Towards a Software Architecture for DRM

Sam Michiels, Kristof Verslype, Wouter Joosen and Bart De Decker
DistriNet Research Group, Department of Computer Science,
K.U.Leuven, Celestijnenlaan 200A, B-3001 Leuven, Belgium
{sam.michiels,kristof.verslype}@cs.kuleuven.be

ABSTRACT

The domain of digital rights management (DRM) is currently lacking a generic architecture that supports interoperability and reuse of specific DRM technologies. This lack of architectural support is a serious drawback in light of the rapid evolution of a complex domain like DRM. It is highly unlikely that a single DRM technology or standard will be able to support the diversity of devices, users, platforms, and media, or the wide variety of system requirements concerning security, flexibility, and efficiency. This paper analyses state-of-the-art DRM technologies and extracts from them high level usage scenarios according to content consumers, producers, and publishers. In addition, the key services are identified both from a functional and security perspective. Identifying key DRM services and locating them in an overall structure brings us one step closer to a software architecture for DRM. Having available a software architecture should help the DRM community in reasoning about DRM systems, and in achieving reuse and interoperability of multiple domain-specific DRM technologies and standards.

Categories and Subject Descriptors

K.5.1 [**Legal Aspects of Computing**]: Hardware/Software Protection—*Licensing, Proprietary rights*; D.2.11 [**Software Engineering**]: Software Architectures—*Domain-specific architectures*; D.2.13 [**Software Engineering**]: Reusable Software—*Domain engineering*

General Terms

Design, Security, Standardization.

Keywords

Software architecture, DRM

1. INTRODUCTION

Systems that provide digital rights management (DRM) are highly complex and extensive [17]: DRM technologies must support a diversity of devices, users, platforms, and media, and a wide variety of system requirements concerning security, flexibility, and manageability. This complexity and extensiveness poses three major challenges to DRM development. These challenges provide the context of this paper: fragmentation of individual solutions, limited reuse and interoperability of DRM systems, and lack of a domain-specific structure that supports and guides the design and implementation of DRM systems and their applications.

The first challenge relates to the fact that state-of-the-art DRM technologies are often ad-hoc, which leads to fragmented solutions and makes it very difficult to complete the global DRM picture. A complete DRM solution should provide a single platform that can support every aspect of digital rights management [12].

The second challenge, limited interoperability, is partly caused by in-house developed solutions that are incompatible with similar systems produced by other parties. Although various research groups have produced "vertically integrated" designs in which their particular set of components are specifically conceived to collaborate, their solutions are unable to interoperate with components from other groups. Given the complexity and extensiveness of DRM, interoperability between specific DRM technologies is crucial to integrate existing solutions [12].

The third challenge, lack of guiding software structure, is typical for complex software systems in evolution, and providing such a context is often a sign of growing maturity of the application domain [15]. In order to evolve towards a complete set of interoperable DRM solutions, we need a well-defined software architecture that identifies the major services and defines how they interact [12, 3].

The challenges of integrating independent system components are well-recognized and are being addressed in other application domains than DRM, such as network protocol stacks, web services, or graphical user interfaces [21]. The Internet architecture, for instance, convincingly demonstrates how a properly chosen set of guiding principles can shape the evolution of a complex system across vast changes in technology, scale, and usage [5]. The power of the Internet lies not so much in the elegance or efficiency of its individual components, but in the overall ability to encompass tremendous growth in scale and diversity as usage and technology continue to evolve.

This paper argues for a layered DRM architecture that supports DRM developers in producing complete and interoperable systems. The architecture is approached both from a functional and a security perspective. The functional

DRM'05, November 7, 2005, Alexandria, Virginia, USA.

perspective zooms in on the top layers, closest to the applications using the architecture. The security perspective focuses on the bottom layers, which offer cryptographic primitives to enforce digital rights. In other words, the cryptographic primitives at the bottom layers lay the foundation for the upper layers to build upon. Finally, the proposed architecture is validated by matching it to state-of-the-art DRM technologies.

Our research relates to the work of Jamkhedkar and Heileman [12], which presents a layered architecture for DRM. This architecture is inspired by the TCP/IP architecture, which structures various combinations of highly complex network protocols as a stack of protocol layers [5]. Key to the TCP/IP architecture is that it defines a unifying abstraction that permits a wide variety of uses above, and a range of implementations below. In the TCP/IP architecture this unifying abstraction is offered by the Internet Protocol (IP), while the DRM architecture of Jamkhedkar and Heileman provides rights expression and interpretation as unifying abstraction. This paper builds on the latter architecture by identifying the key DRM services of each layer.

The main contribution of this paper is that it presents a next step towards a software architecture that supports reuse and cooperation of multiple domain-specific DRM technologies and standards. It is our belief that this architecture lays the foundation for addressing the above-mentioned challenges of fragmentation, reusability and interoperability, and guides developers of DRM software systems and applications in the right direction.

The remainder of the paper is structured as follows. After briefly sketching the context of DRM in Section 2, Section 3 introduces the core elements of a generic DRM architecture from three application viewpoints: the content consumer, the content producer, and the content publisher. Section 4 zooms in on those points in this architecture where specific security services should be injected in order to establish a secure DRM system. Section 5 combines the functional and security services and presents an architectural overview. Section 6 validates our approach by mapping state-of-the-art DRM technologies onto the architecture and discussing how it supports the three main challenges formulated earlier. Conclusions are formulated in Section 7.

2. CONTEXT

This section introduces the main concepts of DRM, based on which services will be identified in Section 3.

Content refers to the data to protect. This can be audio, video, images, formatted or unformatted text, or other data constructs. Different actions on content are possible (e.g. play, print, or save), each of which may be content type dependent. For example, what could we mean by printing a song? Performing an action on content is called content *consumption*.

The main roles involved in DRM are the producer, the consumer and the publisher. The *producer* is the entity that produces content, or at least owns the rights to distribute and sell it, and represents the starting point of the DRM chain. This can be, for instance, an author, a musician, or a record label (such as EMI, Sony and AOL Time Warner). At the other end of the DRM chain, the *consumer* is the entity who wants to obtain and consume content. For example, a home user that wants to download and play music. The *publisher* is the entity that owns and manages the DRM system used to distribute content, and forms the bridge between consumers and producers. Most DRM technologies available today focus on the consumer aspect.

The on-line *DRM system* is the set of DRM related services offered by the publisher to consumers and producers. The *DRM client* is the entity at consumer side that is responsible for performing the DRM-specific operations in a secure way while obeying the right specifications. The *producer tool* enables producers to add content and corresponding contracts to the DRM system. In this way, content becomes publicly available and licenses can be issued. The *publisher tool* for its part allows a publisher to manage and maintain the DRM system.

The DRM client is the only entity at the consumer side that is allowed to unprotect the protected content; it must not expose this content to other hard- or software. Usually, some kind of encryption is used to protect content. This abuse *prevention mechanism* is often combined with a *detection mechanism*, which allows to identify the source of misuse when illegally distributed content is found.

To formalize the rights a consumer may obtain on some content, usage rules can be defined in a *Rights Expression Language (REL)*. A REL defines a language and vocabulary that enable the specification and interpretation of usage rules in an unambiguous way. For example it can be used to formally describe that a person Bob can listen to a specific DRM protected song only five times in the year 2005. The two most successful RELs are ODRL and MPEG REL [9, 6].

Usage rules must be associated with the corresponding piece of DRM protected content. The concept of *licenses* introduces a separation of DRM protected content and the sets of usage rules to be associated with it. Licenses are typically bound to protected content, but may also be associated with one or more specific devices or legitimate consumers. In the latter case, only these specific devices or persons are able to consume the corresponding protected content using that license. Each distinct set of usage rules can define another *license type*. Different license types may correspond to the same content.

Producers often want to specify the license types themselves, i.e. they want to specify different sets of usage rules describing the rights consumers can obtain. The producer may also want to associate business information with a license type, such as the price and distribution period. All this information is negotiated with the publisher and combined in a *contract* corresponding to the submitted content. This contract contains all information that is needed by the on-line DRM system to issue different license types.

Before consumers are able to use protected content, they first have to obtain it along with a corresponding license that enables them to use the content according to the usage rules described therein. In a typical, simplified scenario, which is illustrated in Figure 1, a consumer uses its DRM client to contact an on-line *Content Service* (1). This service enables to look up and download protected content (2). After a license has been requested (3) and acquired (4) via the *License Service*, the content can be consumed. Instead of distributing content using on-line services, super-distribution might be possible as well. When one obtains the protected content in this way, a corresponding license still has to be obtained by contacting the License Service.

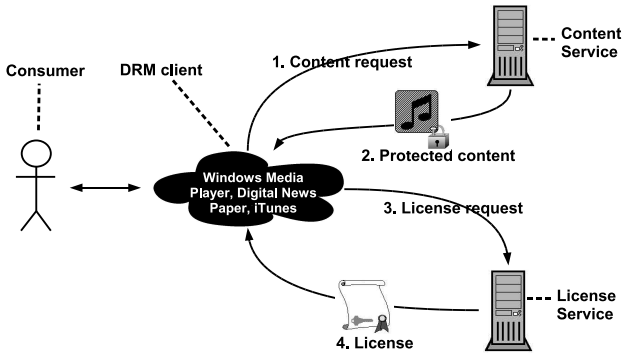

Figure 1: A typical DRM use case: a consumer uses its DRM client to contact an on-line Content Service and search for content. Only when a license has been acquired via the License Service, content can be consumed.

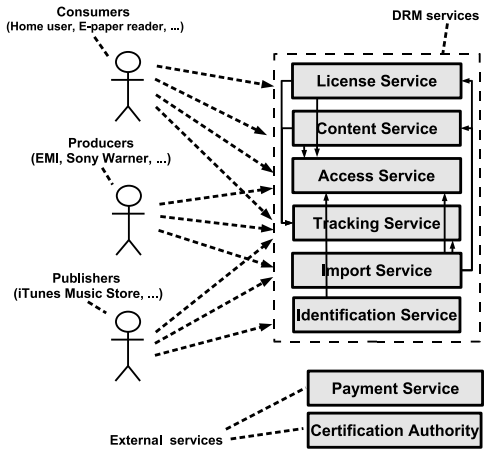

Figure 2: Relevant services in DRM systems

3. A GENERIC DRM ARCHITECTURE

Having sketched the working principles in recent DRM technologies, this section identifies the high level DRM services. First, the high level functionality is studied from different viewpoints, followed by an identification of the different services that should be present.

3.1 Application viewpoints

What is needed at application level to have a complete DRM system? In order to answer this question, three main roles are distinguished: the consumer, the producer and the publisher. In addition, some external roles are briefly discussed. Identifying multiple roles in a DRM system is crucial to view the complete picture.

Consumer. Consumers want to consume protected content in a user-friendly way. They want to be able to browse the content catalog of the on-line DRM system where the content at stake can be obtained. Since consumers also need a license, they must be able to select a license type and view the usage rules associated with it. Generally, consumers first have to pay, one way or another; different business models should be possible (e.g. subscription, pay-per-license, or pay-per-use). When time-based licenses expire, it must be possible to update them, which may also require some financial transaction. Consumers also want to browse their obtained licenses locally and view the usage rules in a human readable format. Finally, consumers want to consume the protected content, according to the usage rules associated with the corresponding license. In order to fetch licenses (and sometimes also protected content), consumers need to authenticate to the on-line DRM system.

Producer. Producers want to easily compose a contract. Both content and contract must be submitted to the on-line DRM system. After some time, they may want to update the contract or maybe even cancel it, i.e. stop the distribution of the content. Content producers expect a financial compensation from the DRM service for the trade of their content. Therefore, they want to receive statistical information from the DRM service about the number of downloads or content usage patterns. In order to query or submit content to the on-line DRM system, content producers need to authenticate themselves.

Publisher. When one or more DRM clients are no longer secure, their right to consume content must be revoked. It may also be necessary to update some parts of the DRM system (and the DRM client). Content publishers may want an overview of system usage patterns. When content is found mass-distributed, the source of abuse must be identifiable. Publishers need to authenticate to the DRM system first.

External roles. Some other roles include the financial institution that offers support for billing issues, and the owner of the network that is used to distribute content and licenses. When certificates are used, an external Certification Authority (CA) is needed to obtain certificates and to check the validity of certificates

3.2 Core components

The previous section presented the main requirements from multiple application-level viewpoints. We are now able to discuss the different high level services and interactions that are needed in a complete DRM system to provide this application-level functionality. We first identify services needed from the consumer's viewpoint, and subsequently add the services needed from the producer's and the publisher's viewpoint. An overview is shown in Figure 2.

3.2.1 Content Consumer

We identify four key DRM services with respect to the content consumer: the Content Service, the License Service, the Access Service, and the Tracking Service. In addition, we describe two external services for payment and certification.

Content Service. When consumers want to obtain protected content, they use their DRM Client to contact the Content Service where they can search for content. Before a consumer obtains any content, the Content Service protects it. Since this protected content may be personalized, the Content Service needs certainty about the identity of the consumer. This is obtained by interaction (i.e. running a protocol) between the Content Service, the DRM Client and the Access Service. This identification phase, however, is not always needed. Dropping it disables the detection mechanism in case of mass-abuse but allows for super-distribution. The Content Service may need to send protection data, specific to that piece of content, to the License Service, to allow

this service to issue associated licenses. The Content Service receives its content and potentially some additional data from the Import Service, which is discussed in the producer part.

License Service. Once the License Service has received the protection data from the Content Service, it is able to issue corresponding licenses upon consumer request. The License Service may offer different license types corresponding to the same content. The consumer requires a description of the different types and selects one. The consumer's DRM client provides some system or user specific parameters to generate a license that is only usable by that consumer. Before the license is issued, the consumer usually has to pay for it. Therefore, the License Service asks the Access Service if it may issue the license. The Access Service in turn contacts the external Payment Service and will only answer positively if the consumer has effectively paid. Expired licenses can be updated by sending them to the License Service, which will issue new ones to the consumer, often after a new payment. The License Service may also send statistical information about the license issuance to the Tracking Service. The Content Service receives its contracts and potentially some additional data from the Import Service, which is explained in the content producer part.

Access Service. The Access Service is responsible for the authentication of the content producers, consumers (or its DRM clients) and publisher. It is also responsible for checking payments of both consumers and producers before allowing particular actions on the DRM system. The Access Service may, for example, deny access if some bills are not paid or if consumers fail to identify themselves.

Before content consumers are able to obtain licenses or even content, some registration procedure may be necessary beforehand. This procedure usually involves the Access Service and, if some financial transaction is needed, the External Payment Service.

Tracking Service. The DRM Client, the License Service and the Content Service can generate statistical usage information (e.g. the number of times a piece of content is downloaded, the type of licenses issued for it, or the number of times it is consumed). These components (or a subset) can forward such information to the Tracking Service, which in turn can produce usage statistics.

Payment Service (external). The financial aspects are taken care of by an external Payment Service, typically a bank or another financial institution. Obviously, some interaction is needed between the external Payment Service, the Access Service and the DRM client, producer tool or consumer tool.

Certification Authority (external) is only necessary if certificates are used. The CA is responsible for the issuance, distribution and revocation of certificates.

3.2.2 Content Producer

We describe two main DRM services offered to content producers: the Import Service and the Tracking Service.

Import Service. Before producers can put content into the DRM system by contacting the Import Service, they must identify themselves via the Access Service. This is necessary to prevent misuse of the DRM system: only the owners of the rights on the content may submit that content to the DRM system. After identification, the producer can submit the content to the Import Service, including metadata and a corresponding contract. Upon receipt, the Import Service may first convert and manipulate the content such that the DRM system is able to deal with it. Subsequently, the content is sent to the Content Service, while the contract and additional information (such as the content identifier) are send to the License Service. In a similar way, content can be updated or removed from the DRM system.

Tracking Service. Producers can also ask the Tracking Service for usage statistics corresponding to their content and can check, by contacting the external Payment Service, if they are compensated accordingly.

3.2.3 Content Publisher

The content publisher uses three major DRM services: the Access Service, the Identification Service, and the Tracking Service.

Access Service. The publisher first of all authenticates to the DRM System using the Access Service. Secondly, the Access Service allows the publisher to revoke a single DRM client or a whole class of clients, for example all DRM clients that have not yet been renewed after a security threat.

Identification Service. The publisher can send content that is found on mass-distribution networks to the Identification Service, which will reveal the identity of the source of abuse.

Tracking Service. Similar to the other roles, the content publisher is also able to obtain statistical information about the use of the DRM System.

4. INTEGRATING SECURITY TECHNOLOGIES

While the previous section focused on the main functional components of a generic DRM architecture, this section switches attention to security. It first identifies hot spots in the architecture that highlight where to inject specific security services to establish a secure DRM system. Secondly, it provides an overview of the main cryptographic primitives and means that are required to implement the security services.

4.1 Locating security hot spots

Where in the proposed architecture should one inject specific security services to establish a secure DRM system? The major security services that are identified are unforgeability, integrity, authentication, confidentiality, non-repudiation, and anonymity. We discuss what services are needed to secure licenses, protected content, contracts, newly submitted content, the on-line DRM system and the consumer system.

Licenses. A secure license must first of all be *unforgeable*, meaning that it should be impossible to parties other than the License Server to construct a valid license. A second important property to be guaranteed is *integrity*, i.e. any change in a license will be detected and thus becomes invalid. Thirdly, a license must be *uniquely bound to* its corresponding content. The same is true for the owner of the license. Today, it is still difficult in practice to bind a license to a physical person. Binding to one or more devices owned by that person is often applied instead.

Content When a content producer submits new content, a third party may not tamper with (*integrity*) or have access to (*confidentiality*) it. *Authentication* must be possible

to determine who is responsible for the content submission to the DRM system. The protected content spread by the Content Service should only be consumable by owners of a legal corresponding license; to all others, *confidentiality* of the content is needed. Practice has shown that it is very hard to prevent consumers from getting hold of the content in an unprotected way. For this reason, the Content Service may also need to apply a detection mechanism on the unprotected content to support non-repudiation when mass-abuse is detected.

Contracts. If a person illegally submitted content to the DRM system, one must be able to prove that this person committed a violation. This requires *authentication* and *non-repudiation*, such that the suspect cannot deny the facts. *Integrity* also protects against tampering by a third party.

Consumer system. At the system of the consumer, only entities (such as the operating system, a sound card, or a DRM client) that can *authenticate* themselves as trustworthy may become authorized to get hold of the content in an unprotected way. In this way, *confidentiality* of the content is maintained. *Integrity* of authorized software entities should be maintained; if not, one might as well insert a Trojan horse that sends unprotected content to a file or another party. Because the rights defined in licenses are often time related, there is a need for *secure time*: it should be impossible to use an expired license by simply changing the system time. Another important property for DRM clients is *individualization*, which reduces the danger of global breaks if one specific DRM client is compromised.

Online DRM system. In the DRM system, sensitive data (unprotected content, key data) are stored and sent from one component to another. For example, the License Service holds secret data that are needed for creating valid licenses. Thus, *confidentiality* is needed and *integrity* must be kept for both the sending and storing of sensitive data. When sensitive data are sent from one component to another, mutual *authentication* is needed. When usage information is sent to the Usage Tracking Server, the *anonymity* of the sender must be respected. No third party may change the statistical information. This requires again *integrity* and *authentication* of both the sending and the receiving component. *No replay-attacks* are allowed to prevent usage information being sent more than once.

4.2 Establishing security services

Section 4.1 located security hot spots and identified the necessary security services. This section outlines how these security services can be established in particular hot spots. Since the same security service may be established by different security services, depending on the situation, this section presents a mapping between security hot spots and cryptographic primitives that can be used to secure them (e.g. digital signatures, certificates, encryption, and digital watermarks[1]).

Licenses. *Unforgeability* and *integrity* can be realized by applying digital signatures. *Binding* the license to its corresponding content can be done by inserting a unique identifier of the content in the license. Such an identifier may be a hash of the data, a fingerprint of the content, a Digital Object Identifier [22], or some other kind of identification. The content can be bound with a user can be done by inserting the user's personal certificate or a unique identifier of that certificate.

Content. When content is submitted, the combination of digital signatures and certificates can be used to obtain *integrity* and *authentication*. *Confidentiality* can be obtained using encryption. Obtaining confidentiality of DRM protected content has shown to be far from trivial. Content may be encrypted with a key that can only be reconstructed by a specific DRM client, potentially with the help of the consumer. Key reconstruction may depend on hardware specific parameters such as the CPU serial number. Another possibility is putting key data in the corresponding licenses. *Non-repudiation* in the detection mechanism of protected content can be obtained by a combination of digital signatures, certificates and watermarking. Identifying data signed by the consumer can be embedded as watermark in the content before it is encrypted.

Contracts. *Integrity*, *authentication* and *non-repudiation* can be obtained by using digital signatures combined with certificates.

Consumer system. To obtain *authentication* of the different trustworthy entities, combined with *confidentiality* of the sensitive data, Trusted Computing [2] techniques can be used. To decrypt the DRM protected content, the DRM client needs a secret key that is not known to the consumer. To prevent consumers or other entities from getting hold of the secret decryption data, white box cryptography [4] can be used. Such key data can also be used to maintain *confidentiality* of license status info (e.g how often it has already been consumed). To obtain *integrity* of the DRM client (and possibly other software entities) self checking techniques [10] can be used. Code obfuscation [24] is a usable technique to hinder sensitive code analysis. It also provides a way for *individualization*. To obtain secure time, one can use secure clocks, i.e. tamper-resistant hardware clocks.

Online DRM System. *Confidentiality* can be obtained using encryption, *integrity* using digital signatures and *authentication* using digital certificates. *Anonymity* can be obtained by removing any identifying data out of the usage information, while the usage information itself is sent to the Tracking Service using anonymizers [20]. By using digital signatures for authentication, anonymity would be lost. Therefore, zero-knowledge proofs [14] that contain the usage information should be used instead.

5. AN ARCHITECTURAL OVERVIEW

This section integrates the results of sections 3 and 4, i.e. the three main roles identified in a DRM system (consumer, producer, and publisher), the seven most important DRM services (content, license, access, tracking, payment, import, and identification), and the various cryptographic primitives to implement security services (a.o. encryption, digital signatures, certificates, watermarks, and secure clocks). Identifying key DRM services and locating them in an overall structure brings us one step closer to a software architecture for DRM.

The section starts by positioning the content consumer, producer, and publisher in a high level distributed view on the DRM architecture. Subsequently, it identifies the layers for each party of the distributed view. After this high level overview, the section zooms in on the top layers of the architecture, which represent application services, and

[1]Watermarking can be used to embed information into content in an imperceptible way.

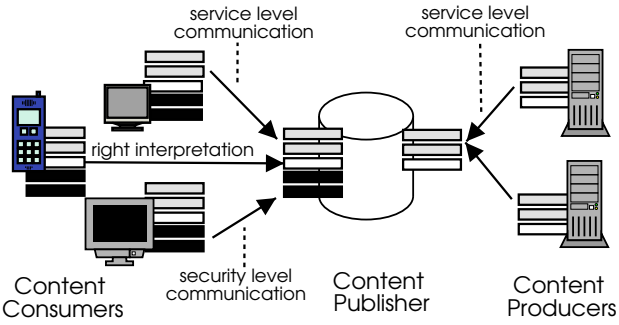

Figure 3: A distributed view on a DRM architecture with content consumers, producers and a publisher. The figure shows different levels of communication between layered architectures in each party. The gray levels represent application services (e.g. payment, tracking, or licensing). The white level represents right expression. The black levels represent rights enforcement technologies (e.g. watermarking, certificates, or digital signatures). Each party can provide a different subset of layers.

the bottom layers, providing security and rights enforcement technologies.

5.1 A distributed view

Figure 3 illustrates three aspects of DRM. First of all, it gives a high-level overview of the distributed DRM architecture, with content consumers, producers and a publisher. This overview matches the application viewpoints presented in Section 3.1.

Secondly, the figure shows parties interacting at different levels, which clearly matches a layered architecture. Content consumers, for instance, communicate with the publisher at both service and security level. Service level communication relates to issues such as licensing, tracking, importing, or paying. Security level communication enforces protection of published content by using techniques such as watermarking, encryption, or certificates. In addition to multi-level communication, the figure illustrates that not every level is relevant for each party. Content producers, for instance, interact with the publishing system by exclusively using service level communication, while consumers communicate at both the service and the security level.

Finally, the figure highlights a client-server interaction pattern between consumer and publisher, and producer and publisher. This implies that both sides must provide (part of) the interaction protocol or technology in use. Consequently, the publisher perspective on services, presented in Section 3.1, should be extrapolated to both the consumer's and the producer's side.

In summary, the figure intuitively motivates why the three parties should provide a layered DRM architecture, while not every party has to offer each level of communication.

5.2 Identifying layers

Based on the architecture presented by Jamkhedkar and Heileman [12], we identify five layers, from top to bottom: the application layer, the negotiations layer, the rights interpretation layer, the upper rights enforcement layer (REL), and the lower REL (see also Figure 4).

We classify these layers into three main groups: the service level (shown in gray), the right interpretation level (in white), and the security level (in black). At the upper two services layers, content is used and manipulated according to the rights associated with it. In the middle, the rights interpretation layer is concerned with how rights are specified and how they should be interpreted in particular environments. The lowest layers are responsible for rights enforcement on the consumer side.

The following sections map the negotiation and rights enforcement layers onto the discussions in previous sections. The right interpretation layer falls outside the scope of the paper, and will not be discussed in further detail.

5.2.1 Negotiation layer

The diversity of services is located at the negotiation layer. This layer provides a selection of services to satisfy the varying needs of different applications [12]. Figure 4 shows the main services that have been discussed in detail in Section 3.2: content, license, access, payment, import, and identification. The Tracking Service can be seen as a sub-layer that intercepts and logs relevant communication of upper services.

5.2.2 Rights enforcement layers

The security technologies introduced in Section 4 reside at the two bottom layers of the architecture. The upper layer is responsible for type dependent manipulations, while the lower layer handles content irrespectively of its type. Figure 4 shows rights enforcement technologies such as watermarking, digital signatures, certificates, and encryption.

6. VALIDATION

By way of validation, we describe how state-of-the-art DRM technologies can be mapped onto the identified service components. The objectives are to verify to what extent existing technologies can be represented by using the layered architecture proposed in this paper, and to identify the service components that are represented most often, i.e. the places in highest need for reuse and interoperation.

The validation is based on six DRM technologies of which technical information was publicly available: Windows Media DRM, Lightweight DRM, EMMS, Helix DRM, Aegis DRM, and the OMA specification. Unfortunately, we could not integrate well-known technologies like Fairplay and Intertrust by lack of information. Yet, we are convinced that for this evaluation the presented group of technologies offers a sufficiently broad overview.

First of all we introduce each individual technology and map its services to the seven we have identified in Section 3.2. Secondly, we discuss this comparison and summarize it in a matrix overview.

6.1 Technology overview

6.1.1 Windows Media DRM

Microsoft's Windows Media DRM [16] (WMDRM) allows protection of audio and video. The content can be played on a Windows PC or portable device.

WMDRM consists of a set of Software Development Kits (SDKs): *Content Packaging* to protect compressed media, *Content hosting* to host and distribute digital media content, *License Clearinghouse* to issue licenses and track trans-

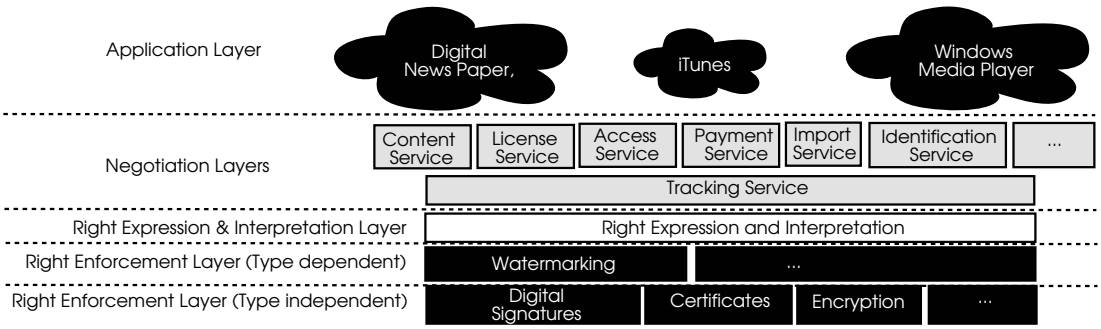

Figure 4: DRM as a layered architecture identifying key services at the negotiation layer, such as content, license, access, payment, import, identification, and tracking. Security technologies, such as watermarking, digital signatures, certificates, and encryption, are located at the rights enforcement layers.

actions, *Content playback* to play protected digital media, and *Portable device playback* to transfer and play protected digital media.

The **Content Service** can be established using Content Hosting, probably in combination with Content Packaging when content needs to be protected on the fly. The **License Service** functionality is part of the License Clearinghouse. The **Tracking Service** can be implemented using part of the License Clearinghouse functionality. A **Payment Service** is not supported, but existing payment solutions can be integrated when a DRM system is developed. An **Import Service** can be implemented using Content Packaging. No **Payment Service** and **Access Service** are supported in WMDRM, but existing payment or access control solutions can be integrated. No information has been found about support for an **Identification Service**.

6.1.2 Light Weight DRM

The usual DRM schemes are strong in the sense that they enforce the usage rules at the consumer side very strictly (e.g. a consumer can play specific content only ten times or can play a song on only three devices). According to Fraunhofer Institute, this is an important reason why DRM is currently not very well accepted by the market. To overcome this problem, they propose Light Weight DRM (LWDRM [8]), which allows consumers to do everything they want with the content they bought, except for large scale distribution. This form of use is called *fair-use*.

LWDRM uses two file formats: the Local Media File (LMF) and the Signed Media File (SMF) format. An LMF file is bound to a single local device by a hardware-driven key but can be converted into the SMF-format, which can be played on whatever device that supports LWDRM. However, the identity of the legitimate owner is then embedded in the content.

Although development of LWDRM has stopped since 2004, the concepts and services offered are highly relevant when comparing various DRM technologies.

The LWDRM architecture contains the following components. The *Client Tool* is a consumer side application and is used by consumers to register, search, obtain and play content. The only component interface that is queried by the Client Tool is the *Accounting Service* interface. For being able to download protected content, consumers need to identify themselves with a certificate and have to pay using the *Payment Service*. The Accounting Service administers relevant information about the available content, such as price, meta-data and owner, and stores the user accounts with all associated transactions and purchases. By contacting the Certificate Authority the certificate is validated. Following, the Accounting Service sends to the *Content Packer* a request to send content to the Client Tool. The Content Packer generates a new LMF file when it receives a request from the Accounting Service, and sends the LMF file directly to the Client Tool. The external *Payment Service* offers public interfaces to do and check payments. The Certification Authority issues and revokes certificates.

The **Content Service** functionality is partly offered by the Accounting Service and the Content Packer. Because LWDRM does not offer licenses, there is no need for a **License Service**. The **Access Service** functionality is offered by the Accounting Service. Limited **Tracking Service** functionality can be offered via the Access Service. The **Payment Service** is external to the LWDRM system, and no **Import Service** is offered. Although an **Identification Service** is absolutely necessary in a system as LWDRM, we could not find any information on it.

6.1.3 EMMS

IBM's Electronic Media Management System (EMMS [11]) offers DRM protection for video, music, documents, rich media and software, independent of the format used. Streaming is supported. IBM Recently announced that it will withdraw EMMS v2.1 from the market without releasing a replacement product. Nevertheless, it is still interesting to match its high level components with the services defined in Section 3.

The whole system consists of seven components. The *Web Commerce Enabler* provides EMMS DRM functionality for web applications and enables consumers to query for content and licenses. Transaction information is gathered and sent to the *Clearinghouse*. The *Client SDK* offers an API enabling development of DRM aware consumer applications. It enables to browse, buy and download content and/or licenses on Web Commerce Enabler servers and to consume the DRM protected content according to the licenses. The *Content Preparation Development SDK* offers producers the functionality to package content. Corresponding contracts can be made and meta-data can be specified. The protected content can be distributed using any means, but is usually

sent to a server running the *Content Hosting Program*. The corresponding key data are sent to the Clearinghouse via the Web Commerce Enabler. The Hosting Program offers support for hosting. This component receives protected content from the *Producer Client*, using the Content Preparation Development SDK. All distribution activity is tracked and sent to the Clearinghouse which is the central control point. It receives key data and contracts from the Web Commerce Enabler, which thus enables the Clearinghouse to issue licenses. It receives its tracking data from the Web Commerce Enabler and the Content Hosting programs, and can generate license transaction tracking data itself. Finally, transactions are authorized here, for which contacting an external *Payment Service* may be needed.

The Hosting program offers the **Content Service**, The Clearinghouse offers the functionality of the **License Service**, the **Tracking Service** and the **Access Service**. The Content Preparation Development SDK enables one to set up an **Import Service**. Nothing is known about an **identification Service**. An external **Payment Service** can be contacted by the Web Commerce Enabler.

6.1.4 Helix

Helix DRM [19] was announced on January 9, 2003 by Real Networks. It is designed to be integrated in existing e-commerce applications. Multiple business models can be applied such as subscription, pay-per-consume and promotions. Helix DRM focuses on video and audio. Streaming, downloading and other delivery methods are possible (using super-distribution). Different platforms are supported.

Helix DRM consists of four central components. The *DRM Packager* protects content as preparation for distribution. The *DRM License Server* verifies license requests and issues licenses and provides auditing information to facilitate royalty payments. It allows to manage, authorize, and report content transactions. The *DRM Client* allows streaming and playback of content by the consumer and allows content cosumption according to the licenses. The *DRM for devices* enables device manufacturers to equip their devices with Helix DRM support.

The DRM License Server plays the role of the **License Service**, **Tracking Service** and **Access Service**. The DRM Packager corresponds to the **Import Service** and the **Content Service**. Nothing is known about an **Identification Service**. An external **Payment Service** can be used.

6.1.5 Aegis

AegisDRM [1] focuses on intra-company content distribution. A myriad of content types such as spreadsheets, PDF documents, web pages and applications are supported.

AegisDRM offers four entities, which can be used to interact with an e-commerce server. The *Protector* allows the content producer to protect content and to specify usage rules. The *PaM (Protector add-in)* is a plug-in for MS Office that allows the producer of an office document to specify usage rules. Every time consumers want to perform actions on some protected content, they must get authorization of the *Rights Server*. Using the Rights Server, information can be changed or revoked at any time. The *License Master* issues licenses and is an alternative to the Rights Server.

Both Protector and PaM correspond to the **Producer Tool** together with a producer side **Import Service** (packaging). The License Master and Rights Server correspond to the **License Service** and the **Access Service**. Other Services are not supported since AegisDRM focuses on intra-company content, which makes the **Payment Service**, the **Content Service**, and **identification Service** less relevant.

6.1.6 OMA

The Open Mobile Alliance (OMA [18]) is a forum that offers the mobile industry a comprehensive open specification to enable interoperability of different service providers and mobile devices. Different companies can implement the specification while maintaining interoperability. One topic OMA focuses on is DRM for mobile devices.

The DRM architecture as specified in the OMA DRM v2.0 Candidate Enabler is very simple. At the server side, we have a *Content Issuer* and a *Rights Issuer*. The *DRM Agent* (i .e. the DRM client) receives protected content by super-distribution from another DRM Agent or can look up and download protected content from a Content Issuer. For being able to preview or consume content, the DRM Agent contacts the Rights Issuer where it can obtain a *Rights Object* (i.e. a license). A lost or damaged Rights Object may still be restored via the Rights Issuer by requesting a new Rights Object. A DRM Agent can also request the Rights Issuer to join or leave a domain, which is a set of DRM Agents able to use the same *Domain Rights Objects*, i.e. Rights Objects dedicated to a domain. Before a DRM Agent is able to download a Rights Object, it has to complete a registration procedure with the Rights Issuer. The DRM Agent is obliged to prove ownership of a certificate, which is checked on validity by the Rights Issuer by contacting an external Certification Authority.

The **Content Service** corresponds to the Content Issuer, and the **License Service** to the Rights Issuer. No specifications for the **Access Service** have been found, but there is a track within OMA that focuses on access control. Nothing is specified about a **Tracking Service**, **Import Service**, **Identification Service** or a **Payment Services**, although an interface is specified for the latter.

6.2 Discussion

As the overview in Table 1 shows, some services are provided almost uniformly by all technologies, while others are only offered sporadically. The Content and License Services are almost always implemented, which seems nothing but normal for such key services. Services for accessing, tracking, paying and importing are provided in approximately 50% of the cases, while the Identification Service is not implemented by any of the studied DRM techniques, at least not to our knowledge.

When relating these results with the three main DRM challenges presented in Section 1 (completeness, interoperability, and software architecture support), we can draw the following conclusions. First of all, the fact that so many different DRM technologies implement the same or similar services confirms our claim that we need an architecture that promotes reuse and interoperation of individual service components.

Secondly, the table shows that the services with the highest benefit from reuse and interoperation are the Content and License Service. All DRM technologies that need these services would benefit from a reusable implementation.

DRM Technology/Service	Content	License	Access	Tracking	Payment	Import	Identification
WMDRM	X	X	-	X	-	X	-
Light Weight DRM	X	-	X	-	X	-	-
EMMS	X	X	X	X	X	X	-
Helix	X	X	X	X	-	-	-
Aegis	-	X	X	X	-	-	-
OMA	X	X	X	-	X	-	-

Table 1: Overview of provided services of state-of-the-art DRM technologies.

Thirdly, since judging from the table different DRM technologies implement different sets of services, trying to standardize 'the' DRM technology seems less efficient than focusing on particular services these technologies are composed of. This brings us back to the analogy with the Internet architecture, which clearly identifies service responsibilities and a common platform that can support a wide variety of networking services. The key message is that this architecture proves that a complete solution can be offered by a single platform if it allows reusable services to be plugged in, without trying to provide a single overall standard implementation. Until today, many different companies and organizations extend the TCP/IP architecture with protocols for quality-of-service, wireless communication, routing, media streaming, or security. If we are to provide complete DRM solutions, following the Internet approach seems to be a good idea.

However, we should be aware that the Internet approach cannot be adopted as such in the domain of DRM. Although the idea of using a layered architecture for DRM solutions looks very promising, we argue that the match between TCP/IP and DRM is not complete for two reasons. First of all, the DRM architecture does not completely adhere to a layered structure. This is especially true when looking at the architecture from the perspective of adaptability and manageability, two crucial quality attributes for DRM systems, which often have to be tuned to various business policies or local legislations [7]. Such concerns can turn the main advantage of layering, i.e. virtualization of lower layer details, into a major disadvantage. This situation occurs, for instance, when lower layers do not behave exactly as required by upper layers or applications [23]. In this case, applications should be able to fine-tune the underlying system by injecting specific policies [13]. This is a generic problem that has already been explored in other application domains than DRM, for example protocol stacks [15].

The second reason to be careful when comparing TCP/IP and DRM is that the architecture of the latter will not always be symmetric: while a TCP/IP client runs exactly the same protocols as the server, this is not necessarily the case for DRM systems. The right expression layer, for instance, will probably be fully implemented at the publisher's side to allow for content producers to associate with their content a wide variety of business policies. Yet, at the content consumer's side, this layer will be minimally implemented to prevent clients from tampering with business policies. The same is true for rights enforcement technologies such as watermarking, digital signatures, or certificates.

7. CONCLUSION

Starting from the identification of major DRM components, this paper has proposed a next step towards a generic DRM architecture and subsequently mapped six DRM systems onto it, thereby enabling a critical evaluation. Special attention went to three main DRM challenges: completeness, interoperability, and software architecture support. The underlying model consist of a distributed view and perspectives from consumer, producer, as well as the publisher, a layered architecture for each party, and identification of components in each layer. This model has proved to be a useful framework to inventory, analyze, and discuss research in this field, and to set the agenda for the future. Inspection of this framework shows the use of Content and License Services to be well represented in the DRM technologies described. The uniform application of services like access, import, tracking, and payment services is less developed, whereas identification is absent. Therefore, if DRM is not to end as the umpteenth flash in the data protection pan, it may be high time to put software architecture design at the top of its research agenda.

Acknowledgements

Research for this paper was sponsored by IBBT, the Interdisciplinary institute for BroadBand Technology, and conducted in the context of the E-paper project. The authors are very grateful to Ann Heylighen for her valuable comments on the paper and for proof reading the text.

8. REFERENCES

[1] Aegis DRM. http://www.aegisdrm.com/.

[2] R. Anderson. Cryptography and competition policy: issues with 'trusted computing'. In *PODC '03: Proceedings of the 22nd annual symposium on Principles of Distributed Computing*, pages 3–10. ACM Press, New York, NY, USA, 2003.

[3] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[4] S. Chow, P. A. Eisen, H. Johnson, and P. C. van Oorschot. A White-Box DES Implementation for DRM Applications. In *Proceedings of ACM Workshop on Digital Rights Management (DRM2002)*, pages 1–15, Washington DC, USA, Nov 2002.

[5] D. Clark. The design philosophy of the darpa internet protocols. *Proceedings of SIGCOMM, Computer Communication Review*, 18(4):106–114, Aug. 1988.

[6] K. Coyle. Rights expression languages. Technical report, A Report for the Library of Congress, Feb. 2004. http://www.loc.gov/standards/.

[7] J. S. Erickson. Fair use, DRM, and Trusted Computing. *Communications of the ACM*, 46(4):34–39, 2003.

[8] Fraunhofer Institute. Light Weight DRM (LWDRM). http://www.lwdrm.com/.

[9] S. Guth. Rights Expression Languages. In E. Becker, W. Buhse, D. Günnewig, and N. Rump, editors, *Digital Rights Management: Technological, Economic, Legal and Political Aspects*, volume 2770 of *LNCS*, pages 101–112. Springer, Nov. 2003.

[10] B. Horne, L. Matheson, C. Sheehan, and R. E. Tarjan. Dynamic self checking techniques for improved tamper resistance. In *Proceedings of the ACM Workshop on Security and Privacy in Digital Rights Management*, pages 141–159, 2001.

[11] IBM. IBM Electronic Media Management System (EMMS). http://www-306.ibm.com/software/data/emms.

[12] P. A. Jamkhedkar and G. L. Heileman. Drm as a layered system. In *DRM '04: Proceedings of the 4th ACM workshop on Digital rights management*, pages 11–21, New York, NY, USA, 2004. ACM Press.

[13] G. Kiczales, J. Lamping, C. V. Lopes, C. Maeda, A. Mendhekar, and G. C. Murphy. Open implementation design guidelines. In *Proceedings of the 19th International Conference on Software Engineering (ICSE'97)*, pages 481–490, Boston, MA, USA, 1997. ACM Press, New York, NY, USA.

[14] W. Mao. *Modern Cryptography. Theory & Practice*. Prentice Hall PTR, Indianapolis, IN, USA, 2003.

[15] S. Michiels. *Component Framework Technology for Adaptable and Manageable Protocol Stacks*. PhD thesis, K.U.Leuven, Dept. of Computer Science, Leuven, Belgium, Nov. 2003.

[16] Microsoft Corporation. Windows Media DRM (WMDRM), 2005.

[17] D. K. Mulligan. Special Issue: Digital Rights Management and fair use by design. *Communications of the ACM*, 46(4):30–33, 2003.

[18] OMA. Open mobile alliance. http://www.openmobilealliance.org.

[19] Real Networks. Helix DRM. http://www.realnetworks.com/products/drm.

[20] M. Reed, P. Syverson, and D. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communication Special Issue on Copyright and Privacy Protection*, 16(4):482–494, May 1998.

[21] C. Szyperski. *Component Software: Beyond Object-Oriented Programming*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1998.

[22] The International DOI Foundation. The Digital Object Identifier (DOI). http://www.doi.org/, June 2005.

[23] M. Welsh and D. Culler. Virtualization Considered Harmful: OS Design Directions for Well-Conditioned Services. In A. D. Williams, editor, *Proceedings of Eigth Workshop on Hot Topics in Operating Systems (HoTOS-VIII)*, pages 139–146, Elmau, Germany, May 2001. IEEE Computer Society Press.

[24] G. Wroblewsky. *General Method of Program Code Obfuscation*. PhD thesis, Wroclaw University of Technology, Feb. 2002.