note that while we define the above fields for the sharing license, the scheme can be easily extended to include additional restrictions. For instance, attributes that may appear in the device certificate, such as location, can be checked according to criteria which can be added to the sharing license.

In order to support a pay-per-share business model, either device A_0 or device A_1 may be billed by the CP for sharing content. The CSR indicates which device will be charged for the content sharing using the *COD* (Charge Original Device) field.

A certificate of a device is revoked when a device is lost or stolen, or when the CP identifies the device as being hacked. The list of all revoked devices is called a Certificate Revocation List (CRL) [15]. The CP verifies that a device's certificate is not revoked by checking that it does not appear in the CRL. Using CRLs to identify revoked certificates is a standard practice in DRM systems. However, some DRM systems utilize a so-called Online Certificate Status Protocol (OCSP) [22] instead of CRLs.

3.2. Sharing content

When device A_0 wishes to share a specific content C with device A_1, the following process will take place, see Figure 1.

1. The device A_1 sends a request to A_0 to get from it content C by means of sharing.
2. Device A_0 sends to A_1 the message $[SL, Sig_{CP}, COD]$ where the first field in that message is the certified sharing license that A_0 received from the CP when it purchased C.
3. Device A_1 adds to the received message its certificate $Cert_{A_1}$ and a field of Encrypted Payment Information EPI. EPI includes the information needed to charge A_1 for performing this sharing, if required.
4. A_1 sends to A_0 the message $[SL, Sig_{CP}, COD, Cert_{A_1}, EPI, Sig_{A_1}]$, where the last field is A_1's signature on the preceding fields in the message.
5. A_0 checks that the value of COD was not altered and then he verifies that the internal counter ST_C which it maintains for the content C is smaller than NoS (the value that appears in SL). If it is, then A_0 increments ST_C and signs the sharing request. The result is called a CSR (Certified Sharing Request):

$$CSR_1 := [SL, Sig_{CP}, COD, Cert_{A_1}, EPI, Sig_{A_1}, Sig_{A_0}].$$

   Here, Sig_{A_0} is the signature of A_0 on all preceding fields in CSR_1.
6. A_0 sends CSR_1 to the CP.
7. The CP performs the following verifications:
   (a) It verifies all three signatures that appear in CSR_1.
   (b) It checks that the devices A_0 and A_1 are not revoked.

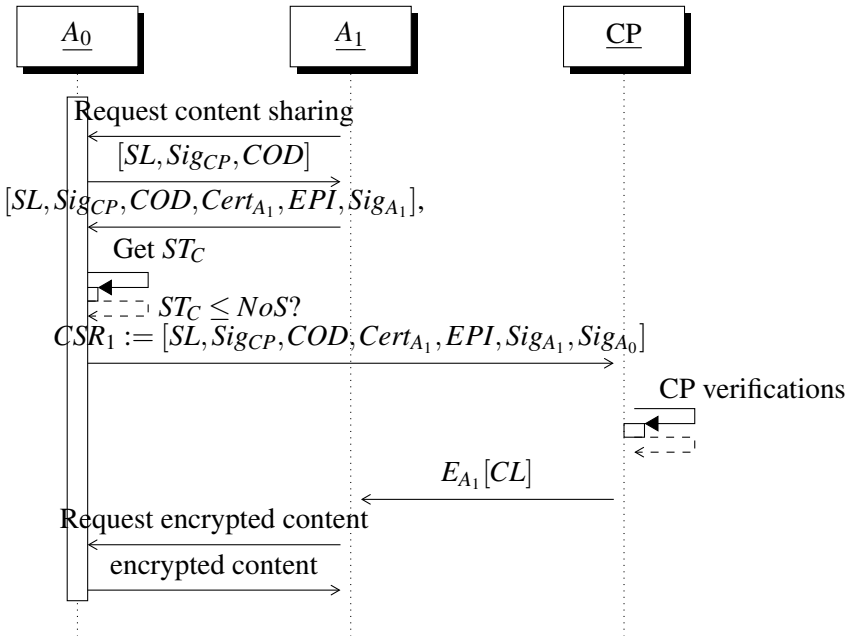

Fig. 1. Content Sharing in the non-trust scenario. The CP verifications in the scheme are: (a) all signatures are valid? (b) the certificates of A_0 and A_1 are not in CRL? (c) $MSL \geq 1$? (d) $ST_{C,A_0} \leq NoS$?

(c) It checks that the *MSL* field, as appears in *SL*, is at least 1 (in order to allow this sharing request which is of depth 1). Note that the sharing license is used mainly for checking the sharing depth and does not contain any keys.

(d) It retrieves the record (A_0, C, ST_{C,A_0}) which it stored for the device A_0 and content C, and checks that ST_{C,A_0} (the number of times that A_0 had already shared the content C) is smaller than NoS (as appears in *SL*). If it is, the CP increments the value of ST_{C,A_0}.

8. If all verifications were successful, the CP encrypts the content license with A_1's public key and sends it to A_1 (either via A_0 or directly).
9. A_0 sends to A_1 the encrypted content (which A_0 already possesses, since it received it from the CP upon purchasing that content).
10. A_1 decrypts the content license using its private key, in order to recover the content key. It then proceeds to decrypt the content using the content key.

Note that the device A_0 has to verify the internal counter ST_C in Step 4 in the protocol above in order to refrain from unnecessary communications vis-a-vis the CP. The CP repeats the same check (Step 6d) since it does not trust A_0, but, as explained above, the check by A_0 in Step 4 is still necessary for communication overhead considerations. We note that if the check in the device is performed in the TCB, it can prevent Denial of Service attacks.

3.3. Re-sharing content

We have described in Section 3.2 the basic sharing scenario. Here we discuss how re-sharing is done; namely, how a device that got a content by sharing or re-sharing can re-share it with another device. Our description of the process will be done by means of induction. Assume that a specific content C was

already shared by the following chain of devices, $A_0, A_1, \ldots, A_{i-1}$; i.e., A_0 purchased that content from the CP, and then it shared it with A_1, who continued to share it with A_2, and so forth. Below we describe how A_{i-1} can re-share the content with a new device A_i.

The re-sharing protocol proceeds as follows:

1. The device A_i sends a request to A_{i-1} to get from it content C by means of sharing.
2. Device A_{i-1}, that already possesses $[SL, Sig_{CP}, COD, Cert_{A_1}, \ldots, Cert_{A_{i-1}}]$ from the protocol that took place when it obtained the shared content, sends this sharing request to A_i.
3. A_i adds to the sharing request its certificate $Cert_{A_i}$ and the payment information field EPI.
4. A_i sends to A_{i-1} the sharing request

$$[SL, Sig_{CP}, COD, Cert_{A_1}, \ldots, Cert_{A_{i-1}}, Cert_{A_i}, EPI, Sig_{A_i}],$$

   where the last field is A_i's signature on the preceding fields in the message.
5. The message is sent up the chain of devices, where device A_j adds its own signature on the message that it received from A_{j+1}, and then sends it to A_{j-1}, $j = i-1, \ldots, 1$.
6. A_0 verifies that ST_C is smaller than NoS. If it is, then A_0 increments ST_C and signs the sharing request. The resulting CSR is:

$$CSR_i := [SL, Sig_{CP}, COD, Cert_{A_1}, \ldots, Cert_{A_i}, EPI, Sig_{A_i}, \ldots, Sig_{A_0}].$$

7. A_0 sends CSR_i to the CP.
8. The CP performs the following verifications:
   (a) It verifies all $i+2$ signatures that appear in CSR_i.
   (b) It checks that all devices $A_0, \ldots, A_i$ are not revoked.
   (c) It checks that the MSL field, as appears in SL, is at least i.
   (d) It retrieves the record (A_0, C, ST_{C,A_0}) and checks that $ST_{C,A_0} < NoS$. If so, it increments the value of ST_{C,A_0}.
9. If all verifications were successful, the CP encrypts the content license with A_i's public key and sends it to A_i (either directly or via $A_0, ..., Ai-1$).
10. A_{i-1} sends to A_i the encrypted content (which A_{i-1} already possesses, by definition of content re-sharing).
11. A_i decrypts the content license using its private key in order to recover the content key. It then proceeds to decrypt the content using the content key.

4. The CSR scheme in the partial trust scenario

Here we further improve the CSR model by removing from the CP the overhead of authorizing each content sharing transaction. We discuss a scenario in which the CP has a partial trust in the device A_0 who originally purchased content from the CP, and allows A_0 to authorize sharing requests. Since the CP's trust in A_0 is not full, the CP still performs some monitoring of the number of sharing requests made by A_0. An inordinate number of sharings requests by A_0 within a period of time serves as a warning sign. Consequently, the CP limits the number of sharings that A_0 can do, in total (for any content), within each time period, and might decide to further investigate the sharing behavior of A_0.

The computational and storage overhead of the CP are reduced significantly in the partial trust scenario. First, the CP does no longer need to hold a counter per content per device; it only has to hold a counter per device that limits the number of sharing requests that the device may submit within a set time period. Second, the act of verifying the chain of signatures in the CSR and checking the certificates are valid is transferred from the CP to the sharing device A_0. For checking certificate verification the device can either obtain the CRL from the CP, or use an interactive protocol such as OCSP [22]. Below we describe the modifications to all three protocols.

4.1. Purchasing content

Purchasing content is performed as described in Section 3.1. The sharing license which A_0 receives together with the content C that he purchased, together with the counter ST_C created internally, will be used when A_0 wishes to share the content C with some other device A_1.

4.2. Sharing content

The process of content sharing in the partial trust scenario is as described below and in Figure 2. Here, each device wishing to share content must receive from the CP a signed CRL. This CRL will be used by the device to verify the certificates of devices wishing to receive shared content, and it will be updated periodically.

1. The device A_1 sends a request to A_0 to get from it content C by means of sharing. In the request it includes its certificate $Cert_{A_1}$.
2. Device A_0 verifies the certificate of A_1 against the CRL, to see that it is not revoked.
3. A_0 sends to A_1 the sharing request $[SL, Sig_{CP}, COD]$ where, as before, the field COD indicates who will pay for this sharing.
4. A_1 adds to the sharing request its certificate $Cert_{A_1}$, the EPI field (as explained in Section 3.2) and his signature, and sends back to A_0 the signed sharing request,

$$[SL, Sig_{CP}, COD, Cert_{A_1}, EPI, Sig_{A_1}].$$

5. A_0 checks that $ST_C \leq NoS$, $MSL \geq 1$, and verifies the signature of A_1.
6. If all checks were successful, A_0 sends to the CP the following signed sharing request:

$$CSR_1 := [SL, Sig_{CP}, COD, Cert_{A_1}, EPI, Sig_{A_0}].$$

7. The CP verifies the certificate of A_0 against the CRL, verifies the signatures of A_0 and itself, and verifies that the number of sharing requests from A_0 does not exceed the allowed periodic threshold.
8. If both checks passed successfully, the CP charges either A_0 or A_1, as indicated by the field COD, encrypts the content license $m = \text{CL}$ with A_1's public key, and sends it to A_1 (either via A_0 or directly).
9. A_0 sends to A_1 the encrypted content (which A_0 already possesses).
10. A_1 decrypts the content license using its private key. A_1 then extracts the content key and then proceeds to decrypt the content using that key.

Note that replay attacks can be prevented using techniques which are common in protecting other protocols against such attacks, e.g. challenge-response or short life span (see more details in [25]).

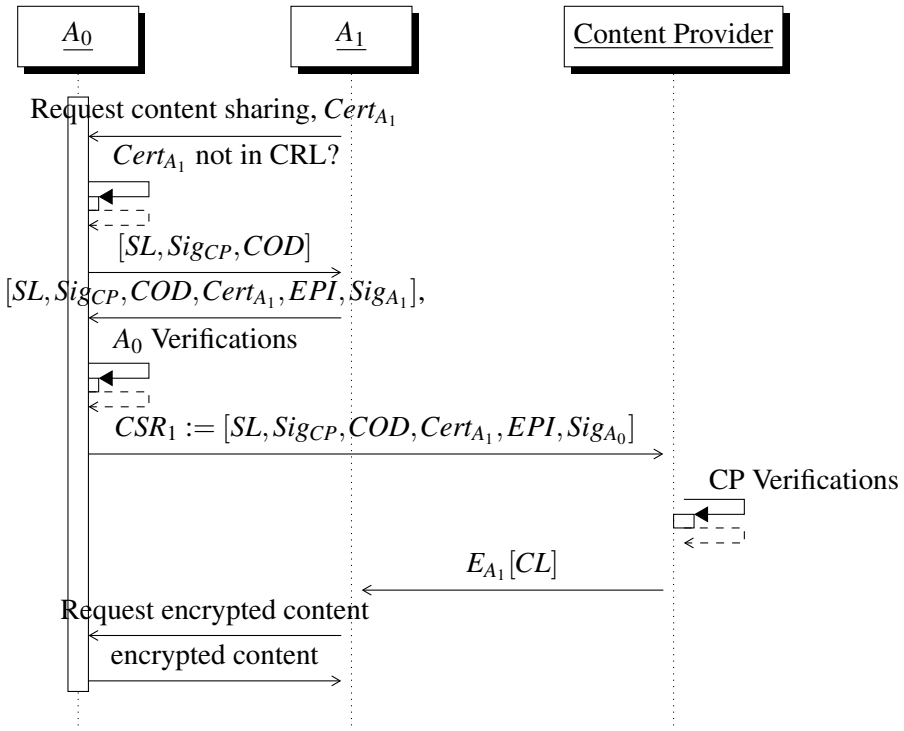

Fig. 2. Content Sharing in the CSR partial trust scenario. The A_0 verifications are: (a) $ST_C \leq NoS$? (b) $MSL \geq 1$? (c) the signature of A_1 is valid? The CP verifications are: (a) $Cert_{A_0}$ not in CRL? (b) the signature of A_0 is valid? (c) ST_{A_0} is within the allowed limit?

4.3. Re-sharing content

Re-sharing is performed as described in Section 3.3, with the following changes to the process that we described in Section 3.3.

- Step 1 is modified as follows: A_i includes its certificate $Cert_{A_i}$ when it requests content C from A_{i-1}
- Step 6 is modified as follows: Device A_0 checks that $ST_C \leq NoS$, verifies the certificates of $A_1, \ldots, A_i$ against the CRL, and verifies the signatures of $A_1, \ldots, A_i$. If all checks passed successfully, A_0 creates a certified sharing request of the form $CSR_i := [SL, Sig_{CP}, COD, Cert_{A_i}, EPI, Sig_{A_0}]$. Note that the message CSR_i that A_0 sends in this case to the CP is reduced in comparison to CSR_i in Section 3.3: here, it includes just the certificate of the receiving device A_i, and is signed only by A_0.
- Step 8 is modified as follows: The CP verifies only the signature of A_0 and that the number of sharing requests from A_0 does not exceed the allowed periodical threshold.

5. Privacy preservation applied to content sharing

A topic not often addressed within the field of content sharing is that of privacy preservation. Particularly when sharing content between different users, rather than between two devices belonging to the same user, the user who receives the content may prefer that his identity, and the shared content he wishes to receive, not be disclosed to the CP. Therefore, by “privacy" we mean hereinafter the prevention of the

CP from finding out the identity of the device that receives the shared content. The receiving device does not need to remain anonymous vis-a-vis the device that provides the shared content, since the business model of content sharing assumes that sharing takes place between friends and peers. As in this study we discuss sharing and re-sharing of content in DRM systems, we shall focus here on privacy-preservation in sharing and re-sharing, rather than on privacy-preservation in purchasing content.

When discussing privacy in DRM systems there are two privacy-related problems: protecting the identity of the original buyer, and protecting the identity of the device wishing to receive shared content. The problem of protecting the identity of the original buyer has been addressed in other works, which we review in Section 5.1, see [11,10,17,16,20]. In this study we focus on the second problem; namely, we assume that the identity of the original buyer is known to the CP, or that it is protected by one of the methods proposed in the above mentioned studies, and we discuss means to preserve privacy when executing content sharing. We are not aware of any other paper that considers this problem, aside for [7] which is limited to the Authorized Domain model.

In Section 5.2 we describe in greater detail a work on privacy-preservation in DRM systems that is applicable to our CSR scheme. In Section 5.3 we describe privacy-preserving enhancements of our CSR schemes.

5.1. Related work on privacy-preservation in DRM systems

Many schemes for privacy-preservation in DRM systems rely on the use of a trusted third party. One such example is the work of Kleiner et al. [11] which describes a method for privacy-preserving DRM that is based on the use of web services. They propose a specific implementation, which uses the Web Services Security (WSS) protocol. Jiang and Yang [10] apply an oblivious transfer protocol for preserving privacy in DRM systems. They consider a multi-party DRM system that consists of a "license center" and a "register server", in addition to the CP. The register server acts as a trusted third party.

Petrlic and Sekula [17] propose an application of proxy re-encryption for DRM systems, where proxy re-encryption is used to support privacy preservation in a multi-party DRM scheme. The use of proxy re-encryption removes the need for a trusted third party in the DRM system. Their solution provides anonymous authentication by storing in all TCBs an identical set of public and private keys and an identical certificate. They consider a multi-party DRM system, rather than the standard two-party DRM system which we address in this paper. In two-party systems all protocols involve only the CP and a single device; in multi-party systems there are devices, Content Providers (CPs) who sell the content to users, as well as Content Distributors (CDs), who physically supply the content. We note that using a common certificate for all TCBs raises obvious security concerns, one of which is the difficulty in revoking a specific TCB.

Conrado et al. [7] define an anonymous DRM system reliant on the use of anonymous smart cards. A limited form of content copying can be performed by means of transferring the CL. Their solution allows only for transferring content, and it requires the revocation of the original CL, which means that once content is transferred, the original device can no longer view it. A list of these revoked CLs is attached to the device certificate, and the CP verifies that a CL is revoked before permitting the CL transfer. This solution can lead to significant storage and processing overhead for content viewing, since the list of revoked CLs for the device must be checked before viewing can take place. That study also proposes a method for preserving privacy in an authorized domain, by means of a domain manager. There is no support for anonymous content sharing outside the authorized domain.

Perlman et al. [16] describe two families of schemes for privacy preserving DRMs: one that uses anonymous cash and another that uses blind decryption [20]. They describe the advantages and disadvantages of each approach.

Win, Thomas and Emmanuel [27] combine anonymous tokens (similar to Perlman et al.'s [16] anonymous cash scheme) with blind decryption [20]. In the next section we describe their model.

5.2. Anonymous Token Sets for Privacy Protection in DRM systems

Win, Thomas and Emmanuel [27] propose a method of privacy preservation for DRM systems, which is based on the use of anonymous tokens. These tokens are created by the CP and they are distributed anonymously, after encryption, to devices who wish to purchase content at a later time. These encrypted tokens may be viewed as "blank checks" which cannot be used until they are decrypted by the CP (an operation which has a similar role to a check being signed in order to allow it to be used). The tokens can be used by the device (after their decryption) to receive from the CP content licenses anonymously, as we proceed to describe in detail below.

We note that a distinction is made in [27] between the CP and another party that manages the tokens which is called there the "System Owner". As noted there, the CP and the System Owner can collaborate, or be the same entity, without causing a breach in privacy. For the sake of simplicity, we shall assume here that the CP and the System Owner are the same entity.

We proceed to outline the protocol suggested in [27].

5.2.1. Preparation of anonymous token sets by the CP

The CP generates anonymous token sets $ATS_1, \ldots, ATS_k$, where each token set ATS_j, $1 \leq j \leq k$, will be given to a different device. (When needed, new token sets can always be generated.) Each token set ATS_j, $1 \leq j \leq k$, consists of ℓ anonymous tokens $\{T_{j,i} : 1 \leq i \leq \ell\}$, where each token takes the form

$$T_{j,i} = [TID_{j,i}, Time_{exp}, Sig_{CP}] .$$

Here, $TID_{j,i}$ is the identity of the token $T_{j,i}$, $Time_{exp}$ is its expiration date, and Sig_{CP} is the signature of the CP on the preceding two fields.

The CP generates those tokens in the following manner: Let s be a secret known only to the CP, and s_j be a value which is unique for the token set ATS_j. Then the CP defines $TID_{j,\ell+1} := s||s_j$ and proceeds to compute $TID_{j,i} = H(TID_{j,i+1})$ for all $i = \ell, \ldots, 1$, where H is a secure hash function.

The CP then encrypts the token sets as follows:

1. It generates a symmetric key k_j for encrypting the token set ATS_j, $1 \leq j \leq k$: $E_{k_j}(ATS_j) = (E_{k_j}(T_{j,1}), \ldots, E_{k_j}(T_{j,\ell}))$.
2. The CP has an RSA key pair — (e,n) is the public key for encryption, and (d,n) is the corresponding private key, known only to the CP. It encrypts k_j as follows: $E_{CP}(k_j) = k_j^e \bmod n$.
3. An anonymous token set package consists of $(E_{CP}(k_j), E_{k_j}(ATS_j))$. As long as the first component $E_{CP}(k_j)$ remains encrypted, a device cannot make use of the tokens in the package.

5.2.2. Purchasing content using the anonymous token sets

A device A that wishes to purchase content, needs first to acquire an anonymous token set package. That transaction takes place anonymously, assuming the device uses an anonymous payment scheme [4,6]. Note that at this stage it is not required to authenticate the device. After acquiring the token set ATS_j, the device will be identified by the CP only by the index j.

Since the tokens remain encrypted with k_j, they cannot yet be used by the device. In order to use the anonymous token set package, device A requests from the CP the decryption of $E_{CP}(k_j)$. This is done using blind decryption [20], in the following manner.

1. A calculates a random secret blinding factor $r \in \mathbf{Z}_n^*$, and computes $x = r^e \bmod n$.
2. A sends to the CP the value y= $x \cdot E_{CP}(k_j) \bmod n$, together with its certificate $Cert_A$.
3. The CP verifies the certificate. If valid, it computes $z = y^d \bmod n$ and sends it to A. Note that as $y = (rk_j)^e \bmod n$ then $z = rk_j$.
4. Finally, A computes $k_j = z \cdot r^{-1} \bmod n$.

Note that this process is done separately from the procedure for acquiring the anonymous token set package, so that the two transactions cannot be linked. Due to the blind decryption, the CP does not learn the value of k_j and thus the token sets remain anonymous, despite the fact that the CP authenticates the device.

The now decrypted token sets can be used by A in order to purchase up to ℓ content items anonymously:

1. A generates a random symmetric encryption key $k_{C,j}$ for each content request for a specific content C.
2. A sends anonymously to the CP the content request $(k_{C,j}, T_{j,i}, C)$, encrypted using the CP's public key, where $T_{j,i}$ is one of the ℓ tokens in the token set ATS_j that was not used so far by A.
3. After decrypting the request, the CP verifies its own signature as embedded in the token $T_{j,i}$, checks that the expiration date of the token has not yet expired, and then encrypts the content license CL with $k_{C,j}$ and sends it to A, together with the encrypted content.
4. Finally, A decrypts the content license and then decrypts the content itself.

The above described solution of [27] does not rely on a trusted third party, and it can be used in both two-party and multi-party DRM systems.

Anonymous token sets can be revoked, without breaking the privacy of the device, and there is a secure method for renewing anonymous token sets upon their expiration. If a device requests the renewal of revoked anonymous token sets, the tokens will be identified by the CP as revoked and the request for renewal by the device will be refused. As those features of the solution are less relevant for our subsequent discussion, the interested reader may find the relevant details in [27].

5.3. A privacy preserving enhancement of the CSR scheme

We will show here how the mechanism of anonymous token sets [27] can be used in the CSR scheme, in order to enable privacy protection for content sharing.

We concentrate here on performing privacy preserving sharing in the CSR scheme in the non-trust scenario. Performing re-sharing in a privacy pa preserving manner can be carried out similarly. Also the enhancement of the CSR scheme in the partial trust scenario follows the same lines where, as in Section 4.2, it is A_0 rather than the CP that authenticates the identity of the device wishing to receive shared content and authorizes the sharing request.

As discussed above, we wish to preserve the privacy of the device A_i which is receiving the shared content. The identity of device A_0 can remain known to the CP. We propose here a privacy preserving scheme where instead of using the certificate of A_i for content sharing, A_i generates a random key pair $(sk_{anon_i}, pk_{anon_i})$ (where pk_{anon_i} is the public encryption key and sk_{anon_i} is the private decryption key), and a corresponding anonymous self-signed certificate $Cert_{anon_i}$. Each Content Sharing Request issued

by A_i includes $Cert_{anon_i}$ (with the corresponding public key pk_{anon_i} which the CP will use for encryption), together with the anonymous token which A_i acquired previously from the CP.

When the anonymous token mechanism is applied within the CSR non-trust scheme, the identity of the content C will be known to the CP, but the identity of the requesting device A_i will not be revealed.

5.3.1. System setup

A device A wishing to receive shared content will be required first to acquire and decrypt an anonymous token set from the CP, as described in Section 5.2. Recall that in doing so, the device authenticates itself to the CP using its real certificate. Note that this step is performed once. In doing this step, the device acquires a large number of tokens that will be used at later times. Only when all tokens are used, this step is performed once again. Thus, it is not possible for the CP to link the time of the anonymous tokens' acquisition with the time of use of the tokens.

In order to support sharing requests by A in a private manner, A generates at this stage a random key pair $(sk_{anon_i}, pk_{anon_i})$ and a corresponding anonymous self-signed certificate $Cert_{anon_i}$. (Typical end-point devices can support public key generation. Platforms such as OpenSSL and Android already provide support for RSA key generation[1][2].) The certificate and random key pair will be used in any request issued by A to receive content from another device. (As noted earlier, in order to prevent attacks that attempt to build user profiles, a different certificate can be used per each request. Here, for the sake of simplicity, we assume a constant anonymous certificate.)

5.3.2. Content purchase

The purchase of content is performed by A_0, which does not require privacy. This step remains as described in Section 3.1

5.3.3. Content sharing

1. Device A_0 sends to A_1 the message $[SL, Sig_{CP}, COD]$.
2. Device A_1 adds to the received message its anonymous self-signed certificate $Cert_{anon_i}$ and a token $T_{j,i}$ from the anonymous token set ATS_j which it acquired in the setup stage.
3. A_1 sends to A_0 the message $[SL, Sig_{CP}, COD, Cert_{anon_i}, EPI, T_{j,i}, Sig_{sk_{anon_i}}]$, where the last field is A_1's signature on the preceding fields in the message, using the randomly generated private key. Regarding EPI there are two options: either A_1 paid for the token set upon acquiring it, in which case EPI is not needed here (but then the payment is fixed for all content items); or A_1 pays for the specific content that he requests now and then EPI includes anonymous payment information.
4. A_0 verifies that the internal counter ST_C which it maintains for the content C is smaller than NoS (the value that appears in SL). If it is, then A_0 increments ST_C and signs the sharing request. This yields an ACSR (Anonymous Certified Sharing Request):

$$ACSR_1 := [SL, Sig_{CP}, COD, Cert_{anon_i}, EPI, T_{j,i}, Sig_{sk_{anon_i}}, Sig_{A_0}].$$

   Here, Sig_{A_0} is the signature of A_0 on all preceding fields in $ACSR_1$.
5. A_0 sends $ACSR_1$ to the CP.
6. The CP performs the following verifications:
   (a) It verifies all three signatures that appear in CSR_1.

[1] http://openssl.org/docs/manmaster/crypto/RSA_generate_key.html

[2] http://developer.android.com/reference/java/security/KeyPairGenerator.html

(b) It checks that the device A_0 is not revoked.
(c) It verifies its own signature as embedded in the anonymous token $T_{j,i}$, and checks that the anonymous token is not revoked, its expiration date has not yet expired, and that it was never used before.
(d) It checks that the *MSL* field, as appears in *SL*, is at least 1.
(e) It retrieves the record (A_0, C, ST_{C,A_0}) which it stored for the device A_0 and content C, and checks that $ST_{C,A_0} < NoS$. If it is, the CP increments the value of ST_{C,A_0}.

7. If all verifications were successful, the CP encrypts the content license with the public key pk_{anon_i} which was included in the anonymous certificate $Cert_{anon_i}$, and sends it to A_0. Note that the CP cannot send the content license directly to A_1 since A_1's identity remains private to the CP.
8. A_0 sends to A_1 the encrypted content and the content license, encrypted by A_1's public key pk_{anon_i}, which it received from the CP.
9. A_1 decrypts the content license using sk_{anon_i} in order to recover the content key. It then proceeds to decrypt the content using the content key.

Note that while the CP cannot perform authentication of the device A_1, which remains anonymous, the CP can rely on the anonymous token set, when it is valid, non-expired, and non-revoked, since the CP authenticated the identity of A_1 when the latter decrypted the encrypted anonymous token set package, as described in 5.2.

This privacy preservation enhancement can be similarly applied to the re-sharing protocol in the CSR scheme (Section 3.3) in order to achieve private re-sharing.

6. Discussion

6.1. Advantages and disadvantages of the CSR scheme

The advantages of the CSR scheme are as follows:

1. The CSR scheme supports re-sharing, where the depth and number of re-sharings can be set upfront and controlled.
2. Payment can be made by either the device which originally purchased the content from the CP or from the device which is the recipient in the re-sharing act, what allows a more flexible pay-per-share business model.
3. Compared to the proxy re-encryption scheme, the CP has to store a smaller database that holds only the counter per device per content, without the need to store a pair of cryptographic keys. In the partial trust scenario the amount of data that needs to be stored by the CP is significantly reduced, since the CP has to store just one counter per device (and not per device per content).
4. The CSR scheme does not rely upon the complex and costly bilinear pairing function. As a result, it is also not restricted to use El-Gamal public key cryptography (like the proxy re-encryption scheme [13]).
5. The CSR scheme can support privacy preservation, namely protecting the identity of the device receiving shared content.

It should be noted that the CSR scheme imposes a higher computational overhead on the device's TCB (Trusted Computing Device) in order to perform signatures. However, those signatures increase the secu-

rity of the system and also the proxy re-encryption method could benefit from augmenting the transmitted messages by signatures in order to prevent man-in-the-middle attacks (as we discuss in Section 6.2).

The CSR scheme for content sharing offers advantages to both the CP and users, that the standard DRM content purchase protocol does not offer: users benefit from being able to share content between devices at a reduced cost; furthermore, the CSR scheme supports interoperability between DRM systems, where the device receiving shared content does not need to communicate directly with the CP from which the content was originally purchased. The CP benefits from being able to keep track of flexible peer-to-peer sharing, in a secure and authenticated way, without significant CP overhead, and without using an authorized domain (which, as we said before, does not allow on-the-fly sharing.)

The CSR model can also be applied in the partial trust scenario, which removes considerable overhead from the CP. By placing a limited amount of trust in the device's TCB, the CP can remain involved in the sharing transaction, while performing only minimal verifications, and with significantly reduced storage overhead. The advantages of the CSR scheme remain, including interoperable peer-to-peer sharing and privacy.

6.2. Security

We separate our security analysis into several aspects.

Encryption. The proxy re-encryption solution [13] uses a first level encryption to encrypt the content license and a second level encryption to encrypt the sharing license. Hence, the content key, which is contained in those licenses, is always encrypted in some variant of El-Gamal encryption, as described in [13]. Also in the CSR scheme, the content key is included in the content license which is always encrypted by the public key of the receiving device. Hence, both solutions offer a comparable level of security for the content keys.

Authentication and non-repudiation. The sharing requests in the CSR scheme are always signed, by the tamper-proof TCB, so that they are authenticated and cannot be repudiated. The proxy re-encryption solution does not use signatures. Hence, such a solution is vulnerable to man-in-the-middle attacks, in which an attacker can send to the CP false requests on behalf of some device to share content, and the CP would not be able to distinguish between them and true requests. As a result, devices may be charged for sharing content requests which they did not issue. Moreover, an attacker can intercept messages sent from the CP to a device and replace them with false messages that would not enable the decryption of content licenses.

CP security. In both schemes the CP is required to store sensitive information like its own private key, all content keys, and in the proxy re-encryption solution also the pair of random keys which are generated whenever a device purchases a content. Hence, in the proxy re-encryption scheme the amount of sensitive information is larger than that in the CSR scheme. Therefore, the demands for secure storage in the proxy re-encryption scheme are much higher and, consequently, so is the risk of a security breach in that case.

Another possible danger to the CP is Denial of Service (DOS) attacks. As explained above, in the absence of certifying signatures in the sharing requests in the proxy re-encryption solution, devices can be easily hacked in order to launch DOS attacks on the CP. Such attacks are prevented in our solution by the usage of signatures. Even if a device is hacked to produce false sharing requests, the CP can detect that all sharing requests from that device are illegal and then revoke that device.

In the CSR scheme, each device A_0 which is a direct client of the CP maintains also an internal counter ST_C. That counter is used to prevent sending to the CP too many sharing requests for the same content.

Table 1
Comparison of solutions: purchasing content

	Proxy re-encryption	CSR Non-Trust	CSR Partial Trust	CSR Privacy Preserving
Messages	2	2	2	2
Public Key Encryptions	2 (By CP - first and second level)	1 (by CP)	1 (by CP)	1 (by CP)
Private Key Signatures	0	1 (by CP)	1 (by CP)	1 (by CP)
Private Key Decryptions	1 (by A_0)	1 (by A_0)	1 (by A_0)	1 (by A_0)
Content Decryptions	1 (by A_0)	1 (by A_0)	1 (by A_0)	1 (by A_0)
Key Generation	1 (by CP)	0	0	0
Data Stored in CP	$\sum_{A_0}\sum_C(A_0, C, ST_{C,A_0}, pk, sk)$	$\sum_{A_0}\sum_C(A_0, C, ST_{C,A_0})$	$\sum_{A_0}(A, ST_{A_0})$	$\sum_{A_0}\sum_C(A_0, C, ST_{C,A_0})$

Such a counter provides another layer of security in case other devices are hacked and send to A_0 false sharing requests.

Device Security. Both solutions require from each device to have a trusted computing base (TCB). The TCB may be either software or hardware based, e.g. a TCB, or an embedded secure chip. The private key of the device is stored and used only in the TCB. Note that a TCB is required for all implementations of DRM, since otherwise it would be possible to distribute the content key in the clear once the device decrypts the content license.

The TCB of A_0 is not required by the proxy re-encryption solution for content sharing, since the re-encryption can be performed on the device itself without exposing either the content key or the device's secret key. In the CSR scheme, on the other hand, signing the CSR is a TCB operation. Other operations, such as checking the counters or verifying the signatures of other devices can be performed outside the TCB in the non-trust CSR scheme. In the partial trust case, device A_0 is required to store a CP-signed CRL and authenticate the certificates of other devices $A_1, \ldots, A_i$; that operation should preferably be performed within the TCB. Hence, both schemes depend on a TCB, but in the CSR scheme the TCB is used more because of the added signatures.

The conclusion is that the security of both schemes is comparable, but the CSR scheme is advantageous in terms of storing less sensitive information on the CP, in providing authentication and non-repudiation to sharing requests, and reducing the risk of DOS attacks. Also, in the original proxy re-encryption scheme man-in-the-middle attacks are possible, but they can be resolved quite easily by using signatures to certify messages.

6.3. Resources

We present here a comparison of the resources required by the two solutions. We compare the number of messages, the number of cryptographic operations, and the size of the data required by each solution, for both purchasing and sharing content.

As can be seen in Table 1 , the resources required by the solutions in order to purchase content are similar, with the difference that the CSR solution does not require the CP to perform bilinear key generation, and instead requires the CP to perform a private key signature, which can be argued to be a less heavy operation. As for data storage, the data required to be stored in the proxy re-encryption solution is much larger, especially compared to the partial trust solution. (The summations in the last line of Table 1 are over all devices A_0 and all contents C which each of them purchased.) So both the computation and storage requirements of the CSR solution are smaller.

Regarding the resources required during content sharing, as shown in Table 2, the CSR solution does require two additional messages to be transferred between devices sharing content, however, these are

Table 2
Comparison of solutions: sharing content

	Proxy re-encryption	CSR Non-Trust	CSR Partial Trust	CSR Privacy Preserving
Messages	4	6	6	6
Public Key Encryptions	1 (by A_1)	1 (by CP)	1 (by CP)	1 (by CP)
Private Key Signatures	0	2 (by A_1 and A_0)	2 (by A_1 and A_0)	2 (by A_1 and A_0)
Private Key Decryptions	1 (by A_1)	1 (by A_1)	1 (by A_1)	1 (by A_1)
Public Key Signature Verification	0	3 (by CP)	2 (by CP) and 1 (by A_0)	4 (by CP)
Content Decryptions	1 (by A_1)	1 (by A_1)	1(by A_1)	1 (by A_1)
Key Generation	1 (by CP)	0	0	1 (by A_0, can be done at setup)

not messages which can cause a bottleneck since they are distributed among the devices. In the CSR solution, the CP and the device transferring the shared content are now required to perform private key signing and verification; that replaces the need of the CP to perform re-encryption key generation. In general, most additional resources are required on the part of the devices and not the CP, and we remove the need for the CP to perform bilinear key generation and storing, which can be seen to be a significant advantage. (Note that in the CSR Privacy Preserving scheme, the system setup stage of anonynmous token acquisition runs once and at an earlier and separate stage from content sharing; in addition, it can be performed on a separate server. Hence, we do not include the resources needed for that stage in the resources that are needed for performing content sharing, as shown in Table 2.)

In order to estimate the overhead for the CP which is required to support content sharing using any variant of the CSR scheme, we note that the CP performs at most five cryptographic operations for each sharing request. We measured the run-times of RSA encryption (which is also equivalent to signature verification) by averaging multiple executions of such operations on a hardware comprised of an Intel i7-4600U processor and 16GB memory; our tests show that such encryption takes at most 2 milliseconds. Thus, the overhead for processing a single sharing request in any of our schemes is at most 10 milliseconds. (In fact, as we expect the CP to deploy stronger computing machines, we expect the actual overhead to be significantly smaller.) Assuming 100,000 sharing requests per 24 hours, the resulting overload is 1000 seconds (less than 17 minutes) over one day. Even if the actual number of sharing requests per 24 hours is larger, say 1 million, it poses no real problem for the CP since it can allocate several dedicated machines for that purpose and distribute the processing of the sharing requests to those machines. As far as the resources required from the end-point device for supporting any of our CSR schemes, these are negligible, since each device will process only one sharing request at a time. The overhead of the cryptographic operations (which, as discussed above, is in the milliseconds), is a few orders of magnitude smaller than the time needed for the actual content transfer.

In summary, the additional resources are few, and are mostly on the part of the devices and not the CP. Those added resources support enhanced security and added functionality that was not supported by the proxy re-encryption solution. Finally, the storage requirements for the CP are significantly reduced.

7. Conclusions and future work

We proposed a scheme for content sharing in a DRM system. Content sharing is performed using a Certified Sharing Request (CSR). In contrast to most related work on content sharing in DRM systems, our approach does not rely on the Authorized Domain model. Since the Authorized Domain model does not address current users' expectations, and along with the inherent difficulty of formally defining a

domain does not allow users who are not in the same domain to still share content, DRM systems which only support content sharing under the Authorized Domain assumption are at risk of alienating users. We propose a flexible scheme for content sharing, to replace the Authorized Domain model, and address users' requirements. The CSR model improves upon the limitations of the Authorized Domain model by supporting "on-the-fly" sharing, controlled content sharing and re-sharing, and a pay-per-share business model. The pay-per-share business model that we support will allow the CP to charge a reduced amount per content share, without having the content transfer overhead, leading to a win-win scenario for both the CP and the users. We proposed versions of our CSR scheme both for the fully secure non-trust scenario and for the partial trust scenario that exhibits improved performance. We also dealt with the problem of privacy preservation and showed how the CSR scheme can be enhanced in order to preserve the privacy of the devices that request shared content.

In the future, we would like to enhance the CSR schemes by expanding the privacy preserving solutions so that they protect also the anonymity of the user sharing the content (A_0). Another possible extension of the privacy preserving scheme, in the partial trust scenario, is to provide means for protecting also the identity of the shared content. In addition, we may increase the flexibility of the sharing model to support more complex sharing rules, and refine the payment model to allow more general usage rules.

References

[1] Imad Abbadi. Authorised domain management using location based services. In *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*, pages 280–287, 2007.

[2] Imad Abbadi. Digital rights management using a master control device. In *ASIAN*, pages 126–141, 2007.

[3] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.

[4] Man Ho Au, Sherman S.M. Chow, and Willy Susilo. Short e-cash. In *Progress in Cryptology - INDOCRYPT 2005*, volume 3797 of *Lecture Notes in Computer Science*, pages 332–346. Springer Berlin Heidelberg, 2005.

[5] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, pages 514–532, 2001.

[6] Jan Camenisch, Jean-Marc Piveteau, and Markus Stadler. An efficient fair payment system. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, pages 88–94. ACM, 1996.

[7] Claudine Conrado, Milan Petkovic, and Willem Jonker. Privacy-preserving digital rights management. In *Secure Data Management (SDM)*, pages 83–99, 2004.

[8] Michal Davidson, Ehud Gudes, and Tamir Tassa. Efficient and enhanced solutions for content sharing in DRM systems. In *Data and Applications Security and Privacy XXVIII - 28th Annual IFIP WG 11.3 Working Conference, DBSec 2014, Vienna, Austria, July 14-16, 2014. Proceedings*, pages 373–381, 2014.

[9] Steven D. Galbraith, Keith Harrison, and David Soldera. Implementing the tate pairing. In *ANTS*, pages 324–337, 2002.

[10] Yajun Jiang and Bo Yang. A privacy-preserving digital rights management protocol based on oblivious transfer scheme. *International Journal of Digital Content Technology and its Applications*, 5(5):337–341, May 2011.

[11] Carsten Kleiner, Lukas Grittner, Daniel Kadenbach, Carsten Kleiner, Lukas Grittner, and Daniel Kadenbach. Secure and privacy-preserving drm for mobile devices with web service security âĂŞ an experience report âĂŞ, 2007.

[12] Sangho Lee, Jong Kim, and Sung Je Hong. Redistributing time-based rights between consumer devices for content sharing in drm system. *Int. J. Inf. Sec.*, 8(4):263–273, 2009.

[13] Guojun Ma, Qingqi Pei, Xiaohong Jiang, and Yuchen Wang. A proxy re-encryption based sharing model for drm. *International Journal of Digital Content Technology and its Applications*, 5(11):385, 2011.

[14] Manoranjan Mohanty, Viktor Do, and Christian Gehrmann. Media data protection during execution on mobile platforms Ű a review. Technical report, SICS Swedish ICT AB, July 2014.

[15] Moni Naor and Kobbi Nissim. Certificate revocation and certificate update. *IEEE Journal on Selected Areas in Communications*, 18(4):561–570, 2000.

[16] Radia Perlman, Charlie Kaufman, and Ray Perlner. Privacy-preserving drm. In *Proceedings of the 9th Symposium on Identity and Trust on the Internet*, IDTRUST '10, pages 69–83, New York, NY, USA, 2010. ACM.

[17] Ronald Petrlic and Stephan Sekula. Unlinkable content playbacks in a multiparty drm system. In *DBSec*, pages 289–296, 2013.
[18] Bogdan C. Popescu, Bruno Crispo, Andrew S. Tanenbaum, and Frank Kamperman. A drm security architecture for home networks. In *Digital Rights Management Workshop*, pages 1–10, 2004.
[19] Ahmad-Reza Sadeghi, Marko Wolf, Christian Stüble, N. Asokan, and Jan-Erik Ekberg. Enabling fairer digital rights management with trusted computing. In *ISC*, pages 53–70, 2007.
[20] Kouichi Sakurai and Yoshinori Yamane. Blind decoding, blind undeniable signatures, and their applications to privacy protection. In *Information Hiding*, volume 1174 of *Lecture Notes in Computer Science*, pages 257–264. Springer Berlin Heidelberg, 1996.
[21] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29:38–47, 1996.
[22] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adam. X.509 internet public key infrastructure online certificate status protocol - ocsp, 2013.
[23] Nicholas Paul Sheppard and Reihaneh Safavi-Naini. Sharing digital rights with domain licensing. In *Proceedings of the ACM Workshop on Multimedia Content Protection and Security*, pages 3–12, 2006.
[24] Douglas Sicker, Paul Ohm, and Shannon Gunaji. The analog hole and the price of music: An empirical study. In *Journal on Telecommunications and High Technology Law*, volume 5, 2007.
[25] William Stallings. *Cryptography and network security - principles and practice (3. ed.)*. Prentice Hall, 2003.
[26] Ruoyu Wang, Yan Shoshitaishvili, Christopher Kruegel, and Giovanni Vigna. Steal this movie: Automatically bypassing DRM protection in streaming media services. In *USENIX*, pages 687–702, 2013.
[27] Lei Lei Win, Tony Thomas, and Sabu Emmanuel. A privacy preserving content distribution mechanism for drm without trusted third parties. *2011 IEEE International Conference on Multimedia and Expo*, page 1, January 2011. 80340186.