Enabling DRM-preserving Digital Content Redistribution

Srijith K. Nair, Bogdan C. Popescu, Chandana Gamage, Bruno Crispo, Andrew S. Tanenbaum
Dept. of Computer Science
Faculty of Sciences, Vrije Universiteit
1081 HV Amsterdam, The Netherlands
{srijith, bpopescu, chandag, crispo, ast}@few.vu.nl

Abstract

Traditionally, the process of online digital content distribution has involved a limited number of centralised distributors selling protected contents and licenses authorising the use of these contents, to consumers. In this paper, we extend this model by introducing a security scheme that enables DRM preserving digital content redistribution. Essentially consumers can not only buy the rights to use digital content but also the rights to redistribute it to other consumers in a DRM controlled fashion. We examine the threats associated with such a redistribution model and explain how our scheme addresses them.

1 Introduction

In recent years, there has been a rapid increase in the popularity of personal devices capable of rendering protected digital contents. Some of these devices [1] also have communication capabilities that allow peer-to-peer interaction among them.

At the same time, also due to the impact of peer-to-peer file-sharing applications, content providers are looking for new and secure business opportunities in the digital world for selling their copyrighted content. Paramount to their interest is the requirement to enforce the Digital Rights Management (DRM) policies they set to protect digital content from illegal copying and unauthorized distribution.

The biggest challenge is in enforcing DRM policies *after* content has been distributed to the consumer. The current approach to solve this problem is to distribute content to only so called *compliant devices* - consumer electronics devices which by construction are guaranteed to always enforce the DRM policies associated with the content they render. Protected content cannot be legitimately distributed to non-compliant devices.

The current approach with DRM systems has so far been based on a single business model, where content providers are the only authorized distributors and the only parties entitled to collect revenue from sales of content. Technically, this is usually implemented as a closed system with very few authorized distribution points that distribute the protected content either off-line (e.g. CD shops) or on-line (e.g. Apple iTunes[2]).

We propose to extend this business model by allowing distributors to sell not only the right to use content, but also the right to redistribute or resell content in a controlled manner. From a security point of view, this is technically very challenging, because the resulting system is a network of peer-to-peer independent devices, each of them a potential consumer, authorized distributor, but also an attacker.

In this paper we present a security scheme that allows enforcement of DRM mechanisms on a network that is physically, and not just logically, peer-to-peer. Our solution takes into consideration the cost constraints that are typical for the consumer electronics market, avoiding the use of very expensive tamper-resistant hardware. Our scheme cannot prevent a single motivated attacker from violating the security of a single device and by doing so, being able to steal the digital content stored in that device. The damage can also extend to the point of illegally redistributing such a content. However, our distributed DRM enforcement mechanisms allow the detection and the exclusion of such an attacker from the system. The probability of successful detection is directly proportional to the extent of the illegal activity. Thus, our scheme is particularly suitable for detecting widespread frauds and powerful attackers rather than occasionally misbehaving users.

The rest of the paper is organized as follows. In Section 2 we present a high level model of the system, followed by a discussion on compliant devices in Section 3. The threat model is covered in Section 4. We discuss the content distribution and redistribution process in Section 5. In Section 6 we discuss payments related issues and in Section 7 we describe security mechanisms associated with the system. Finally, in Section 8 we review related work and conclude in Section 9 noting possible future work.

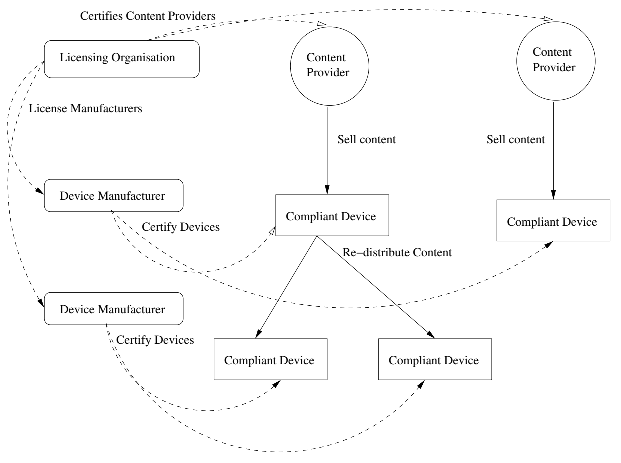

Figure 1. Entities involved in the system

2 System Model

We consider the system model as shown in Fig.1 that consists of the following entities:

- Device manufacturers: These are the manufacturers of certified compliant device. Compliant devices render digital content for consumers, while enforcing the policies set by content providers.
- Content providers: These are companies that commercialize digital content to consumers. With each item sold, providers associate a content license. The license describes the buyer's rights with respect to the content. This may include the right to render (play) the content, possibly for a limited amount of times or a limited period as well as the right to redistribute or resell the content to other consumers in a controlled manner.
- Licensing Organisation: It is an organisation trusted by all parties involved. Its public key is embedded in all compliant devices and forms the root of trust for digital certificate chains. The LO licenses device manufacturers and certifies content providers. Our model consists of these licensed manufacturers and certified content providers.
- Consumers: These are human users who buy the digital content and the associated licenses from the providers. They may or may not wish to buy the rights to redistribute/resell the content using the compliant device they possess.

Note that in Fig.1 the dotted lines represent licensing/certification flow, while solid lines represent the digital content flow.

3 Compliant Devices

In a typical DRM system, content providers and device manufacturers set up a license organisation with the aim of certifying compliant devices and (rarely) revoking the circumvented ones. Usually, the licensing organization delegates the certification task to licensed device manufacturers that are contractually bound to only produce compliant devices according the specified security rules. Devices built accordingly to these rules guarantee the enforcement of DRM policies specified by the content providers members of the licensing organization.

3.1 Requirements

To be able to guarantee DRM policies enforcement, compliant devices must incorporate a *tamper-resistant* hardware module typically implementing the following functionalities:

- A cryptographic hashing scheme (SHA-1 Engine).
- A PKI system for performing signing/verification of signatures and public key encryption/decryption (RSA Engine).
- A symmetric key block cipher scheme for bulk encryption (AES Engine).
- Secure Storage that can hold data securely in a tamper resistant manner. Because of cost constraints the size of this memory is limited (for example less than 1 MB).
- Secure scratchpad memory to store sensitive data in unencrypted state, while it is being processed.
- Embedded content decoder to prevent tapping of unencrypted content while it is being rendered, the content decoder needs to be integrated into the trusted hardware.

A Trusted Platform Module (TPM) specifications [3] compliant hardware already supports the first three functionalities with limited support for the fourth (protected storage in TPM is optimized for storage of small objects like symmetric and asymmetric keys). The latest series of multimedia processors like Freescale i.MX21 [4] supports secure internal as well as encrypted data storage and hash accelerators while providing up to 30fps MPEG-4 and H.264 decode/encode accelerations. Hence, we can demonstrate that our device requirements are practical and economical with current technology.

3.2 Compliance Testing

Compliant devices need to be able to prove their compliance so they can be trusted with protected content. This is typically done by means of cryptographic keys embedded in the device at manufacturing time. Both asymmetric [5, 6] as well as symmetric [7] key schemes have been proposed for compliance checking. In this paper we rely on the former.

The basic idea behind public-key compliance checking protocols is to assign each device (at the time of manufacture) a unique public/private key pair, with the private key securely stored in the tamper-resistant module, and public key certified through a chain of digital certificates issued by the licensing organization and/or manufacturers. In our scheme, each compliant device is loaded with an asymmetric key pair at the time of manufacturing. The private key is embedded within the tamper-resistant module and is never available outside this module.

The public key is then certified by the manufacturer who is, in turn,licensed to do so by the licensing organisation. This certificate is used to prove that a device is compliant. When device A wishes to prove its compliance to device B, it sends its public key, the digital certificate associated with this key as well as its manufacturer's credentials to B. Because A and B share the same root of trust, B can verify A's public key as that of a compliant device. This requires that compliant devices are preloaded with the public key of the licensing organisation. This key cannot be modified or deleted to prevent users from adding their own set of keys to the root of trust and thereby circumventing the compliance testing process.

4 Threat Model

All DRM architecture specification proposed so far [5, 6, 8, 9] typically includes a comprehensive discussion of the threat model and of the mechanisms for countering the various types of attacks. As we are advocating the support for secure content redistribution that extend existing DRM protection mechanisms, the architecture introduced in this paper can be seen as a superset of existing DRM architectures. Based on this observation, we structure our threat model as follows:

- Threats to the system in absence of consumer-initiated content redistribution.
- New threats introduced by the consumer-initiated content redistribution process.

4.1 Traditional Threats

With respect to the first class of threats, it is important to understand that apart from cryptanalytic attacks, which hardly represent the most severe threat, the overall security of current DRM systems relies on the tamper-resistance property of individual compliant devices. However, as previous research has demonstrated [10, 11], good tamper-resistance is difficult to achieve, especially considering the cost limitations associated with the mass produced consumer electronic devices. Given this constraint, it is realistic to assume that a highly motivated attacker might be able to circumvent the tamper-resistance property of a device. This introduces two types of threats:

- Extraction of copyrighted content from circumvented devices. This allows for unlimited distribution of protected content on the Internet (through file sharing networks [12] for example), and unrestricted rendering of copyrighted content on compliant devices, by uploading content on the device *without* the associated license.
- Extraction of cryptographic keys from circumvented devices. These keys can then be incorporated into mass-produced pirated devices. Pirated devices are typically used for unauthorized access to subscription-based digital content distribution services.

There are a variety of technical mechanisms for countering such attacks. Digital watermarking [13] prevents uncircumvented compliant devices from rendering illegally extracted content. The idea is to have copyrighted content always incorporate a watermark indicating its origin. Before rendering a piece of content, a compliant device always checks for the watermark, and refuses to process watermarked content that is not securely licensed. Digital watermarks can be removed, but this results in loss of content quality.

To counter the threat of mass produced pirated devices, most DRM systems support key revocation. Once a compromised key used for pirate devices is identified, that key can be revoked; no provider will supply content to a revoked device, rendering pirated device useless. Depending on the algorithm used for compliance checking, identifying a compromised key may only involve inspecting the digital certificate associated with the pirate device, or more complex *traitor tracing* schemes [14].

The content distribution architecture we propose can easily accommodate all these protection mechanisms against traditional threats.

4.2 Device Revocation

Devices known to be no longer compliant are revoked by the licensing organization by having their public keys listed on a device revocation list (DRL). DRLs are bundled with the digital content supplied by providers that distribute the content.

To ensure that a fresh DRL is pushed into the network, it can be distributed by the content providers along with content and with our scheme it can be also exchanged between consumers while they re-distribute content (thus, acquiring new content automatically updates the revocation information in the device).

To manage the update of old versions of DRLs, each list has a counter as version number. A device will update its cached DRL only if the version of the list it got with the content is fresher than the list in the cache.

Revocation lists can grow large, since they contain information regarding *all* compromised devices in the world. A partial solution to this problem is the partitioning of the devices, such that instead of having a worldwide list, there is a list associated with device of a specific domain (e.g, country, continent, etc.). A list can also be periodically purged from obsolete devices not capable of rendering new content that is actually what providers really care to protect.

4.3 New Threats

In addition to the traditional threats discussed in the previous section, the ability to redistribute protected content introduces additional threats, which can be divided into three categories:

- Content masquerading during the redistribution process. This occurs when a consumer tries to pass off a different content as the one requested by another consumer.

- Untrusted storage backup attacks. Because storage media in any consumer device is susceptible to failure, devices usually allow storage backup to prevent loss of content. This introduces certain types of attacks that may allow malicious consumers to restore an older version of the license file to try and obtain more content redistribution rights than originally issued.

- Unlimited content redistribution through circumvented devices. A circumvented device (unless revoked) will appear to other compliant devices as a legitimate content source. Because the illegally redistributed content is legitimate, originally provided by the content provider to the circumvented device, watermarking techniques in place at the receiving compliant device will not be effective in this case. Hence, we need to provide mechanisms for identifying and revoking circumvented devices that engage in illegal distribution of protected content.

We will discuss how our system counters these new threats in Section 7.2.

5 New Content Distribution Scheme

The content distribution process can be split into two parts, first where a provider (P) distributes the content and the associated license to a consumer (C_1) and the second where C_1s redistributes the content to another consumer (C_2).

Before proceeding any further, we introduce some notation we will use in the rest of the paper.

y_A/X_A - the public/private key pair of entity A
$[D]_{y_A}$ - data D encrypted under public key of A
$[D]_{X_A}$ - data D signed with private key of A
$[D]_K$ - data D encrypted using a symmetric key K
$h(D)$ - a collision-free hash function h applied on data D

5.1 Provider distributes content and rights to consumer

In this part, the content provider P distributes content M and associated rights R to consumer C_1.

(1) $C_1 \rightarrow P$: request content
(2) $C_1 \leftrightarrow P$: mutual authentication, [payment]
(3) $P \rightarrow C_1$: $[M]_K, [K]_{y_{C_1}}, R, \sigma, \Lambda$

The consumer C_1 starts the transaction in step (1) by requesting a particular content item. In step (2), the two parties (P's distribution server and C's device) mutually authenticate each other using their public/private key pairs, and their corresponding digital certificates. During this step, C_1 may also pay for the requested content. Our scheme can use different generic payment schemes. We discuss this in more details in Section 6.

P in step (3), then encrypts the requested content M with a unique symmetric *content-key* (K) and sends the encrypted content, the content key encrypted under C_1's public key, the rights R granted to C_1 for that content, the metadata σ associated with the content (name of the artist, and the album and song title, etc.), as well as a *content license* Λ for the content:
$\Lambda = [h(y_P, y_{C_1}, M, \sigma, R)]_{X_P}$

The purpose of Λ is to certify that C has been granted rights R with respect to content M. As we will show next, this is useful when R include the rights to redistribute M to other consumers. Λ also tightly binds the metadata to the content, preventing spoofing attacks on the content.

Rights R can be represented using authorization and access policy languages like XACML [15] or XrML [16]. A full discussion on these languages is beyond the scope of this paper.

Note that P generates a new key K for each transaction with a consumer and not an unique key per content. This prevents the scenario where the same content bought by different consumers is encrypted with the same symmetric key and hence the compromise of one single device could give access to the same content on other devices.

5.2 Consumer redistributes content

The rights R may allow C_1 to redistribute M. Another consumer C_2 can then acquire M from C_1 following the protocol described below:

(1) $C_2 \rightarrow C_1$: request content
(2) $C_2 \leftrightarrow C_1$: mutual authentication
(3) $C_1 \rightarrow C_2$: $[M]_{K'}, [K']_{y_{C_2}}, R, R', \sigma, \Lambda, \Lambda'$
(4) $C_2 \leftrightarrow C_1$: check σ, [payment]
(5) $C_2 \rightarrow C_1$: ψ

C_2 starts the transaction in step (1) by requesting a particular content item. In step (2) the two devices mutually authenticate using their public/private key pairs, and their corresponding digital certificates. If the authentication is successful, $C1$'s TPM decrypts the content and re-encrypts it with another symmetric key K'. The K' is then encrypted under C_2's public key. This re-encryption is necessary to prevent the situation where compromise of one compliant device can trigger all contents in that device to be open to attack on other devices. C_1 then sends the encrypted content, the encrypted K', the original rights R, the new rights R' granted to C_2, the original content license Λ, as well as a new license Λ', specifically customized for C_2 in step (3). The new content license Λ' has the following format: $\Lambda' = [h(y_{C_1}, y_{C_2}, M, \sigma, R')]_{X_{C_1}}$

In step (4) C_2 verifies C_1's signature on the new license Λ', and validates R and M using the original content license Λ which is signed by P (assumed to be trusted by all parties). It then checks R' to make sure it can be derived from R and σ to make sure it matches the requested content (this prevents certain content masquerading attacks which we will discuss in Section 7.2). If all these checks succeed, in step (5) C_2 approves the transaction, and sends C_1 a receipt ψ:
$\psi = [h(y_{C_1}, y_P, [M]_{K'}, \sigma, R')]_{X_{C_2}}$

ψ acts as an acknowledgment from C_2 that it has received the content M with the rights R'. Depending on the business model for redistribution, ψ may also include some additional information for payment processing.

Essentially, the two licenses Λ and Λ' form a chain and serve the same purpose Λ serves for C_1 - they prove that C_2 has been granted rights R' with respect to M. If R' include the right to redistribute, C_2 can use Λ and Λ' in the same way C_1 used Λ in the redistribution protocol.

6 Payment Issues

Our scheme can support a variety of economic models for digital content redistribution, such as:

- content redistribution without consumer-to-consumer payment; this is the simplest scenario; for an additional fee, consumers may be allowed to redistribute a limited number of copies of certain content items (e.g., consumers may want to be able to render that content in all the devices they own and capable of doing that).
- content redistribution with a consumer-to-consumer payment mechanism; for an additional fee, a producer may sell a consumer the rights to redistribute a limited number of copies of a content item. The consumer recovers this fee by selling copies of the content to other consumers. Consumers are free to decide on the payment scheme that best suits their needs (for example direct cash exchange, digi-cash, etc.).
- content redistribution with a consumer-consumer-provider payment mechanism; this is similar to the previous scenario, except that the provider is also involved in the payment protocol. For example, part of the content redistribution protocol described in Section 5.2, the party receiving content may incorporate her customer account identifier (with the provider) in the transaction receipt ψ. Upon receiving ψ, the provider will debit the content price on the buyer's account, and credit some amount to the seller's account (as a reward for reselling the content). Here, the payment mechanism is more complicated, but has the advantage that the seller must report transactions back to the provider in order to receive his perks.
- content redistribution with a consumer-consumer-provider-bank payment mechanism; this is similar to the previous scenario, except that a commercial entity (e.g. bank) is also involved in the payment protocol. In this way providers do not have to keep customer accounts for all consumers that may potentially be involved in the redistribution process.

Depending on the content type, certain economical models for redistribution are more appropriate than others. Economical models where sellers need to contact a trusted third party (the provider or a bank) in order to redeem their perks are better suited for high value content, because they limit the incentives for illegally redistributing content through circumvented devices (discussed in more detail in Section 7.2). Finally, economic models based on direct consumer-to-consumer transactions and payment are more vulnerable to the circumvented device class of attacks, but they require very low overhead (for providers), so they may be an attractive alternative for low-value content.

7 Security Mechanisms

7.1 Secure storage of contents and licenses

Since the secure storage area available in a compliant device is limited, it cannot store all the content and associated rights. Compliant devices need to use insecure storage for this purpose. However, this allows possible attacks on the confidentiality of the protected content and on the integrity of the associated rights and policy files.

A simple but effective way to tackle this issue is to store each content's license in a separate file called *Content License Files* ($CLFs$) in the insecure storage. A CLF contains a copy of the license and a secure hash of the associated content. Each CLF's hash is then stored in the secure memory.

Assuming a device with 80 GB hard disk, an average size of 5 MB for the encrypted content and 20 bytes for the hash of each CLF, about 400 KB of secure storage is needed for storing the CLF hashes of the 16,000 digital content files that the device can hold. This, we argue, is a feasible requirement.

Every time the device wants to redistribute a content, it passes the associated CLF to the trusted hardware module. The module performs a hash of the CLF and compares this value with the hash value stored in its secure storage. If they match, the module can be certain that the license file has not been tampered with. Then, it opens the file, checks the redistribution license status of the content and if redistribution is allowed, processes the redistribution request and then updates the license status of the content's CLF. The new hash of this updated CLF is then calculated and stored in the secure storage, replacing the older hash. To make this process secure, all the computations have to be performed in the secure memory to keep the unencrypted CLF secure while it is being processed.

To get an idea of the size of a CLF, assume metadata to be 512 bytes long, the SHA-1(encrypted content) is 20 bytes, the redistribution license 4 bytes for a total length of 536 bytes. This is a small enough to fit in the secure memory while being processed.

7.2 Countering new threats

As discussed in Section 4, the distribution model proposed in this paper introduces several threats that are unique to the system.

7.2.1 Content masquerading

Content masquerading occurs when C_1 redistributes a piece of content that is different from the one requested by C_2. For example, C_1 may try to redistribute a low-value (e.g.: lower bit rate MP3 file) piece of content in place of a high-value one. C_1 can perform this type of attack even without circumventing his device; this is possible because the consumer to consumer redistribution protocol involves a number of I/O interactions between consumers and their devices (e.g. C_1 selects the content item to be sent to C_2, and possibly types in the requested price). The trusted module in C_1's device cannot prevent C_1 from redistributing the wrong item, since it does not know which item C_2 has requested.

This threat is countered by having the content metadata σ securely associated with the content item during the redistribution transaction. The content redistribution protocol requires C_2 to inspect the content metadata, before accepting the transaction. The association between content and metadata is secure, since they are both provided by the trusted module in C_1's device. Assume C_2 asks C_1 for item M_1, but C_1 tries to send M_2 instead; the trusted module in C_1's device correctly associates content with metadata, so C_2 will receive the metadata for M_2; upon visual inspection C_2 will detect the mismatch and reject the transaction.

It is required that compliant devices provide output capability to inspect content metadata. This should not be a problem, since nowadays even low-end electronic devices are equipped with LCD displays.

7.2.2 Untrusted storage backup attack

The storage backup attack is made possible by the fact that both content and the associated rights are not stored directly in secure storage (due to space limitations), but instead on external (untrusted) storage. The content in the untrusted storage is secured by means of encryption and secure hashing (see discussion in previous section) using cryptographic keys stored in the trusted module. However, to prevent data loss in case of hardware failure, devices may allow backup of the untrusted storage. Now, suppose C_1 acquires a content item M with the rights to redistribute it n times. Suppose C_1 performs a storage backup before any redistribution; the backup states that C_1 can still redistribute M for n times. Once C_1 has exhausted all his redistribution rights, he may try to revert the entire untrusted storage of his compliant device to the backup copy, which states C_1 can still redistribute the content.

A general way to counter this threat is to differentiate between dynamic licenses that change with usage, and static licenses which do not change. Examples of dynamic licenses are "play 'm times", "redistribute n times", etc. The device associates a version number with these licenses, stored both on untrusted storage, and inside the trusted module; this version number is incremented whenever dynamic licenses are updated. In our system the hash of the CLF serves this purpose.

Whenever the device restores state from a backup, it checks that the CLF's hash on the backup matches the hash in the trusted module. This guarantees that dynamic licenses have not changed since the backup, and prevents the attack described earlier. If there is a mismatch between the two hash values, the dynamic rights from backup are reset to the most restrictive setting (for example if the backup states "redistribute 3 times", the new state will be "no redistribution"). Providing redundant external storage for the dynamic rights can alleviate the user-unfriendliness of this solution.

7.2.3 Circumvented device attack

Once an attacker circumvents a device, he gains control of all key material inside the trusted module of that device. This allows the attacker to modify the data stored on untrusted storage at will, for example preventing the update of dynamic rights. Once dynamic rights are not properly enforced, protected content can be redistributed without restrictions to other compliant *and uncircumvented* devices.

Clearly, this threat is more serious than uploading cracked content on compliant devices from the Internet, since in the latter case content watermarking provides some form of protection (either the watermark is identified, and a correct device will refuse to render the cracked copy, or the watermark is removed, which leads to quality loss).

Consider a circumvented device C_1 that redistributes a content item M to a compliant device C_2. Originally, C_1 has legally acquired M from a provider, with some limited redistribution rights, so C_1 has a valid content license Λ for M. As part of the redistribution protocol, C_1 can also produce a valid license Λ' for the copy of M it sends to C_2. In this way, even if M is watermarked, it will appear to C_2 as legitimate, so C_2 will have no problems rendering it. However, because dynamic rights are not properly enforced, C_1 can repeat this redistribution process an unlimited number of times, which violates the original license.

To counter this threat, we need to identify circumvented devices and revoke them. The fact that a device needs to generate a distinct license for each content copy it redistributes is key to our solution. Periodically (each time it acquires content directly from the provider), each compliant device will also report content licenses acquired from other devices. By inspecting those, the provider can count the number of copies redistributed by each device; if that number is greater than what the original license allowed for, the source device must have been circumvented, so it is revoked. Once a device is revoked, other compliant devices will refuse to accept content licenses from it. Device revocation works as described in Section 4.2 - basically revocation information is bundled with content, thus acquiring new content automatically updates revocation information.

This mechanism is not perfect. Timely acquisition of the revocation data by all the devices cannot be guaranteed by this mechanism. Furthermore, if C_2 is colluding with C_1, she may prevent her (uncircumvented) device from reporting Λ' to the provider, so that C_1 would not be detected as having redistributed one copy too many. However, this requires C_2 to never contact the provider to obtain new content, which greatly reduces her incentives for colluding with C_1. Should C_1 engage in mass illegal content redistribution, the only way to remain undetected would be to ensure the collaboration of the vast majority of her redistribution group, which is even more unlikely.

7.3 Privacy issues

The method proposed to identify tampered devices by examining the accumulated licenses raises a number of consumer privacy issues. Since licenses contain the public keys of the devices involved in a transaction, the content provider could use them to keep track of consumers' buying and selling activities. The latest developments in the area of trusted computing could provide a possible solution to these privacy issues. We assume the reader is familiar with some of the concepts introduced by the Trusted Platform Module specification [3], including *remote attestation, memory curtaining*, and *sealed storage*.

Essentially, consumer privacy rights can be enforced by having the content provider run the license double-checking server on a trusted computing platform *T*. Using the remote attestation mechanisms provided by *T*, the provider can prove to consumer devices that *T* is running a privacy-preserving software stack *S*. *S* is designed to accept content licenses and double-check them against consumer records, but never export these licenses, except for reporting abuses (e.g. a consumer redistributing more copies than he is allowed to). Furthermore, *S* protects the data received from consumer devices using the memory curtaining and sealed storage facilities provided by *T*, which guarantees that no other application except *S* can access them. Finally, *S* can be validated for correct behavior, either by a trusted third-party (e.g. consumer protection agency), or through public source code inspection.

8 Related Work

As far as we know, we are the first to propose a system that supports consumer redistribution of digital content while enforcing the DRM policies associated with protected content.

So far, online music distributors like Apple iTunes [2] have relied on the single centralised provider model to distribute content to end users.

Several papers [5, 6, 8, 9] have proposed schemes to distribute protected content within so-called "personal private networks" or "domains" consisting of small number of compliant devices owned by individual consumers. xCP protocol [8] proposed a similar model for access to copy-protected content within a home network among various devices like audio/video players, set-top boxes, PCs etc.

Open Mobile Alliance's DRM architecture [1] supports a distribution process called "Super Distribution" by which protected content can be copied to a compliant device outside the domain. However the "Rights Object" needed to use the content still has to be purchased from the centralised Rights Issuer.

None of these works support the ability to let one consumer redistribute content as well as appropriate rights to another.

9 Conclusions and Future Work

DRM systems are used by digital content providers to restrict the ways the consumer use the content. In this paper we proposed a scheme to support the process of redistribution of digital content while preserving the DRM policies specified by the content provider. We explained in detail the requirements of the consumer device and the steps involved in the process of content redistribution. We analysed the security issues associated with our model and described schemes to counter them.

Currently we are working on adding a more robust and privacy preserving mechanism to identify tampered devices. We also plan to develop specific payment schemes to integrate with the redistribution process presented in this paper.

Acknowledgement

We thank the members of Security Group at VU for their valuable suggestions and comments.

References

[1] Open Mobile Alliance, *DRM Architecture Candidate Version 2.0*, July 2004, http://www.openmobilealliance.org

[2] Apple iTunes, Apple Computers, Inc., http://www.apple.com/itunes/

[3] Trusted Computing Group, *Trusted Computing Platform Alliance Main Specification*, February 2002, Version 1.1b, http://www.trustedcomputinggroup.org

[4] MC9328MX21 Applications Processor, Freescale Semiconductor, Inc., http://www.tinyurl.com/65cax

[5] Smartright technical white paper, http://www.smartright.org/images/SMR/content/SmartRight_tech_whitepaper_jan 28.pdf, Jan. 2003

[6] S.A.F.A. van den Heuvel, W. Jonker, F.L.A.J. Kamperman, and P.J. Lenoir., *Secure Content Management in Authorized Domains*, Proc. IBC 2002, pages 467-474, Sept. 2002.

[7] Amos Fiat and Moni Naor, *Broadcast Encryption*, Advances in Cryptology - CRYPTO '93, pages 480-491, Aug. 1993.

[8] Internation Business Machines Corporation, *xCP Cluster Protocol DVB-CPT-716*, http://www.almaden.ibm.com/software/ds/ContentAssurance/ papers/xCP_DVB.pdf, October 2001.

[9] Bogdan C. Popescu, Bruno Crispo, Frank L.A.J. Kamperman, Andrew S. Tanenbaum, *A DRM Security Architecture for Home Networks*, 4th ACM Workshop on Digital Rights Management, October 2004.

[10] R. Anderson and M. Kuhn.,*Tamper Resistance - a Cautionary Note*, In Proc. 2nd Usenix Workshop on Electronic Commerce, pages 1-11, Nov. 1996.

[11] Huang, Andrew. Keeping Secrets in Hardware Presentation to CHES2002, August 12 - 15, 2002. URL: http://ece.gmu.edu/crypto/ches02/talks_files/Huang.pdf

[12] KaZaA, http://www.kazaa.com/

[13] I. J. Cox, J. Killian, F. T. Leighton and T. Shamoon, *Secure Spread Spectrum Watermarking for Multimedia*, IEEE Trans. Image Proc., Vol. 6, No. 12, pp. 1673–1687, Dec. 1997.

[14] B. Chor, A. Fiat and M. Naor, *Tracing Traitors*, Proc. Advances in Cryptology, Crypto '94, Springr-Verlag LNCS 839 (1994), 257-270.

[15] eXtensible Access Control Markup Langiage (XACML), http://www.oasis-open.org/committees/xacml

[16] XrML: eXtensible rights Markup Language, http://www.xrml.org