Published in *1st International Workshop on Multimedia, Information Privacy and Intelligent Computing Systems (MPIS 2008)*, Proceedings of APSCC 2008, pp. 1344–1351, IEEE Computer Society, 2008.

LAUNDERING AND REPACKAGING OF MULTIMEDIA CONTENT IN CONTENT DISTRIBUTION SYSTEMS

Alain Durand, Marc Joye, Mohamed Karroumi

Thomson R&D France
Technology Group, Corporate Research, Security Laboratory
1 avenue de Belle Fontaine, 35576 Cesson-Sévigné Cedex, France

ABSTRACT

Content distribution systems enable the secure distribution of multimedia content. At the same time, and sometimes more importantly, they should also disable the illegal [re-]distribution of multimedia content. This paper identifies different types of attacks on current systems: *laundering attacks* and *repackaging attacks*. The attacks are described generically so that they may apply to most systems used for distributing protected content to set of users. First hints to prevent such attacks are also discussed.

Keywords. Content distribution systems, digital rights management, laundering attacks, repackaging attacks.

1. INTRODUCTION

Today, thanks to the availability of high bandwidth connections and efficient data compression technologies, multimedia content can be transmitted with high quality. This generates new businesses and distribution models. Unfortunately, this also gives rise to a high level of piracy. For example, digital content (music, video and software game) can be pirated by users who make copies available to others or by insiders during production processes. Content protection technologies are intended to protect against content piracy. Their proper design and correct implementation are of central importance for the entertainment industry because the incurred losses can be huge.

An investigation on prevailing content protection technologies shows that there are many variants. However, what is common in each content distribution system is the basic security functionality. Before being written to a physical support or otherwise transmitted, content is usually encrypted by content providers. As a consequence, a required step prior to *illegal distribution* (e.g., unauthorized distribution of content in peer-to-peer file sharing like Kazaa or E-mule or on physical media like DVDs) or *illegal consumption* (e.g., unauthorized use of a content legally purchased) is extracting content from its protected form. Most of the time, this means recovering the content in clear. While existing content distribution systems address this issue, they do little or nothing to prevent a pirate or an end-user to divert them from their primary purpose.

Another form of piracy is that of illegal distribution involving bit-to-bit copies or device cloning. There exist techniques (e.g., [5, 6]) based on a unique physical ID[1] making useless a number of duplication methods. These techniques have in common that this ID is located in a non-writable area. The unique physical ID enables to securely bind the multimedia content to the player or physical support so that it can only be played using the corresponding player or the corresponding support. Other concurrent techniques are based on authorization certificates issued by content distributors. The certificate is then linked to a unique user ID (e.g., [2, 8]).

We demonstrate that current implementations may be used as a tool for distributing information in a variety of crimes including child pornography (content laundering) or violations of copyrights (content repackaging). Unauthorized users can exploit distribution systems because the security chain is not addressed as a whole. It is our purpose to study forms of piracy that go beyond merely getting and distributing the content in the clear. To the best of our knowledge, these attacks have not yet been described and therefore may concern most content protection systems.

In this paper, we mainly focus on illegal distribution. We describe different scenarios and use cases wherein our attacks may help a pirate to illegally distribute multimedia content. We also present some simple measures to prevent these attacks. All these measures require to perform a cryptographic computation on the whole content, which might be somewhat expensive. Of course, it would be more desirable to avoid this, but it is an open problem whether this is even feasible at all.

Outline of the paper

The rest of this paper is organized as follows. In the next section, we present a generic architecture for distributing encrypted multimedia content. Section 3 is the core of our paper.

[1]The unique ID is encoded in or computed from the physical structure of a hardware material, generally the user's player or the physical support containing the multimedia content.

We show that current distribution systems may be subject to further attacks than the disclosure of the content in clear. In Section 4, we present various methods for preventing these attacks. Finally, we conclude in Section 5.

2. DISTRIBUTION OF ENCRYPTED MULTIMEDIA CONTENT

In this section, we describe a typical content distribution system. It is subject to numerous variations and optimizations. This is not our focus. Our aim is to introduce the key components and actors involved in the whole process: from the content owner to the content player. The architecture we present is *generic* and can accommodate a large variety of practical systems.

The life cycle of multimedia content consists of four main phases:

Content creation and edition This phase includes for example audio and graphical effects, animation, subtitles, as part of the production process before presentation of the content in its final form. Optionally, watermarks may be inserted to trace the content.

Content packaging This phase encompasses encryption and signature of content to protect it from misuse or unauthorized redistribution.

Content distribution Once content has been prepared and protected, it has to be delivered to end-users. This can be done through a variety of existing channels like satellite, Internet or on physical media (e.g., DVD or cartridge).

Content consumption This phase provides mechanisms for the end-user to access and render content on his player. It is generally carried out within a set-top box, a game console or by a dedicated software on his computer.

We do not consider the content creation phase, which is out of the scope of this paper, and call *multimedia content* its output.

2.1. Content distribution

We let M denote a multimedia content. The *content distributor* assigns a unique identifier # to M and associates with it a header hdr (containing for example the name of the content, its identifier, its encoding format, . . .).

Before being distributed, multimedia content M needs to be "prepared" or, more exactly, *protected*. This is the role of the *content packager*. This involves the encryption of M under a *content key*, say $\mathsf{K_c}$. Without loss of generality, we assume that M is encrypted using a symmetric encryption algorithm, $\mathrm{Enc}()$, to form the encrypted content $C = \mathrm{Enc}(\mathsf{K_c}; M)$. Content key $\mathsf{K_c}$ is randomly generated. The content packager may also add information to the header in order to retrieve the key material (e.g., a URL). The protected content and the corresponding header, namely $[\mathsf{hdr}, C]$, are then made available on a *content server*.

The key material necessary to decrypt the content is made available on a *key server*. Its management and storage are described in § 2.3. But before that, we need to introduce the licensing authority.

2.2. Licensing authority

A multimedia content packaged by a content packager cannot be read on any player. The player must conform to certain requirements enacted by a *licensing authority*. It serves as a trusted third party. The licensing authority possesses a matching pair of public/private keys $(\mathsf{vk}_{\mathrm{la}}, \mathsf{sk}_{\mathrm{la}})$. Private key $\mathsf{sk}_{\mathrm{la}}$ is used to digitally sign while public key $\mathsf{vk}_{\mathrm{la}}$ is used to check the validity of a signature issued by the licensing authority.

If a player manufacturer wishes to produce and sell players that can read content, it must request player keys to the licensing authority. If licensing authority assesses the player to be compliant, the player manufacturer receives one or several player keys. A given player key may be unique to one player or shared among a family of same players. We let p_{id} denote a player representative. In more detail, for each p_{id}, the licensing authority generates a pair of encryption/decryption keys, say $(\mathsf{ek}_{p_{\mathrm{id}}}, \mathsf{dk}_{p_{\mathrm{id}}})$, and provides the player manufacturer with $\mathsf{dk}_{p_{\mathrm{id}}}$. Decryption key $\mathsf{dk}_{p_{\mathrm{id}}}$ and verification key $\mathsf{vk}_{\mathrm{la}}$ are stored in player p_{id}.

The licensing authority maintains a list $\mathfrak{L}$ containing all $(p_{\mathrm{id}}, \mathsf{ek}_{p_{\mathrm{id}}})$. If, at a latter time, a player is no longer compliant (e.g., following a security flaw), the licensing authority removes from list $\mathfrak{L}$ the corresponding entry.

2.3. Key material

The encryption of multimedia content M is (at least) a two-level process. The content packager first encrypt M under a random content key $\mathsf{K_c}$ to form C. Content key $\mathsf{K_c}$ is also encrypted to form what is called a *distribution key*, say $\mathsf{K_d}$. This second level of key enables the distribution of a same protected content to different players.

The encryption of $\mathsf{K_c}$ proceeds as follows. The content packager sends to the licensing authority a set of players, $\mathfrak{P} = \{p_{\mathrm{id}}\}$, the multimedia content is intended for. Next, the licensing authority checks that the set of intended players is valid, namely that each $p_{\mathrm{id}} \in \mathfrak{L}$. If so, the licensing authority issues the distribution key

$$\mathsf{K_d} = \{\mathcal{E}(\mathsf{ek}_{p_{\mathrm{id}}}; \mathsf{K_c})\}_{p_{\mathrm{id}} \in \mathfrak{P}} \,,$$

that is, the encryption of $\mathsf{K_c}$ under encryption key $\mathsf{ek}_{p_{\mathrm{id}}}$ using encryption algorithm $\mathcal{E}()$, for each $p_{\mathrm{id}} \in \mathfrak{P}$.[2]

[2]When the multimedia content is intended to a huge number (several mil-

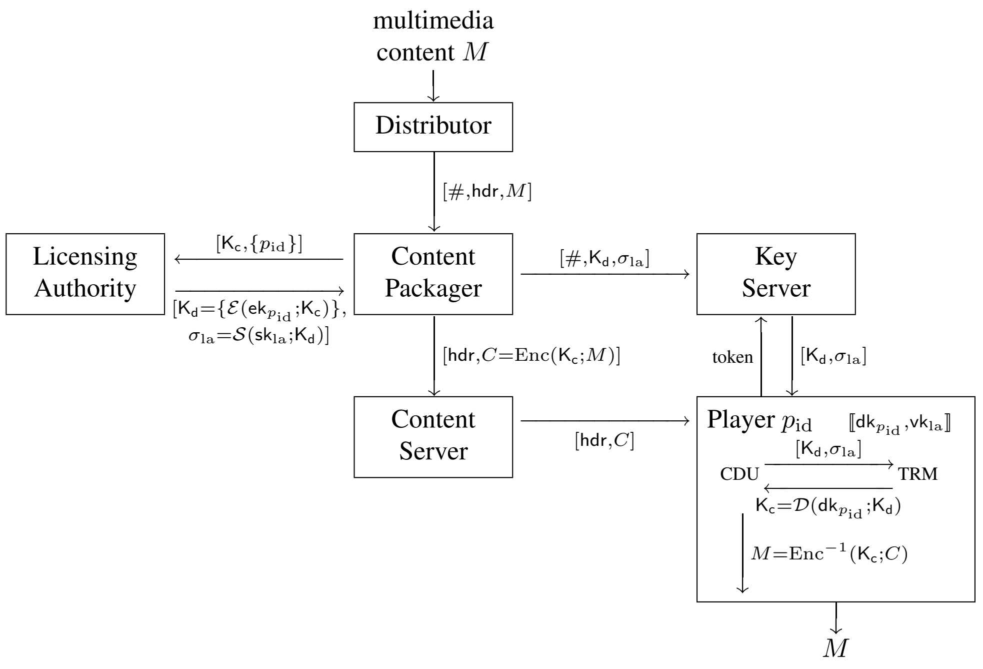

multimedia content M
Distributor
$[\#,\mathsf{hdr},M]$
Licensing Authority
$[\mathsf{K_c},\{p_{\mathrm{id}}\}]$
$[\mathsf{K_d}=\{\mathcal{E}(\mathsf{ek}_{p_{\mathrm{id}}};\mathsf{K_c})\},$
$\sigma_{\mathrm{la}}=\mathcal{S}(\mathsf{sk}_{\mathrm{la}};\mathsf{K_d})]$
Content Packager
$[\#,\mathsf{K_d},\sigma_{\mathrm{la}}]$
Key Server
$[\mathsf{hdr},C=\mathrm{Enc}(\mathsf{K_c};M)]$
token
$[\mathsf{K_d},\sigma_{\mathrm{la}}]$
Content Server
$[\mathsf{hdr},C]$
Player p_{id}
$[\![\mathsf{dk}_{p_{\mathrm{id}}},\mathsf{vk}_{\mathrm{la}}]\!]$
$[\mathsf{K_d},\sigma_{\mathrm{la}}]$
CDU
TRM
$\mathsf{K_c}=\mathcal{D}(\mathsf{dk}_{p_{\mathrm{id}}};\mathsf{K_d})$
$M=\mathrm{Enc}^{-1}(\mathsf{K_c};C)$
M

Fig. 1. Typical multimedia distribution architecture

The licensing authority also signs the distribution key, $\mathsf{K_d}$, using signing algorithm $\mathcal{S}()$ with its private signing key $\mathsf{sk}_{\mathrm{la}}$ to get the signature $\sigma_{\mathrm{la}} = \mathcal{S}(\mathsf{sk}_{\mathrm{la}}; \mathsf{K_d})$. $\mathsf{K_d}$ and σ_{la} are sent to the content packager. These two quantities together with content identifier $\#$ are stored on a *key server*.

2.4. Content play back

We consider two types of distribution channels.

Electronic distribution If a user wants to play a protected content C (with header hdr and identifier $\#$) retrieved from the content server, he first needs to purchase a token for that content to some retailer. This token is then traded against the distribution key $\mathsf{K_d}$ and its signature σ_{la}.

Physical media In this case, there is no distinction between the content server and the key server. Both the protected content C and the corresponding key material, $\mathsf{K_d}$ and σ_{la}, are stored on a physical media (e.g., a DVD or a cartridge). The user simply purchases a copy of the physical media.

It remains to explain how a multimedia content is played from $[C, \mathsf{K_d}, \sigma_{\mathrm{la}}]$. A player comprises two main components: a *content decryption unit* (CDU) and a *tamper-resistant module* (TRM). The tamper-resistant module embeds the player decryption key $\mathsf{dk}_{p_{\mathrm{id}}}$ and the public key of the licensing authority, $\mathsf{vk}_{\mathrm{la}}$. On input $[\mathsf{K_d}, \sigma_{\mathrm{la}}]$, the tamper-resistant module first verifies signature using verification key $\mathsf{vk}_{\mathrm{la}}$. If the verification succeeds, it computes the content key as $\mathsf{K_c} = \mathcal{D}(\mathsf{dk}_{p_{\mathrm{id}}}; \mathsf{K_d})$ using decryption key $\mathsf{dk}_{p_{\mathrm{id}}}$. From $\mathsf{K_c}$ and C, the content decryption unit then recovers (plain) multimedia content $M = \mathrm{Enc}^{-1}(\mathsf{K_c}; C)$, which is played.

3. CONTENT LAUNDERING AND REPACKAGING

In this section, we present two classes of attacks where an attacker can produce protected multimedia content without interacting with the licensing authority that is supposed to provide with the necessary key material. These attacks open the door to illegal distribution of (pirated) multimedia content that cannot be distinguished from genuine one.

In the following, we distinguish three categories of players:

Compliant players Devices or softwares that conform to the system specifications, that have keys obtained from the licensing authority and that are present in list $\mathfrak{L}$.

Conformant but non-compliant players Devices or softwares that conform the system specifications, that have keys (obtained or not from the licensing authority) but that are not present in list $\mathfrak{L}$. This includes implementations that conform to the specifications but do not qualify to the licensing authority validation process or revoked devices.

lions) of players, the generation of $\mathsf{K_d}$ can be optimized using broadcast encryption techniques [3, 10]. But to keep the presentation simple, we see $\mathsf{K_d}$ as the separate encryption of $\mathsf{K_c}$ under the key $\mathsf{ek}_{p_{\mathrm{id}}}$ of each player in $\mathfrak{P}$.

Non-conformant players Devices or softwares that do not conform to the system specifications and are thus unable to play protected multimedia content.

Remark: A player that can play protected content, whether it is compliant or not, will be referred to as a *conformant* player. This includes the two first categories.

We make the assumption that an efficient revocation mechanism is in place since it supposes that compromised players are immediately removed from the list $\mathfrak{L}$. Furthermore, as a first approach, we suppose that conformant players are unable to handle clear content (i.e., they can only consume protected multimedia content). This approach corresponds for instance to the following cases:

- Secure peer-to-peer content distribution systems: Their main purpose is to protect transmissions between peers over an unsecured network. Nevertheless, a peer computer also serves as a network content server and does not necessarily prevent sharing of pirated content. To mitigate piracy in such systems it is important to disable the distribution of unprotected content. So, some of them authorize only the exchange of protected multimedia content (that comes from a trusted source).
- Game consoles: A large number of platforms are not user-programmable since they can only play protected games. However, some home-brew[3] games are developed to be used with emulators that do not necessarily require games to be protected.

We refer to § 3.3 for the case of conformant players able to also deal with clear content.

In the next two sections, we present content laundering and content repackaging attacks. For both type of attacks, we make the assumption that the attacker knows one or several content keys $\mathsf{K_c}$. The relevance of this assumption is discussed in § 3.4.

3.1. Content laundering

Content laundering attacks consist in transforming clear content into protected content so that it can be played on conformant players. Content source is not specified but generally illegal like a peer-to-peer network for example. Here are some example scenarios:

- Pirates may be willing to perform the attack because it augments the interest of their offer. For instance, a mafia organization may "hide" their pirated DVDs to go through customs controls. Clear content on a DVD may appear suspect contrarily to protected one.

[3]Term applied to video games that are produced by users for proprietary game platforms.

- In certain secure peer-to-peer distribution systems (as those mentioned previously), peers cannot act as content sources and can only exchange encrypted multimedia content. Using laundering attacks, a secure peer-to-peer may be hijacked by criminals or terrorists. This would permit them to introduce and exchange video or terrorist messages. In this case, detection will be difficult because the encryption will hide suspected videos.
- A user develops its own home-brew games and wants to play it on a compliant device because it offers a better user experience.
- A user may want to enjoy other games downloaded from the Internet because he finds a widest choice of games (even at low price), which cannot be found on legal cartridges. It could be games of older console generations for example.

3.1.1. *Attack description*

Here is a sketch of the attack. A pirate gets a copy of pirated content P that he wants to launder. Per our initial assumption, the pirate knows as well some content key $\mathsf{K_c}$ and associated hdr and $\#$ corresponding to a genuine content.

The pirate, using $\mathsf{K_c}$, encrypts P and obtains the laundered content $L = \mathrm{Enc}(\mathsf{K_c}; P)$. The pirate then distributes L together with the header hdr and identifier $\#$.

Consider some end-user possessing a conformant player. From header hdr, this user can obtain the corresponding distribution key $\mathsf{K_d}$ and accompanying signature σ_{la} from the key server. The signature verification being correct, the player will decrypt $\mathsf{K_d}$ to obtain $\mathsf{K_c}$; and so it will decrypt L to get P. There is actually no means for the player to detect that it is decrypting a laundered content.

3.1.2. *Discussion*

The attack can be performed as soon as the pirate knows some content key $\mathsf{K_c}$ and associated hdr and $\#$. The same content key can then be applied to any pirated content.

A possible countermeasure to prevent laundering attacks would be to sign the content. Unless getting content with its signature (e.g., from an insider), the pirate cannot dispose of the signature of the content. As a consequence, any pirated content obtained from the analog hole (e.g., cam-corded content) cannot be laundered.

Revocation could be seen as a means to prevent the use of the corrupted key material ($\mathsf{K_c}$, hdr and $\#$). It is however very difficult to make use of that mechanism since, in most of cases, revocation will affect as well the genuine content to which the corrupted key material originally applied, something which is not acceptable.

3.2. Content repackaging

Content repackaging attack allows an attacker to re-encrypt multimedia content using a different content key. Repackaged multimedia content can be redistributed and played back by compliant players.

Repackaging attacks allows a pirate to propose a more attractive offer to the end-user than the original ones. Here are some example scenarios where the attack is useful (for the pirate):

- In geographical black-out scenarios: For some sport games, the region where the match is played is blacked out to encourage people to go the match. Performing the attack, the pirate may propose the match to people from the blacked-out region.
- In situations where a number of devices have been revoked, the pirate may repackage the content using keys of revoked devices to allow revoked devices to also access to the content.
- A pirate can repackage games or movies in order to circumvent zoning control: he will be able to make available content in geographic regions that would have had to wait several months else.
- A pirate can repackage several pieces of content on the same support. He buys for example three protected movies that he recovers in the clear form. Next, he repackages them on the same support that he can sell to millions of users at an attractive price. He can for instance sell three movies at the price of one.

3.2.1. Attack description

To perform the attack, the pirate obtains from a legal distribution channel the content M_a he wants to redistribute. M_a is distributed in encrypted form C_a and distributed together with header hdr_a and identifier $\#_a$.

The attacker has then to recover the corresponding content key $\mathsf{K}_{\mathsf{c}_\mathsf{a}}$ (the feasibility of this step is discussed in § 3.4). Using $\mathsf{K}_{\mathsf{c}_\mathsf{a}}$, he is able to recover the clear content M_a.

Then, using the assumption he knows some other content key K_c and corresponding headers hdr and identifier $\#$, he re-encrypts M_a using K_c to obtain repackaged content R. He may then redistribute R together with hdr and $\#$, as for the laundering attack.

Conformant and/or compliant players will be able to access R in the same way as for laundered content.

3.2.2. Discussion

Contrarily to laundering attacks, the previous attacks work perfectly on signed content. As the attacker gets the content from a legal distribution channel, he obtains content signature together with the content. He can thus repackage it and redistribute it.

Compared with a laundering attack, the attack requires an additional step that shall be done for each repackaged content: the pirate has first to recover the original content key.

Finally, this attack points out that the content key K_c used to repackage the content shall preferably be associated with a header authorizing all users of the system to access the content. This makes the pirate content more attractive. This remark applies as well to laundering attacks.

3.3. Dealing with clear content

In this section, we relax the hypothesis that conformant players accept only encrypted content. This model is particularly adapted to video content where end-users expect to use their player not only for content they buy but also for their own generated content (i.e., personal creations or events like wedding, birth, etc.) or their unprotected copyrighted content (e.g., free-to-air content or content under common creative license).

In such a case, the incentives for performing these attacks may seem unclear as the end-users can already enjoy clear pirated content on their conformant players. Indeed, as clear content can be played on conformant players, it may seem meaningless for the end-user to launder content and mafia organizations may prefer to distribute clear content. However, these organizations may still launder their content to escape e.g. customs controls as was illustrated in § 3.1.

A general trend in this model is to watermark content in order to distinguish user-generated content from clear pirated content (see e.g. [9, 1]). A compliant player, when requested to play clear content, will first try to detect whether the content is watermarked or not:

- If the content is not watermarked, it is supposed to be user-generated content and the conformant player renders the content;
- Otherwise, the content is a clear pirated content and the player refuses to play it.

If this solution is implemented, both end-users and mafia organizations have a true incentive to launder content.

The addition of watermarking does not affect at all the ability to mount redistribution or laundering attacks, since watermarking is only checked for unencrypted content. Verifying watermarks also in the encrypted mode (i.e., checking that encrypted content is watermarked) would only prevent to launder user-generated content (which is useless in this model). Laundering pirated content or redistributing content as previously described would still remain possible.

3.4. Practicability

Both attacks suppose the pirate has been able to recover at least one content encryption key $\mathsf{K_c}$. Repackaging attacks require even to recover such a key for each new content to repackage.

Some DRM systems [7] propose an architecture to protect the content from attacks during the content distribution. The architecture ensures that only a user device can decrypt the digital content. This avoids the recovery of $\mathsf{K_c}$ and thus repackaging attacks to be done by intermediate actors in the distribution process. However, user devices still need $\mathsf{K_c}$ for content consumption.

Even if a tamper-resistant module is used to protect infrastructure keys and data, it cannot be used to manipulate the content itself because of its limited capacity both in terms of memory and of computational power. Content encryption keys are then manipulated in standard memory and are thus more sensitive to implementation attacks.

In broadcast systems (e.g., broadcast television or distribution on DVD-ROMs), a given content is encrypted under the same $\mathsf{K_c}$ even if it is distributed to many players. In that case, the key is more exposed and disposing of one player with a "less-secure" implementation is enough to recover $\mathsf{K_c}$. Other means can also be used to obtain the content encryption key; for example, from insiders, i.e., people working within the company that owns the multimedia content, using social engineering techniques.

It is assumed that the attacker is only able to get access to content encryption keys (cf. Section 3). This assumption is intentionally chosen to be weak as having $\mathsf{K_c}$ is enough for laundering and repackaging attacks. A most powerful attacker would obtain keys at a higher level in the hierarchy (e.g., a player key) that anyway allows the re-computation of the content encryption key.

Finally, attacks feasibility does not depend on the encryption algorithm used to protect $\mathsf{K_c}$. More particularly, if the $\mathsf{K_c}$ is asymmetrically encrypted, the attacks still work. The pirate has still to get one $\mathsf{K_c}$ and replicate the corresponding identifier and distribution key $\mathsf{K_d}$.

4. CONSTRUCTIONS

Whether user-generated content is considered or not, content laundering (in § 3.1) and repackaging attacks (in § 3.2) are possible because protection system designers presume that encrypted content cannot be pirated content. As explained in the previous section, this assumption is not always satisfied.

The ability to perform these attacks comes from the fact that there is no secure link between the encrypted multimedia data and the key material: Any key $\mathsf{K_c}$ can be used to encrypt a given piece of content. Therefore, a content key can be applied to a pirated content or to a content to which it was not intended and may violate the initial content copyright.

We propose here different tracks to remedy to this problem, all the proposed solutions being based on the creation of a secure link between the content key and the content it is protecting. If this secure link exists, none of the aforementioned attacks will work anymore.

All solutions imply to sign large amounts of data. This requires highly efficient signing processes; we refer the reader to [4] for several methods addressing this issue. Furthermore, we consider signature schemes with appendix; in the following, σ (or Σ) will denote the signature only. Surely, methods that do not require to sign a large amount of data are desirable but it is yet an open problem to know whether such methods exist.

4.1. Signing content and keys

The natural solution that comes in mind is to sign multimedia content along with all the keys involved in the content protection. Namely, to sign the content concatenated with all the keys in the key hierarchy.

When receiving M, the content packager generates $\mathsf{K_c}$ and requests the licensing authority a distribution key $\mathsf{K_d}$ for a given set of players $\mathfrak{P} = \{p_{\mathrm{id}}\}$. Upon receiving $\mathsf{K_d}$, content packager signs it together with the content M and the content key, $\Sigma_{\mathrm{cp}} = \mathrm{Sign}(\mathsf{sk}_{\mathrm{cp}}; M \| \mathsf{K_c} \| \mathsf{K_d})$. Σ_{cp} is then made available on the key server together with auxiliary data $[\#, \mathsf{K_d}, \mathsf{vk}_{\mathrm{cp}}, \sigma_{\mathrm{la}}]$. At play-back time, each player in $\mathfrak{P} = \{p_{\mathrm{id}}\}$ verifies signature Σ_{cp}. The multimedia content is played if and only if the verification succeeds.

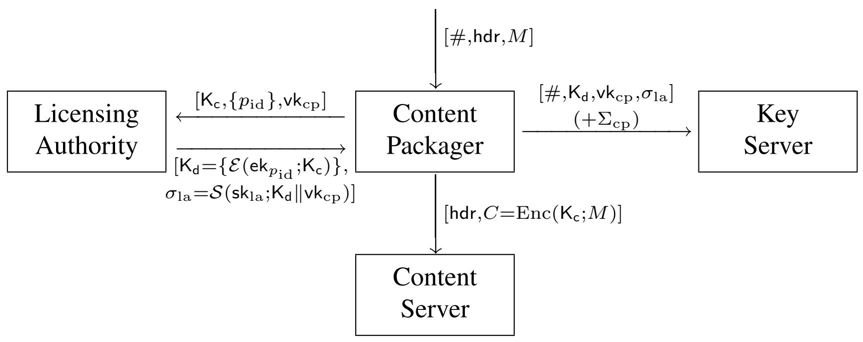
The secure binding is effective as $\mathsf{K_d}$ cannot be used to distribute another content M' (laundering is countered) and another key material $\mathsf{K_d}'$ associated to content $[\mathsf{hdr}', C']$ cannot be used to distribute M (repackaging is countered). Here only two keys were added in the signature because there are only two steps of encryption. If more encryption steps are involved, which is very likely (as in [5, 6]), all the keys must be added to content before signing.

4.2. Signing encrypted content

The idea here is to use the signature on the encrypted version of the multimedia content instead of the one on the clear version. When receiving M, the content packager first encrypts M, $C = \mathrm{Enc}(\mathsf{K_c}; M)$, and signs C with its private key, $\Sigma_{\mathrm{cp}} = \mathrm{Sign}(\mathsf{sk}_{\mathrm{cp}}; C)$. The licensing authority next issues the key material together with Σ_{cp} in order to distribute the data to a given set of players $\mathfrak{P} = \{p_{\mathrm{id}}\}$.

At play-back time, the content decryption unit verifies signature Σ_{cp}. If this signature is valid, the content decryption unit recovers multimedia content that is played.

On the contrary, if the content signature Σ_{cp} is invalid the multimedia content is not played. Two scenarios are then possible: either $\mathsf{K_c}$ is not authentic, either content M has been


$[\#,\mathsf{hdr},M]$
Licensing Authority
$[\mathsf{K_c},\{p_{\mathrm{id}}\},\mathsf{vk}_{\mathrm{cp}}]$
$[\mathsf{K_d}=\{\mathcal{E}(\mathsf{ek}_{p_{\mathrm{id}}};\mathsf{K_c})\}, \sigma_{\mathrm{la}}=\mathcal{S}(\mathsf{sk}_{\mathrm{la}};\mathsf{K_d}\|\mathsf{vk}_{\mathrm{cp}})]$
Content Packager
$[\#,\mathsf{K_d},\mathsf{vk}_{\mathrm{cp}},\sigma_{\mathrm{la}}]$ $(+\Sigma_{\mathrm{cp}})$
Key Server
$[\mathsf{hdr},C=\mathrm{Enc}(\mathsf{K_c};M)]$
Content Server

§ 4.1 – Signing content and keys: $\Sigma_{\mathrm{cp}} = \mathrm{Sign}(\mathsf{sk}_{\mathrm{cp}}; M\|\mathsf{K_c}\|\mathsf{K_d})$;

§ 4.2 – Signing encrypted content: $\Sigma_{\mathrm{cp}} = \mathrm{Sign}(\mathsf{sk}_{\mathrm{cp}}; C)$;

§ 4.3 – Encrypting using signature: $\mathsf{K_c} = \mathrm{Sign}(\mathsf{sk}_{\mathrm{cp}}; M)$.

Fig. 2. Illustration of some solutions

changed. If $\mathsf{K_c}$ is not authentic but $\mathsf{K_d}$ is, then $\mathsf{K_d}$ has been retrieved from another protected multimedia content $[\mathsf{hdr}', C']$ and used to protect this new content M. We are in front of a repackaging attack (cf. § 3.2). If content is not "authentic" then we are in front of a laundering attack (cf. § 3.1). Consequently, the binding is effective using this solution. Again, if there are more than two encryption levels, then the header signature must be applied to the whole key hierarchy and not only to $\mathsf{K_d}$.

4.3. Encrypting using signature

Yet another solution is, instead of generating $\mathsf{K_c}$ at random, to make it depend on the signature of the multimedia content. When receiving M, the content packager first signs it with private key $\mathsf{sk}_{\mathrm{cp}}$, $\Sigma_{\mathrm{cp}} = \mathcal{S}(\mathsf{sk}_{\mathrm{cp}}; M)$. Next, the content packager chooses $\mathsf{K_c}$ as the content signature value, $\mathsf{K_c} = \Sigma_{\mathrm{cp}}$, and encrypts M under it, $C = \mathrm{Enc}(\mathsf{K_c}; M)$. The distributed protected multimedia content becomes $[\mathsf{hdr}, C, \#, \mathsf{K_d}, \mathsf{vk}_{\mathrm{cp}}, \sigma_{\mathrm{la}}]$. There is no need to make signature Σ_{cp} available as it is already present in encrypted form inside $\mathsf{K_d}$.

When a player receives the protected content, it verifies both signatures Σ_{cp} and σ_{la}. If they are valid, it computes the content key as $\mathsf{K_c} = \mathcal{D}(\mathsf{dk}_{p_{\mathrm{id}}}; \mathsf{K_d})$ using decryption key $\mathsf{dk}_{p_{\mathrm{id}}}$. From $\mathsf{K_c}$ and C, it recovers then multimedia content $M = \mathrm{Enc}^{-1}(\mathsf{K_c}; C)$. Finally, it verifies content signature Σ_{cp} $(= \mathsf{K_c})$ using verification key $\mathsf{vk}_{\mathrm{cp}}$. The multimedia content is played if and only if the verification succeeds.

As before, if the last authentication fails then most probably a laundering or repackaging attack occurred. If there are more than two encryption levels, then the header signature must be applied to the whole key hierarchy and not only to $\mathsf{K_d}$.

5. CONCLUDING REMARKS

Current content distribution systems are mainly designed to prevent unauthorized users to get access to protected multimedia content. We discovered a number of situations where it is important to consider attacks beyond this paradigm. We observed that protected content does not necessarily come from an authorized source. We pointed out that, if not properly implemented or used, existing systems may be vulnerable to laundering attacks or to repackaging attacks. In both cases, the goal of the attacker is to benefit from the existing infrastructure to illegally distribute content. This was illustrated through various examples and use cases. Finally, we presented and discussed several modifications to current content distribution systems for preventing such attacks.

While simple, the constructions we presented are resource demanding and may not be fully satisfactory for all distribution models. We invite the research community and the multimedia industry to devise better solutions addressing the issues raised by our extended attack scenarios.

Acknowledgments

We are grateful to Yves Maetz and Nicolas Prigent for useful comments.

6. REFERENCES

[1] J. A. Bloom, I. J. Cox, T. Kalker, J.-P. M. G. Linnartz, M. L. Miller, and C. B. S. Traw. Copy protection for DVD video. *Proceedings of the IEEE*, 87(7):1269–1270, 1999.

[2] C. Conrado, F. Kamperman, G. J. Schrijen, and W. Jonker. Privacy in an identity-based DRM system. In *14th International Workshop on Database and Expert Systems Applications (DEXA '03)*, pages 389–395. IEEE Computer Society, 2003.

[3] A. Fiat and M. Naor. Broadcast encryption. In D. R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Sciences*, pages 480–491. Springer-Verlag, 1994.

[4] R. Gennaro and P. Rohatgi. How to sign digital streams. *Information and Computation*, 165(1):110–116, 2001. Earlier version in Proc. of CRYPTO '97.

[5] Intel Corporation, International Business Machines Corporation, Matsushita Electric Industrial Co., Ltd., and Toshiba Corporation. *Content Protection for Recordable Media: Introduction and Common Cryptographic Elements*, 2000. Revision 0.94.

[6] Intel Corporation, International Business Machines Corporation, Matsushita Electric Industrial Co., Ltd., Toshiba Corporation, Microsoft Corporation, Sony Corporation, The Walt Disney Company, and Warner Bros. *Advanced Access Content System (AACS): Introduction and Common Cryptographic Elements*, Feb. 17, 2006. Revision 0.91.

[7] J. Kim, Y. Jeong, K. Yoon, and J. Ryou. A trustworthy end-to-end key management scheme for digital rights management. In *14th ACM International Conference on Multimedia*, pages 635–638. ACM Press, 2006.

[8] P. Koster, F. Kamperman, P. Lenoir, and K. Vrielink. Identity-based DRM: Personal entertainment domain. In *Transactions on Data Hiding and Multimedia Security I*, volume 4300 of *Lecture Notes in Computer Science*, pages 104–122. Springer-Verlag, 2006.

[9] M. L. Miller, I. J. Cox, J.-P. M. G. Linnartz, and T. Kalker. *Digital Signal Processing for Multimedia Systems*, chapter 18, pages 461–485. Marcel Dekker Inc., 1999.

[10] D. Naor, M. Naor, and J. B. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer-Verlag, 2001.