Unlinkable Content Playbacks in a Multiparty DRM System

Ronald Petrlic, Stephan Sekula

Unlinkable content playbacks in a multiparty DRM system*

Ronald Petrlic (ronald.petrlic@upb.de)
Stephan Sekula (sekula@live.upb.de)

University of Paderborn, Germany

Abstract. We present a solution to the problem of privacy invasion in a multiparty digital rights management scheme. (Roaming) users buy content licenses from a content provider and execute it at any nearby content distributor. Our approach, which does not need any trusted third party—in contrast to most related work on privacy-preserving DRM—is based on a re-encryption scheme that runs on any mobile Android device. Only a minor security-critical part needs to be performed on the device's smartcard which could, for instance, be a SIM card.

1 Introduction

Mobile users access digital content provided in the cloud from anywhere in the world. Music streaming services like *Spotify* enjoy popularity among users. The lack of bulky storage on mobile devices is compensated for by such services by streaming the content to the users's devices. Content is downloaded on demand and can be used only during playback. Thus, paying users are able to access huge amounts of content. There exist certain price models that allow the playback for a certain number of times, until a specific day (e.g., movie rentals), etc.

In such a scenario, we have *content providers* (CPs) that sell licenses to users and there are *content distributors* (CDs) that provide the content. Users can access content from CDs that are closest or provide best service at the moment. This bears advantages for roaming users as they can choose local distributors. Such scenarios are called *multiparty* DRM systems in the literature.

A drawback of today's DRM systems is that CPs/CDs can build content usage profiles of their users as they learn which user plays back content at a certain time, etc. Here we contribute with this paper. We suggest a privacy-preserving multiparty DRM system. In such a system, users anonymously buy content and anonymously playback the content. Moreover, neither CPs nor CDs can link content playbacks to each other and thus cannot build usage profiles under a pseudonym—as the past has shown that profiles under a pseudonym, assumed to be unrelatable to users, can be related to users given external information and thus, inverting user privacy again [1]. One major advantage of our approach compared to related work on privacy-preserving DRM is that we do not need a trusted third party (TTP) that checks licenses.

* This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" (SFB 901). The extended version of this paper can be found at arXiv:1304.8109.

2 Related Work

In [2] a scenario where a content owner provides its content to users via local distributors is presented—similar to our scenario. Users buy licenses for content from a license server (trusted third party). Once a license is bought, the user gets in possession of the decryption key which allows him to access the content as often as desired. Differentiated license models are not intended—however, if license enforcement additionally took place on the client-side, such models could be implemented. As content download and license buying are done anonymously, none of the parties can build user profiles. [3] presents a privacy-preserving DRM scheme for multiparty scenarios without a TTP. A user anonymously requests a token set from the content owner that allows anonymous purchase of content licenses from content providers (CPs). A drawback is that CPs are able to build usage profiles of content executions under a pseudonym. [4] presents a DRM scenario that allows users to anonymously buy content from any CP and execute it at any computing center within the cloud. The users' permission to execute the content is checked before every single execution. Their solution is resistant against profile building. The authors suggest employing a re-encryption scheme based on secret sharing and homomorphic encryption to achieve unlinkability of executions. The approach is extended in [5] by employing an adapted version of proxy re-encryption [6]. The scheme makes explicit use of a service provider as TTP. The approach towards privacy-preserving DRM in [7] also requires a TTP for license checking before execution. It makes use of a number of cryptographic primitives such as proxy re-encryption, ring signatures and an anonymous recipient scheme to provide unlinkability of executions.

3 System Model

Our multiparty DRM scenario involves CPs, CDs, and users. The focus is on mobile users with different *content access devices* (CADs) accessing content. As devices have different hardware trust anchors—e.g., smartphones are equipped with SIM cards, tablet computers have *trusted platform modules* (TPMs), etc.—we subsume those trust anchors under the term *smartcards* in the following.[1]

The CP takes the role of, e.g., a film studio or music label that produces content. Users interact with the CP by buying a license that allows playback of the content—under certain terms that are mediated. The user's smartcard is used to check whether the user is still allowed to access the content. Then, a nearby CD is contacted and the CD streams the content to the user. The CD can have contracts with different CPs, which allows the user to access content by different CPs from a single source—as it is the case with state-of-the-art streaming servers as well. The CD might get paid for providing its services by the CPs (or even the users). We do not cover this aspect in the paper at hand.

[1] SIM cards are smartcards and TPMs are a special form of smartcards as well.

We assume that CPs and CDs are *honest-but-curious*, i.e., they follow the protocol but try to find out as much as possible to track users. Users are assumed as *active adversaries*, i.e. trying to break the protocol to execute content without a license. Our protocol is not based on any TTP checking licenses.

DRM requirements:

We identify the CP, CD, and the user as stakeholders. The requirements are:

Content provider: *Req. I: Support for different license models*, *Req. II: Protection of the content (confidentiality)*, and *Req. III: Enforcement of licenses.*

User: *Req. IV: Profile building (under a pseudonym) must not be possible for any involved party.* To achieve Req. IV, the the following aspects must be met: *Anonymous content (license) buying towards content provider, and anonymous content execution towards content distributor*, *Unlinkability of content (license) purchases towards the content provider*, and *Unlinkability of content executions towards the content distributor.*

4 Privacy-preserving multiparty DRM system

System Initialization: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic groups with the same prime order q, the security parameter $n = ||q||$, $<g> = \mathbb{G}_1$, and $Z = e(g, g) \in \mathbb{G}_2$. Users are equipped with smartcards (SCs) that are programmed and shipped by trustworthy SC providers that install a private key sk_{sc} and the corresponding digital certificate $cert_{sc}$ on every smartcard. The private key and certificate are shared by all SCs since they are used for anonymous authentication towards the CP during the process of purchasing content. Authentication of SCs is required so that only legitimate SCs can be used to purchase content, however, CPs must not be able to recognize SCs. Moreover, the current time of production of the SC is set as the SC's timestamp ts. Content offered by the CP is encrypted using a symmetric encryption algorithm such as *AES* [8] and a separate content key ck_i for each content i. The user employs an anonymous payment scheme with his/her bank to get supplied with payment tokens pt.

Content Purchase: We assume that the connection between user and CP is anonymized (e.g., by using an anonymization network such as Tor [9]). The user initiates the content purchase via his/her content access device (CAD) by authenticating towards the SC with his/her PIN and initiating the TLS [10] handshake with the CP. The SC executes the KG algorithm as in [6] to generate a temporary key pair[2] ($pk\text{-}tmp_{sc} = (Z^{a_1}, g^{a_2}), sk\text{-}tmp_{sc} = (a_1, a_2)$), where $a_1, a_2 \in \mathbb{Z}_q$ are chosen randomly. During the TLS handshake, CP challenges CAD's SC with a nonce r and asks for SC's certificate. CAD forwards r to SC which signs r and $pk\text{-}tmp_{sc}$ with SC's private key sk_{sc}. The *signature* and SC's certificate $cert_{sc}$, as well as $pk\text{-}tmp_{sc}$ are forwarded to CAD and CAD forwards them, together with the $content\text{-}id_i$ of the content i to be bought, as well as the payment token pt to pay for the license. From this moment on, the communication between CAD and CP is authenticated and encrypted via TLS. CP

[2] A new temporary key pair is used for each content purchase.

verifies the response by checking the *signature*. This way, CAD's SC has anonymously authenticated towards CP, meaning CP knows that $pk\text{-}tmp_{sc}$ is from an authentic SC and the corresponding $sk\text{-}tmp_{sc}$ does not leave the SC. CP creates the license for content i. This license includes a license identifier id, a timestamp ts, the $content\text{-}id_i$, the license $terms$, and CP's certificate $cert_{cp}$. Note that the license terms depend on the license model. The license is encrypted under SC's $pk\text{-}tmp_{sc}$. Moreover, the content key ck_i for content i is encrypted under $pk\text{-}tmp_{sc}$ as well. The *license*, the *signature* of the license, the $content\text{-}id_i$ and the encrypted content key $(ck_i)_{pk-tmp_{sc}}$ are forwarded to CAD. CAD stores $(ck_i)_{pk-tmp_{sc}}$ and forwards the *license* and the *signature* to SC. The SC verifies the license's signature and decrypts the license with $sk\text{-}tmp_{sc}$. Then it checks whether the id was not used before and whether ts is newer than the current ts on the SC—both to prevent replay attacks. The SC's ts is then set to the newer ts of the license.[3] Finally, the license is stored under the $content\text{-}id_i$ on the SC.

Content Execution: To playback the purchased content, the user first selects a CD of his choice (this choice could be automated as well, e.g., dependent of the region the user currently is in). We assume that the connection between user and CD is anonymized (e.g., by using Tor [9]). The CAD establishes a TLS connection [10] with CD—CD authenticates towards CAD with its certificate. CAD afterwards requests a new certificate from CD. CD creates a new key-pair using KG as in [6]: $(pk\text{-}j_{cd} = (Z^{a_1}, g^{a_2}), sk\text{-}j_{cd} = (a_1, a_2))$, where $a_1, a_2 \in \mathbb{Z}_q$ are chosen randomly and j denotes the j^{th} request to the CD. The $pk\text{-}j_{cd}$ is included in the newly generated certificate $cert\text{-}j_{cd}$, as well as a unique certificate id and the current timestamp ts. CD self-signs the certificate[4]. The certificate is forwarded to the CAD. The user authenticates towards the SC with his PIN entered on the CAD and the SC then forwards the list of available *content-id*s to CAD. The user chooses the $content\text{-}id_i$ to be executed and forwards it, together with $cert\text{-}j_{cd}$ to SC. SC checks whether the signature of $cert\text{-}j_{cd}$ is valid, whether CD was certified by a known CA, whether the certificate id was not used before and whether the ts is newer than the current ts on the SC. If these tests pass, the new ts from the certificate is set on the SC. It is important to note, that SC checks whether the certificate really belongs to a CD. If this was not the case, the user might be able to launch an attack by including a self-signed certificate that he has generated himself. Hence, if SC would not verify that the certificate belonged to a CD, the user might acquire a re-encryption key from SC that allowed him to decrypt the content key, granting him unlimited access to the content. Furthermore, SC checks whether the license $terms$ still allow the content to be played back. If this is the case, the $terms$ are updated. Then, SC generates the re-encryption key $rk_{pk-tmp_{sc} \to pk-j_{cd}}$ by using the RG algorithm as in [6], taking as input CD's public key $g^{a_2} \in pk\text{-}j_{cd}$, and its own private key $a_1 \in sk\text{-}tmp_{sc}$ (as created during the content purchase). The re-encryption key is then forwarded to CAD. CAD re-encrypts the encrypted content key

[3] Note that the SC does not have an internal clock and thus cannot keep track of (authenticated) time. The time can only be set via new and verified licenses.

[4] The signing certificate was issued by a valid certificate authority, though.

$(ck_i)_{pk-tmp_{sc}}$ by employing the R algorithm as in [6] with $rk_{pk-tmp_{sc} \to pk-j_{cd}}$ as input to retrieve $(ck_i)_{pk-j_{cd}}$—i.e., the encrypted content key under CD's public key. The re-encrypted content key is then forwarded to CD and CD decrypts the ciphertext using the D algorithm as in [6] with its private key $a_2 \in sk\text{-}j_{cd}$ as input to retrieve ck_i. The content—retrieved from CP—can now be decrypted by CD using ck_i and the symmetric scheme as employed during system initialization. Eventually, the content is provided, for example, streamed, to the user's CAD.

Authorization Categories [11]: There might be content that should not be accessible to everybody, such as X-rated content. Before initially obtaining a SC, the user provides certain information to the SC provider (e.g., his passport). The SC provider will then securely[5] store the required information on the user's SC. If we assume that the user's SC now contains information like the user's date of birth or home country, it can check whether or not the user is allowed to access content. This means that if the user requests access to, for instance, X-rated content, the SC checks the user's date of birth and according to this information either allows or denies access to the queried content.

5 Evaluation and Discussion

Performance Analysis: The user's CAD performs the re-encryption of the content key. CP and SC are involved in a challenge-response protocol for authentication of SC which is not too expensive. Further, CP has to encrypt the content key using SC's public key and the content using the content key. The latter is a symmetric encryption executed only once per content. Additionally, CD decrypts the re-encrypted content key as well as the content obtained from CP. The required generation of keys is not expensive. We show that current smartphones are easily capable of executing the required tasks by implementing a demo application on an Android smartphone. We have implemented the re-encryption using the jPBC (*Java Pairing Based Cryptography*) library[6]. The app that has been developed re-encrypts 128 Bytes of data—the length of a symmetric key to be encrypted—in $302\,ms$ on a Samsung Galaxy Nexus ($2 \times 1.5\,GHz$) running Android 4.2. Due to a lack of a proper SC[7], we could not implement the re-encryption key generation algorithm RG as in [6]. Thus, to show the practicability of the implementation, we must refer to [12]. The authors have implemented elliptic curve scalar point multiplications and additions for a smartcard in C and Assembler—which are needed in our approach as well. As the authors conclude, the standard Javacard API (version 2.2.2) cannot be used as the available EC Diffie-Hellman key exchange only provides the hashed version of the key derivation function. [12] However, we need the immediate result of the key derivation function, i.e., the result of the EC point multiplication. Our own implementation of the EC point multiplication on the smartcard's CPU did not

[5] Secure storage in this context especially means integrity-protection

[6] `http://gas.dia.unisa.it/projects/jpbc/`

[7] According to the specifications, the NXP JCOP card 4.1, V2.2.1 can be used to implement the needed functionality.

yield practicable results—as the efficient cryptographic co-processors could not be utilized due to proprietary code.

Evaluation of Requirements: CP is able to provide different kinds of rights to users for content playback. Our system allows for the most popular models like *flatrate*, *execute at most n-times*, *execute until a certain date*, etc., and thus we meet *Req. I*. CP distributes its content only in encrypted form. Thus, none of the parties not in possession of the content decryption key is able to access the content and our protocol meets *Req. II*. Smartcards, as trusted devices, are used in our protocol to enforce licenses. Thus, if the SC's check of a license fails, the re-encryption key is not generated and the user is not able to execute the content. A replay attack with an "old" CD certificate fails as the SC does not accept the ts—since it is older than the current one stored on the SC. The SC's property of tamper-resistance is required since we assumed users to be active adversaries. Thus, we meet *Req. III*. Concernng *Req. IV* we have:

(1) Users anonymously pay for content (licenses), i.e., they do not need to register with CP/CD and need not provide their payment details, which is why they stay anonymous during their transactions with CP and CD.

(2) All SCs use the same certificate for anonymous authentication towards CP, thus CP cannot link different purchases made with the same SC. SC's public key $pk\text{-}tmp_{sc}$ is newly generated for each content (license) purchase—preventing CP from linking purchases to each other. Moreover, the anonymous payment scheme provides unlinkability of individual payments. Furthermore, we assumed the connection between user and CP to be anonymized via Tor. Thus, unlinkability of content (license) purchases is achieved.

(3) The user only provides the re-encrypted content key to CD. Content i is only encrypted once during initialization with ck_i and thus, ck_i does not contain any information connected to the user or the user's CAD. As a new re-encryption key is generated for each content execution, the encrypted content key "looks" different for CD each time and hence, CD cannot link any pair $(ck_i)_{pk-j_{cd}}$, $(ck_i)_{pk-k_{cd}}$, for $j \neq k$ to each other. Further, we assumed the connection between user and CD to be anonymized via Tor. Therefore, multiple transactions executed by the user are unlinkable for the CD.

Moreover, even if an attacker gets access to the user's CAD, he does not learn which content has been bought and executed. The list of available content is only revealed by the SC after authentication with the proper PIN and the CAD application does not keep track of executed content. Thus, to sum it up, profile building (even under a pseudonym) is neither possible for CP nor CD.

Comparison to related work: In Tab. 1 we compare our proposed scheme to related work in the field of privacy-preserving digital rights management.

Need for TTP: One of the main advantages of our scheme compared to related work is that it does not need a trusted third party which is involved in the license checking process as in [5, 7] during each content execution. In [2], the license server constitutes the TTP. However, it is not involved in the protocol for each single content execution but only once, when retrieving the license.

Table 1: Comparison of our scheme to related work in terms of properties.

Properties	Paper at hand	[7]	[5]	[2]	[3]
Need for TTP	no	yes	yes	yes	no
Need for trusted hardware	yes	no	no	no	yes
Support for differentiated license models	yes	yes	yes	no	yes
Unlinkability of content executions	yes	yes	yes	yes	no
Computational efficiency	good	medium	bad	good	good
Flexibility in choosing content distributor	yes	yes	yes	yes	yes

Need for trusted hardware: In our protocol a smartcard performs the license checking. Trusted hardware is not needed by other protocols that rely on some TTP. A trusted platform module (TPM) is needed in the protocol presented in [3] to securely store tokens at the user's computing platform.

Support for differentiated license models: The protocol presented here and in [5, 7] allow for differentiated license models. The protocol presented in [2] does not allow such flexibility—once a license is bought for some content, it may be executed by the user as often as desired. The authors of [3] do not clearly state whether differentiated license models are intended. From the protocol's point of view, it should be possible to implement, e.g., *execute at most n times*-models as a token set provided by the content owner. Such token sets could include n tokens. Further, licenses that allow only a single content execution could be mapped to each token by the content provider[8] later on.

Unlinkability of content executions: All of the approaches covered here, except for [3], provide unlinkability of content executions and thus, prevent any party from building a content usage profile (under a pseudonym).

Computational efficiency: In terms of computational overhead, our proposed scheme is very efficient, as discussed above. The scheme presented in [7] makes use of a number of different cryptographic primitives and thus performs less well. In [5], the entire content is re-encrypted for each content execution. Efficient standard cryptographic primitives are used in [2, 3].

Flexibility in choosing content distributor: All the schemes presented in this overview provide users with the possibility to freely choose the CDs. In other two-party DRM scenarios, such a flexibility is typically not provided.

6 Conclusion

We have come up with a privacy-preserving multiparty DRM concept. Users anonymously buy content licenses from a CP and anonymously execute the content at any CD by, for example, streaming the content from CDs nearby.

[8] Content distributor in our scenario.

Anonymity in this context means that none of the involved parties is able to build a content usage profile—not even under a pseudonym. In contrast to related work on privacy-preserving DRM, our approach does not require a trusted third party. We implemented our concept on a state-of-the-art smartphone and proved its practicability for a multiparty DRM scenario in a mobile environment in which a user buys a license allowing the playback of, e.g., some TV show—roaming in different regions, the user is free to choose the nearest streaming server (content distributor) and hence, getting the best throughput.

References

1. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: Proceedings of the 2008 IEEE Symposium on Security and Privacy. SP '08, Washington, DC, USA, IEEE Computer Society (2008) 111–125
2. Mishra, D., Mukhopadhyay, S.: Privacy rights management in multiparty multilevel DRM system. In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics. ICACCI '12, ACM (2012) 625–631
3. Win, L.L., Thomas, T., Emmanuel, S.: A privacy preserving content distribution mechanism for DRM without trusted third parties. In: IEEE International Conference on Multimedia and Expo (ICME). (july 2011) 1–6
4. Petrlic, R., Sorge, C.: Privacy-preserving DRM for cloud computing. In: Proceedings of the 26th International Conference on Advanced Information Networking and Applications Workshops, IEEE Computer Society (2012) 1286–1291
5. Petrlic, R.: Proxy re-encryption in a privacy-preserving cloud computing DRM scheme. In: Cyberspace Safety and Security, Springer (2012) 194–211
6. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. **9** (February 2006) 1–30
7. Joshi, N., Petrlic, R.: Towards practical privacy-preserving digital rights management for cloud computing. In: Proceedings of The 10th Annual IEEE Consumer Communications & Networking Conference (CCNC 2013). (2013) 259–264
8. National Institute of Standards and Technology (NIST): Advanced Encryption Standard (AES) (FIPS PUB 197). (Nov. 2001)
9. Dingledine, R., Mathewson, N., Syverson, P.: TOR: the second-generation onion router. In: Proceedings of the 13th conference on USENIX Security Symposium - Volume 13. SSYM'04, Berkeley, CA, USA, USENIX Association (2004) 21–21
10. Internet Engineering Task Force (IETF): The Transport Layer Security (TLS) Protocol, Version 1.2 (Aug. 2008) RFC 5246.
11. Perlman, R., Kaufman, C., Perlner, R.: Privacy-preserving DRM. In: Proceedings of the 9th Symposium on Identity and Trust on the Internet. IDTRUST '10, New York, NY, USA, ACM (2010) 69–83
12. Ullmann, M., Kügler, D., Neumann, H., Stappert, S., Vögeler, M.: Password authenticated key agreement for contactless smart cards (2008) Proceedings of Workshop on RFID Security (RFIDsec08).