Claudia Eckert, Frederic Stumpf
Fraunhofer Insitute for Secure Information Technology (SIT), Germany
{claudia.eckert|frederic.stumpf}@sit.fraunhofer.de
Omid Tafreschi
Technische Universität Darmstadt, Germany
tafreschi@is.tu-darmstadt.de

ON CONTROLLED SHARING OF VIRTUAL GOODS

Abstract: Digital Rights Management Systems aim at protecting copyrighted virtual goods against illegal distribution and usage. They enable content owners to formulate usage policies, which are supposed to be enforced on consumers' devices. However, prevailing systems have two shortcomings. They give little attention to consumers' expectations by being too restrictive and they fail at protecting digital content, since they build up their protection techniques on unreliable basis, e.g., obscurity methods. This paper presents an approach that overcomes these two shortcomings. It supports the transfer of digital content between users and devices and, thus, provides more flexibility. This transfer is based on mutual consent between a consumer and a content provider at the time of purchase and happens in a preconcerted and reliable way. In addition, the transfer between users does not require any interaction with the content provider at the time of transfer.

Keywords: Trusted Computing, Digital Rights Management, Policy Enforcement

1. Introduction

Efficient compression algorithms and high bandwidths allow users to share digital content, such as music, films, and books, over communication networks at low cost. This led to the emergence of file sharing networks which are often used for illegal distribution of digital content. The content owner and the content provider respond to that illegal distribution with the concept of Digital Rights Management (DRM). Basically, DRM aims at controlling the usage of digital content. For this purpose, so-called DRM systems were developed. They enable content providers to define usage rules with the help of Rights Expression Languages (RELs). RELs consist of a grammar (a schema), also called the language concept [10], that describes the basic grammar for the expression of rights terms and conditions, and their relation to the assets in question and the parties involved. The resulting right definition is referred to as a license object. Since this object defines the usage of certain content for a certain customer, the rights object itself and the applications processing it, i.e., DRM reference monitor, and the rendering application, should be protected against manipulation. Otherwise, the content could be used in a way other than defined by the content provider.

Existing DRM systems have two shortcomings. First, they build their protection measures on the basis of unreliable software-based solutions, such as obscurity techniques. Publicly available documentation for common rights management systems from Microsoft [13] or Apple [2] does not disclose how they resist replay attacks for their (proprietary) dynamic license implementations. Obscurity techniques are as long effective as long as the obscurity is undiscovered. Moreover, most of the current DRM solutions are closed software and cannot be verified for inherent security flaws. In addition, this approach may lead to unwanted side-effects. A prominent example for these unwanted side-effects is the Sony BMG CD copy prevention scandal [11]. Sony BMG included a software for copy prevention on music CDs. This software was automatically installed on Windows computers when customers tried to play their CDs. This installation has proven to be particularly problematic since the installed software interferes with the normal way in which the Windows operating system plays CDs. To make matters worse, it created security holes that were exploitable by malicious software such as worms or viruses [14].

Second, existing DRM systems have been developed from the content providers' point of view. Consequently, they do not consider consumers' expectations. As stated in [7], the neglect of consumers' expectations may hamper the overall acceptance of DRM. According to [6], 75% of the consumers want to share music with others and would pay more for this kind of usage.

Against this background, we realize that a successful DRM system has to respect consumers' expectations while providing reliable security mechanisms to protect digital content against unauthorized usage.

Trusted computing technologies provide the necessary technical foundation for building reliable systems which behave in an expected way. They basically provide a set of techniques to enforce security policies. Although trusted computing has a bad reputation, we believe that this technology has the necessary potential for a wide range of sensitive applications.

In this paper, we present an approach that overcomes the two shortcomings mentioned above. It supports the transfer of digital content between users and, thus, provides more flexibility. This transfer is based on mutual consent between a consumer and a content provider at the time of purchase and happens in a preconcerted and reliable way. In addition, the transfer between users does not require any interaction with the content owner at the time of transfer.

This paper is organized as follows: First, we give an introduction to trusted computing technologies and explain the concept of remote attestation in Section 2. We present a masquerading attack against this concept in Section 2.2. In Section 3, we propose a new protocol which is robust against masquerading attacks. This protocol can be used for developing a robust DRM system that supports transfer of licenses to other users or devices. We then present such a DRM system in Section 4. We discuss related work in Section 5 and conclude in Section 6.

2. Background on Trusted Computing Mechanisms

The introduction of the Trusted Platform Module (TPM) which is specified by the Trusted Computing Group (TCG) provides vital functions for IT-security. Besides the potential of offering a protected storage for storing cryptographic keys, the TPM also offers the possibility to attest to the configuration of the local platform to a remote platform which is called remote attestation (RA). This process is of particular interest in the context of DRM [12] or Enterprise Security [16] to ensure that a client platform is trustworthy and is behaving in accordance with a defined policy. This becomes relevant when sensitive data is transferred which should only be used in an authorized way, i.e., the content can be used by an authorized user on a specified platform. To detect attacks that manipulate the enforcement mechanisms by modifying the local client software, one may validate whether the client platform is untampered using RA. Thus, the sender has a confirmation about the trustworthiness of the platform in question. However, RA protocols are not per se resident against masquerading attacks. We explain this issue in Section 2.2

The core of the TCG mechanisms [22] is the TPM, which is basically a smartcard soldered on the mainboard of a computer. The TPM serves as the root of trust, because its hardware implementation makes it difficult to tamper with, and therefore it is assumed to be trustworthy. One must also assume that the hardware vendor is trustworthy and has designed the TPM chip according to the specification. Although the TPM chip is not specified to be tamper-resistant, it is tamper-evident, meaning that unauthorized manipulations can be detected.

The TPM provides a mechanism to store software integrity values in a protected storage, more precisely, inside so called Platform Configuration Registers (PCRs), which are initialized on power up. Software components are measured by a measurement engine and the corresponding hash-values are securely stored inside the PCRs. For every measured component an event is created and stored in the stored measurement log (SML). The PCR values can then be used together with the SML to validate the state of a platform. To make sure that these values are authentic, they are signed with a non-migratable TPM signing key, the Attestation Identity Key (*AIK*). The remote platform can compare these values with reference values to see whether the platform is in a trustworthy state or not.

2.1 Remote Attestation

RA is used to attest to the configuration of an entity to a remote entity. This procedure can be used to get integrity information before a client proceeds with the communication in order to use a service or receive data, e.g., digital content. The underlying mechanism of a RA protocol is referred to as integrity reporting. Sailer et al. [17] state that the goal of integrity reporting protocols is "to enable a remote system, i.e., the challenger, to prove that a program on another system, i.e., the attesting system owned by the attestor, is of sufficient integrity to use". The integrity reporting protocol proposed by Sailer et al. is based on a simple challenge-response authentication scheme. We will briefly explain this protocol by looking at the transmitted messages shown in Figure 1. In this figure, Platform *A* requests integrity information by delivering a nonce to *B*. *B* proves freshness of the integrity of its system configuration by signing the current PCR values and the delivered nonce with an *AIK*. Further details on this protocol can be found in [17] and [21].

$$
\begin{array}{lcll}
A & \rightarrow & B & : Na \qquad (1) \\
A & \leftarrow & B & : \mathrm{SML},\ Cert(AIK,\ K_{AIK}),\ \{Na,\ PCR\}_{K_{AIK}^{-1}} \qquad (2)
\end{array}
$$

Figure 1. Integrity Reporting Protocol [17]

We have shown in [21] that this integrity reporting protocol is vulnerable to a masquerading attack. We will shortly outline this attack in the next section.

2.2 Masquerading Attack Scheme

The attacker considered here has two platforms under his control. One platform runs a trustworthy operating system with the client software that enforces a certain policy, e.g., the DRM reference monitor. This platform (*C*) represents the client that is conform to the original client software and therefore untampered. This platform is also equipped with a genuine TPM that supports the policy enforcement of the DRM system. The attacker is also in control of one malicious client platform (*M*) that wants to gain control of protected digital content. We refer to this client as malicious client, since his enforcement mechanism has been tampered with and it is not conform to the original client software. We require for our attack that *C* answers the request from *M*, i.e., *C* is not configured to answer only requests from *A*.

In our attack scheme, the attacker bypasses the remote attestation of *M*, by using the platform configuration of the honest client *C* to attest his malicious client running on *M*. It should be noted that this attack is successful even if this integrity reporting protocol is integrated inside another established cryptographic channel, such as TLS. Details on this issue can be found in [21].

Figure 2 depicts the attack against the integrity reporting protocol. The challenging party *A* wants to securely validate the integrity of the attesting malicious system *M* using the protocol proposed by Sailer et al. The malicious system *M* itself transfers all messages from *A* to the honest client *C*.

$$
\begin{array}{lclll}
A & \rightarrow & M & : Na & (1) \\
M & \rightarrow & C & : Na & (2) \\
M & \leftarrow & C & : \text{SML},\ Cert(AIK,\ K_{AIK}),\ \{Na,\ PCR\}_{K^{-1}_{AIK}} & (3) \\
A & \leftarrow & M & : \text{SML},\ Cert(AIK,\ K_{AIK}),\ \{Na,\ PCR\}_{K^{-1}_{AIK}} & (4)
\end{array}
$$

Figure 2. Masquerading Attack on the Integrity Reporting Protocol [21]

The shortcoming of the integrity reporting protocol is caused by the restricted usage possibilities of *AIK*s. According to [22], the *AIK*s can exclusively be used for proving the authenticity of a platform. This means *AIK*s can neither be directly used to establish secure channels nor to authenticate communication partners. Therefore, the challenger cannot be sure whether the received message belongs to the attesting system. He only knows that he received a message from a genuine TPM. Additionally, the possession of multiple *AIK*s is possible, the only requirement is, that the corresponding certificate must be certified by a trusted Privacy-CA thus belonging to a valid endorsement key. As a consequence, the challenging party cannot verify the integrity of a platform only with the help of the *AIK*s belonging to that platform.

Even if the server has the certificate which belongs to the *AIK* and forbids the creation of new *AIK*s, masquerading attacks cannot be prevented, since the attesting system *M* pretends that it is in possession of the corresponding K^{-1}_{AIK} by using the integrity values with a valid *AIK* signature from platform *C*. The challenging system is therefore not able to detect this attack, since the attesting system delivers all information that identifies it as platform *C*. After the platform is authorized, the malicious platform is assumed to be trusted.

We explain the relevance of the masquerading attack by a concrete usage scenario, i.e., the initialization phase of a DRM system. At this time, the DRM system has not yet exchanged a cryptographic key with the server. This key is needed to bind the digital content to the client platform by means of encryption. Before exchanging the key, the server validates the state of the client platform in order to ensure that the client is running a trustworthy DRM system. If the attestation is successful, the DRM system supports the user by the creation of an account and an appropriate cryptographic key on the client machine. This key is later used to establish a mutual authenticated channel (e.g., a TLS-channel) which also authenticates the user. After the initialization, the server encrypts the digital content using the client key and transfers it to the client. If now an attacker performs a masquerading attack as outlined in Figure 2, this key gets bounded to the platform configuration on which the enforcement of usage rules are bypassed, i.e., the malicious system. Thus, it cannot be ensured that only trustworthy clients can access protected digital content.

3. A Robust Integrity Reporting Protocol

In this section, we present a protocol to overcome the previously described shortcoming of integrity reporting protocols. A detailed description and evaluation of this protocol is given in [20]. The presented protocol is able to prevent masquerading attacks since a potential attacker, who is placed between the server and the collaborative host, is excluded from the following communication flow.

To protect the integrity reporting protocol against masquerading attacks, we enhance it with a key agreement protocol. The modified integrity reporting protocol is shown in Protocol 3.1 with the extension to use Diffie-Hellman parameters. It is essential for the protocol that both parties agree on one common generator *g* and one common group *m*. The asymmetric keys are then generated and computed as described in [5].

The major enhancement is that the prover generates $g^b \mod p$ and includes the resulting Diffie-Helman public key into its signed attestation response. The public key is also transmitted to the challenger in plaintext using $Cert(AIK,K_{AIK})$ to enable the challenger to validate the *AIK* signed message. Because *B* is running a trusted OS with a trusted platform configuration, the secret part of the DH-key is not accessible to a potential malicious client. Finally, both parties compute the shared session key $K_{session}$ which is also used to deliver the privacy-critical SML.

We have implemented the protocol and run performance measurements on the computational overhead introduced through the additional cryptographic operations [20]. The additional cryptographic operations increase the time for answering one attestation request by 0.70% on an Atmel 1.2 TPM compared to the protocol proposed by Sailer et al. The complete time for execution of the protocol and for establishing the session key $K_{session}$ is smaller than 1 second. Thus, the protocol is very efficient.

Protocol 3.1: Robust integrity reporting protocol

SUMMARY: B answers the attestation challenge of platform A
RESULT: Key establishment with key confirmation: Confidential integrity reporting

1. *Notation.*
   $K_{session}$ denotes the negotiated key between A and B.

2. *System Setup.*
   A must possess the certificate of the Privacy-CA to validate *Cert(AIK,* K_{AIK}*)*.

3. *Protocol messages.*

$$
\begin{array}{lcll}
A & \rightarrow & B: Na,\ g^a \bmod p, g,\ p & (1)\\
A & \leftarrow & B: Cert(AIK,\ K_{AIK}),\ \{Na,\ PCR,\ g^b \bmod p\}_{K^{-1}_{AIK}} & (2)\\
A & \rightarrow & B: \{Nb,\ g^a \bmod p\}_{K_{session}} & (3)\\
A & \leftarrow & B: \{Nb,\ Na,\ SML,\ g^b \bmod p\}_{K_{session}} & (4)
\end{array}
$$

4. *Protocol actions.*
   (a) *Precomputation by A. A* selects an appropriate prime p and generator g of $\mathbb{Z}_p^*(2 \leq g \leq p-2)$.
   (b) A chooses a random secret $a, 2 \leq a \leq p-2$, and computes $g^a \bmod p$.
   (c) *Attestation challenge. A* sends message (1) to B.
   (d) B also chooses a random secret $b, 2 \leq b \leq p-2$ and computes $g^b \bmod p$.
   (e) B computes the attestation response message by signing the own public key, Na and a set of PCRs using an *AIK*.
   (f) *Attestation response.* B delivers the signed message together with the *AIK* credential to A (2). Finally, B computes $K_{session} = (g^a)^b \bmod p$.
   (g) A verifies whether the delivered credentials are authentic and verifies if all signatures are valid.
   (h) A also receives $g^b \bmod p$ and computes $K_{session} = (g^b)^a \bmod p$.
   (i) *Key-Confirmation. A* generates a second non-predictable nonce (*Nb*) and transfers message (3) to B
   (j) B decrypts the message (3) and transfers message (4) to A.
   (k) *Platform validation. A* checks the freshness of message (4) using the delivered nonce (*Nb*). Based on the received *SML* and the *PCR* values, A processes the *SML* and re-computes the received *PCR* values. If the computed values match the signed *PCR* values, the *SML* is valid and untampered. Finally, A has to verify that the delivered integrity reporting values match the given reference values; thus, A can decide if B is in a trusted system state.

This robust integrity reporting protocol can be used as a building block for robust DRM systems. We present such a system in the following section.

4. A Fair Digital Rights Management System

In this section, we present a fair DRM system. As argued in Section 1, existing DRM systems neglect consumers' expectations. In particular, they bind legally purchased content to consumers' hardware and do not allow consumers to transfer their content to their other devices or to other users. However, 75% of consumers want to share music with others and would pay more for this kind of usage [6]. This willingness to pay would enable content providers to set up a new range of business models. In the following, we sketch a DRM system which enables consumers to transfer their legally purchased content.

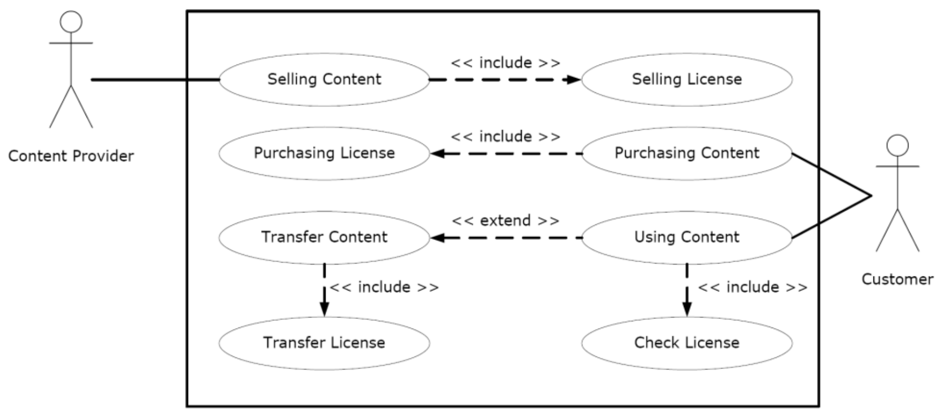


Figure 3. Use Case Diagram of Fair DRM System

Figure 3 shows the use cases supported by our DRM system. The system enables a content provider to offer his digital content. As discussed in Section 1, selling digital content usually implicates selling the associated license. Accordingly, purchasing digital content implicates purchasing the corresponding license. The DRM system checks the availability of an appropriate license each time the customer wants to use, e.g., to render, digital content. A special kind of usage supported by our DRM system is transferring digital content from one user to another. In this case the according license or a part of it is transferred, as well. Figure 4 depicts the architecture of our DRM system.

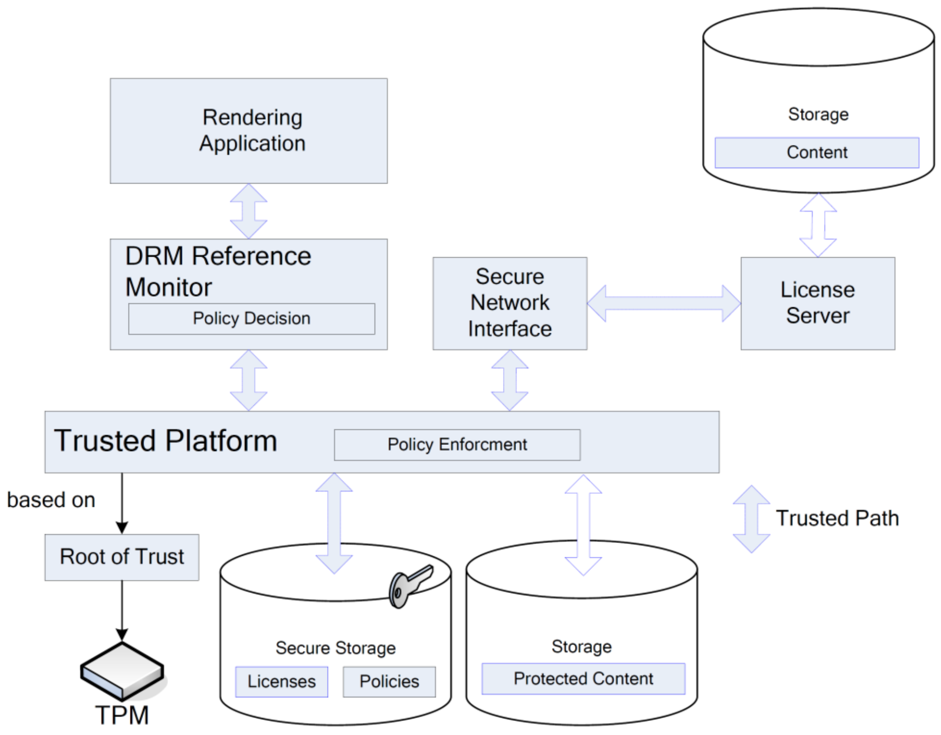


Figure 4. Architecture of the Fair DRM System

Figure 4 shows two kinds of applications, i.e., a rendering application on the client side and a license server. Each time a rendering application wants access to protected digital content, the DRM reference monitor is contacted which decides whether the access is granted or denied. Thus, the DRM reference monitor has similar characteristics as a policy decision point. The DRM reference monitor itself runs on top of a trusted platform. This platform offers two functionalities, i.e., a secure storage and a secure network interface. The secure storage is necessary for storing licenses for digital content and policies, which determine the behavior of the DRM reference monitor, e.g., evaluating the trustworthiness of other platforms. The secure storage protects licenses and policies by sealing all data stored inside the secure storage, i.e., binding data to the state of the complete platform including the DRM reference monitor. Thus, sealed data can only be read if the DRM reference monitor is untampered and in a trusted state. We assume that access to the secure storage is only possible by the DRM reference monitor. This means that it is not possible to access the secure storage with different means rather than the interface provided to the DRM reference monitor. The secure network interface is responsible for building trusted communication channels. This is done using the robust integrity report protocol presented in Section 3.

Basically, the DRM system offers three functionalities: (I) purchasing content, (II) using content and (III) transferring content. We present the necessary protocols in the following sections.

4.1 Purchasing Content

The protocol for the purchasing process is depicted in Figure 5. At the beginning, the content provider has to verify the trustworthiness of a client's platform, i.e., its DRM reference monitor. The integrity reporting protocol which we presented in Section 3 is used for this purpose. The result of this step is the shared session key $K_{session}$. This key is used in the next step to encrypt the customer's order including information about the digital content he wants to purchase and his intended usage. The encrypted order is transmitted to the content provider. The content provider decrypts the order with the help of $K_{session}$, generates a symmetric key, i.e., $K_{content}$ and encrypts the desired content with that key. Afterwards, he generates a license including the usage rules and $K_{content}$. The license itself is encrypted using $K_{session}$. The encrypted license and the encrypted content are sent to the customer.

After decrypting the license, it is bound to the client's platform configuration by sealing it. The client saves the encrypted content without decrypting it.

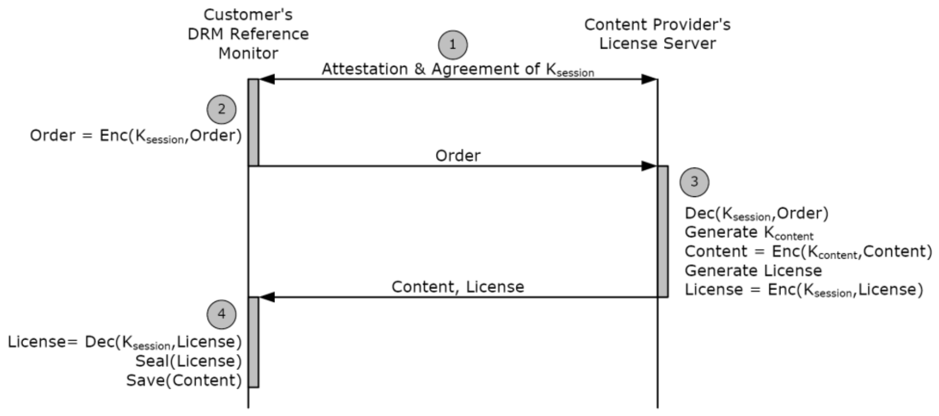

Customer's DRM Reference Monitor
Content Provider's License Server
1
Attestation & Agreement of $K_{session}$
2
Order = Enc($K_{session}$,Order)
Order
3
Dec($K_{session}$,Order)
Generate $K_{content}$
Content = Enc($K_{content}$,Content)
Generate License
License = Enc($K_{session}$,License)
Content, License
4
License= Dec($K_{session}$,License)
Seal(License)
Save(Content)

Figure 5. Protocol for Purchasing Digital Content

4.2 Using Content

Figure 6 depicts the process of using content. A rendering application and a DRM reference monitor are involved in this process. Both applications are located at the user's platform (Compare Figure 4). At the beginning, the rendering application has to prove its trustworthiness using platform attestation. Based on the received integrity information, the DRM reference monitor decides whether the rendering application is trustworthy or not. For this, we use the protocol proposed in Section 3. The resulting key $K_{session}$ is used to ensure confidentiality during the next steps. The rendering application prepares a UsageRequest. This message is encrypted with $K_{session}$ and contains information about the content and the intended usage. After receiving the UsageRequest, the DRM reference monitor decrypts it with $K_{session}$ and checks the availability of the corresponding license. If the license is available, the DRM reference monitor checks whether the customer possesses the necessary usage rights which are given in the UsageRequest. In the case of a positive result, the DRM reference monitor decrypts the encrypted content with $K_{content}$ which is stored in the license. Afterwards, the license is encrypted again with the $K_{session}$. This approach ensures that only the trusted rendering application can access the content. Before sending the encrypted content, the DRM reference monitor updates the license. This is an optional action and is only necessary when the license limits the number of usage. After receiving the encrypted content, the rendering application decrypts it with $K_{session}$ and renders it to the user.

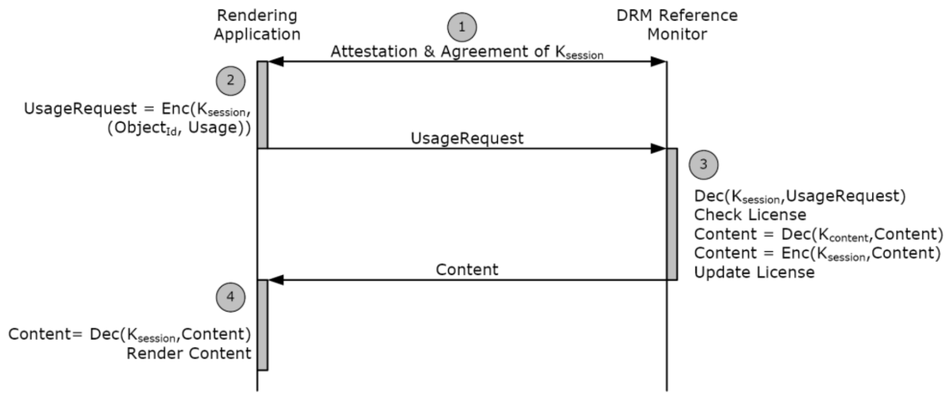
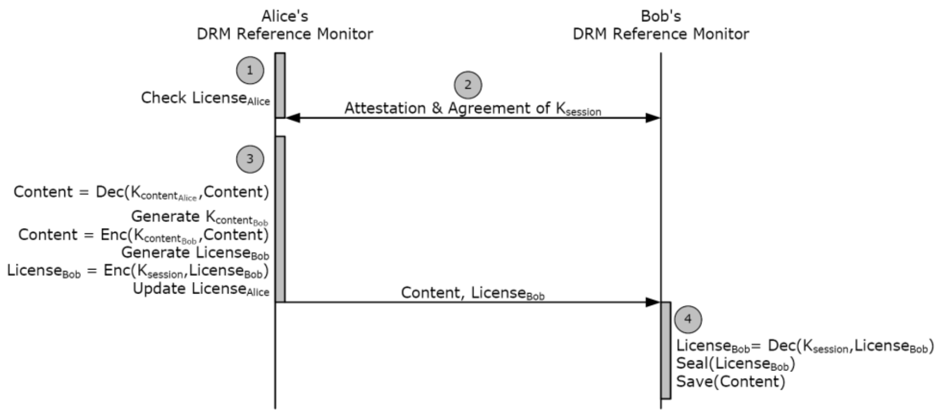

Rendering Application
1
DRM Reference Monitor
Attestation & Agreement of K_session
2
UsageRequest = Enc(K_session, (Object_Id, Usage))
UsageRequest
3
Dec(K_session,UsageRequest)
Check License
Content = Dec(K_content,Content)
Content = Enc(K_session,Content)
Update License
Content
4
Content= Dec(K_session,Content)
Render Content

Figure 6. Protocol for Using Digital Content

A critical attack on the secure storage of our DRM system is a replay attack, where a user replays the actual secure storage with an old secure storage. This attack is especially relevant if a user only possesses a limited number of usage rights on a particular protected content. To achieve protection against attacks of this type, the monotonic counter of the TPM could be used as already explained in [15] or [19].

4.3 Transferring Content

Figure 7 shows the process of transferring digital content from one customer to another customer. This kind of usage is a key feature of our DRM system and does not only support the exchange of digital content between customers. In addition, it allows the transfer of digital content between devices of one customer.



Alice's DRM Reference Monitor
Bob's DRM Reference Monitor
1
Check License$_{Alice}$
2
Attestation & Agreement of K$_{session}$
3
Content = Dec(K$_{content_{Alice}}$,Content)
Generate K$_{content_{Bob}}$
Content = Enc(K$_{content_{Bob}}$,Content)
Generate License$_{Bob}$
License$_{Bob}$ = Enc(K$_{session}$,License$_{Bob}$)
Update License$_{Alice}$
Content, License$_{Bob}$
4
License$_{Bob}$= Dec(K$_{session}$,License$_{Bob}$)
Seal(License$_{Bob}$)
Save(Content)

Figure 7. Protocol for Transferring Digital Content

We assume two customers in our scenario, namely Alice and Bob. Alice wants to transfer a certain digital content or part of it to Bob. At the beginning, Alice's DRM reference monitor checks whether she has the right to transfer the content in question. This permission is stored in the corresponding license, which can be accessed only if Alice's DRM reference monitor is trusted. If Alice has the right to transfer the content, her DRM reference monitor requires an attestation from Bob's DRM reference monitor to verify its trustworthiness. For this purpose, we apply the protocol presented in Section 3. The key $K_{session}$ is the result of this step and is used for the next steps to ensure that only Bob's DRM reference monitor can access the upcoming messages. After the attestation, Alice's DRM reference monitor decrypts the content using the symmetric key $K_{contentAlice}$. This key is stored in the corresponding license, which is only accessible if Alice's DRM reference monitor is untampered. After this decryption, Alice's DRM reference monitor generates the symmetric key $K_{contentBob}$ and encrypts the content using that key. Afterwards, Alice's DRM reference monitor generates a license for Bob and updates its own license. Bob's license includes the $K_{contentBob}$ and usage rights and is encrypted with $K_{session}$ and is sent together with the encrypted content to Bob's DRM. After receiving the message, Bob's DRM reference monitor decrypts the license and binds it to

Bob's platform configuration by sealing it to the actual platform configuration. Afterwards, Bob's DRM reference monitor stores the encrypted content.

The process of transferring digital content between different users or devices of one user is similar to the process of purchasing content except for two additional steps. These are checking the sender's license with respect to transfer rights and updating the sender's license before generating and transferring the receiver's license. In contrast to purchasing content, transferring content does not require any interactions with the content provider. However, the proposed protocol ensures that the transferred content can only be used according to usage rules, which were defined by the content provider at the time of purchase.

To prevent that a customer transfers digital content to another customer and then replays an old secure storage, we again propose to use the monotonic counter of the TPM. Using this concept, it can be ensured that a misbehaving customer cannot replay an old secure storage after he transferred digital content to another customer.

5. Related Work

Grimm et al. [9] give a comprehensive evaluation of current DRM systems and conclude that successful DRM systems have to fulfill two requirements. They should respect privacy concerns of consumers and should not neglect the interests of consumers. There are several proposals for fulfilling one or both of these requirements [1, 3, 8, 15].

Arnab and Hutchison introduce in [3] the concept of usage contracts. Usage contracts are defined with the help of RELs and can be the result of a negotiation between end users and rights holders. Usage contracts give end users more flexibility with respect to the usage of virtual goods. Concerning this feature, the concept of usage contract is similar to our work. However, in contrast to our work, Arnab and Hutchison do not present any enforcement mechanisms.

Sadeghi et al. [15] present a sophisticated security architecture that supports the enforcement of stateful licenses. This kind of license can be used to define rights for usages during a fixed time period or for fixed number of usages. In addition, Sadeghi et al propose a protocol based on trusted computing functionality for transferring licenses via trusted channels. Such a trusted channel establishes a session key between a content provider and the customer and is comparable to the channel established by our robust integrity reporting protocol. However, only the negotiation of the session key between content provider and the customer using a trusted channel takes between 2 and 3 seconds on an Atmel 1.1b TPM and between 23 and 24 seconds on an NSC 1.1b

TPM. In contrast to that, a complete protocol run in our approach is achieved in under one second.

Adelsbach et al. [1] propose a DRM system which enables users to create (and distribute) copies of their own content as long as the number of copies does not exceed a previously defined threshold. In this case, the identity of the user remains hidden. In cases of unauthorized usage, i.e., making too many copies, the identity of the end user can be revealed with the help of a secret sharing scheme. The basic idea of this work differs from our proposal. We define the right for making copies using the license object and prevent unauthorized usage by enforcing the usage policies.

Grimm and Aichroth [8] present the concept of Lightweight Digital Rights Management (LWDRM). This concept introduces two formats, i.e., the Local Media File (LMF) and the Signed Media File. LMF binds digital content to a hardware platform. Consequently, the content can be rendered only on a special platform. SMF can be used to eliminate this binding. To this end, the user signs the content and a digital watermark containing the user's identity and adds this data to the content. Content in SMF format can be transferred to other users and devices. The idea behind LWDRM is that a user will not illegally distribute his purchased content out of his reach, since it contains information about him. This assumption is the main difference between LWDRM and our approach. However, in contrast to [8], we do not rely on the correct behaving users. In addition, we do not add any information about users to the content.

6. Conclusion

In this paper, we addressed the issue of controlling access to digital content. First, we argued that current DRM systems rely on protection measures on the basis of unreliable software-based solutions. In addition, they do not sufficiently consider customers' expectations. To overcome these shortcomings, we proposed to use trusted computing technologies because of two reasons. First, they allow us to build hardware-based (and thus more robust) DRM systems. Second, trusted computing is an emerging technology and will thus be very widely available on different platforms. This trend will spur the interoperability of systems based on trusted computing technologies and allow consumers to transfer and use their purchased content to other devices and/or users.

We proposed a DRM system, which considers both, the requirements of content providers and the requirements of consumers. It provides a high protection level. At the same time, it allows users to transfer their purchased content to other devices or users. We have implemented the proposed DRM system. A detailed description of the prototype is given in [18].

References

[1] A. Adelsbach, U. Greveler, and J. Schwenk. *Fair DRM -Ermoeglichen von Privatkopien und Schutz der digitalen Ware*. In Tagungsband des 9. Deutschen IT-Sicherheitskongresses des BSI, 2005.

[2] Apple. *FairPlay DRM*. http://www.apple.com/itunes/, 2008.

[3] A. Arnab and A. Hutchison. *Fairer Usage Contracts for DRM.* In DRM ’05: Proceedings of the 5th ACM workshop on Digital rights management, pages 1–7, New York, NY, USA, 2005. ACM Press.

[4] E. Becker, W. Buhse, D. Günnewig, and N. Rump, editors. *Digital Rights Management -Technological, Economic, Legal and Political Aspects*, volume 2770 of Lecture Notes in Computer Science. Springer-Verlag, 2003.

[5] W. Diffie and M. Hellman. *New Directions in Cryptography*. IEEE Transactions on Information Theory, IT-22(6):644–654, 1976.

[6] N. Dufft, A. Stiehler, D. Vogeley, and T. Wichmann. *Digital Music Usage and DRM -Results from an European Consumer Survey*, May 2005.

[7] M. Fetscherin. *Evaluating Consumer Acceptance for Protected Digital Content*. In Becker et al. [4], pages 321–333.

[8] R. Grimm and P. Aichroth. *Privacy Protection for Signed Media Files: A Separation-of-Duty Approach to the Lightweight DRM (LWDRM) System*. In Proceedings of the 2004 Workshop on Multimedia and security, pages 93–99. ACM, 2004.

[9] R. Grimm, S. Puchta, M. Mueller, J. Bizer, J. Moeller, A. Will, A. Mueller, and S. Jazdzejewski. *privacy4DRM Datenschutzverträgliches und nutzungsfreundliches Digital Rights Management*. Technical report, Studie im Auftrag des Bundesministeriums für Bildung und Forschung, 2005.

[10] S. Guth. *Rights Expression Languages*. In Becker et al. [4], pages 101–112.

[11] M. Hansen. DRM-Desaster: *Das Sony BMG-Rootkit*. Datenschutz und Datensicherheit (DuD), 30(2):95–97, 2006.

[12] Q. Liu, R. Safavi-Naini, and N. P. Sheppard. *Digital Rights Management for Content Distribution*. In ACSW Frontiers ’03: Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003, pages 49–58, 2003.

[13] Microsoft.*Windows Media Rights Manager10*. http://www.microsoft.com/windows/windowsmedia/ drm/default.aspx, 2008.

[14] M. Russinovich. *Sony, Rootkits and Digital Rights Management Gone Too Far*, 2005.

[15] A.-R. Sadeghi, M. Scheibel, C. Stüble, and M. Wolf. *Play it once again, Sam - Enforcing Stateful Licenses on Open Platforms*. In Proceedings of the Second Workshop on Advances in Trusted Computing (WATC ’06), 2006.

[16] R. Sailer, L. van Doorn, and J. Ward. *The Role of TPM in Enterprise Security*. Datenschutz und Datensicherheit (DuD), 28(9):539–544, September 2004.

[17] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. *Design and Implementation of a TCGbased Integrity Measurement Architecture*. In SSYM’04: Proceedings of the 13th conference on USENIX Security Symposium, pages 16–16. USENIX Association, 2004.

[18] B. Stärz. *Technische Infrastrukturen zur Umsetzung neuer Erlösmodelle für digitale Güter*. Diplomarbeit, TU Darmstadt, 2006.

[19] F. Stumpf and C. Eckert. *Enhancing Trusted Platform Modules with Hardware-Based Virtualization Techniques*. In Proceedings of the Second International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2008), Cap Esterel, France, August 25-31 2008. IEEE Computer Society. to appear.

[20] F. Stumpf, A. Fuchs, S. Katzenbeisser, and C. Eckert. *Improving the Scalability of Platform Attestation*. In Proceedings of the Third ACM Workshop on Scalable Trusted Computing (ACM STC’08), pages 1–10, Fairfax, USA, October 31 2008. ACM Press.

[21] F. Stumpf, O. Tafreschi, P. Röder, and C. Eckert. *A Robust Integrity Reporting Protocol for Remote Attestation*. In Second Workshop on Advances in Trusted Computing (WATC’06 Fall), November 2006.

[22] Trusted Computing Group. *Trusted Platform Module (TPM) specifications*. Technical report, https://www.trustedcomputinggroup.org/specs/TPM, 2008.