

DRM Interoperability Analysis from the Perspective of a Layered Framework

Gregory L. Heileman and Pramod A. Jamkhedkar
Department of Electrical & Computer Engineering
University of New Mexico
Albuquerque, NM 87131
{heileman,pramod54}@ece.unm.edu

ABSTRACT

Interoperability is currently seen as one of the most significant problems facing the digital rights management (DRM) industry. In this paper we consider the problem of interoperability among DRM systems from the perspective of a layered architectural framework. The advantage of looking at the problem from this point of view is that the layered framework provides a certain amount of structure that is very helpful in guiding those working on DRM interoperability issues. Specifically, the layered framework we describe provides a useful design abstraction along architectural lines. One of the advantages of this perspective is that it allows us to consider the level within computing/communication architectures at which certain functionality should be provided, and then to address how the functionality between layers should interact in order to provide specific DRM capabilities. The communications that occur between layers, both within a single system and between two communicating systems, are the places where protocols can be defined and possibly standardized. Thus, they provide focal points for studying and addressing interoperability in DRM systems.

Categories and Subject Descriptors

C.2.6 [Computer-Communication Networks]: Internetworking—*Standards*; H.5.1 [Information Systems]: Interfaces and Presentation—*Multimedia Information Systems*; K.5.1 [Legal Aspects of Computing]: Hardware/Software Protection—*Copyrights*

General Terms

Design, Legal Aspects, Security, Standardization

Keywords

DRM, Interoperability, Layered Architecture

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DRM'05, November 7, 2005, Alexandria, Virginia, USA.
Copyright 2005 ACM 1-59593-230-5/05/0011 ...\$5.00.

1. INTRODUCTION

Interoperability is a difficult issue to address in any computing infrastructure, and it is currently seen as one of the most difficult problems facing the digital rights management (DRM) industry. A number of approaches have been proposed for supporting interoperability among DRM systems [10, 14, 15], and a high-profile consortium was recently formed with the express purpose of promoting interoperability between DRM technologies used in the consumer media market [4]. The goal of this consortium is to create an open technology framework that will enable a simple and consistent digital entertainment experience for consumers. In this paper we provide a architecture-based systematic framework for considering how interoperability can be pursued in DRM systems. We believe that any standardization or consortium effort could benefit from considering this point of view.

The benefits provided by making the components associated with complex technology-related systems interoperable are well known, and in fact the desire to exploit these benefits provides one of the primary driving forces behind the creation of standards bodies. We have seen in the past how the adoption of standards, *de facto* or otherwise, that promote interoperability have greatly facilitated the development of certain markets. For example, the early market for personal computers grew significantly with the availability of the “open architecture” provided by IBM. Furthermore, the subsequent explosive growth of telecommunications markets was greatly facilitated by the adoption of standard protocols that allowed for interoperability, including TCP/IP, HTTP and SMTP, to name a few. It is interesting to compare the situation in these markets to those that existed in the early DRM markets. The early non-trivial DRM solutions, i.e., those that involved the explicit management of rights and not just content protection, were single-vendor monolithic systems. Here we are referring to the DRM solutions provided by for example Intertrust and ContentGuard circa 2000. Many cite the lack of interoperability in these types of systems as one of the main reasons that projected DRM markets did not develop. In response, a variety of standardization efforts associated with certain aspects of DRM have recently been initiated. Still, these efforts, and the field of DRM itself, are at an early stage of development, and therefore a viable open DRM architecture has yet to emerge.

In this paper we consider the interoperability problem from an architectural perspective, while the work we are aware of in this area to date tends to view the problem along process lines. In particular, these previous efforts tend

to pose and consider DRM problems in the form of what are in essence use-cases. Generally, the actors associated with a particular DRM problem are defined (e.g., the Creator, the Rights Holder, the Distributor, the Purchaser, etc.) and then a particular DRM functionality (e.g., subscriptions, superdistribution, etc.) is developed as a use-case by specifying how these actors should interact in order to implement the specific DRM functionality. Finally, the use-case is analyzed with interoperability in mind. This tends to yield a fairly coarse-grained partitioning along the lines of how particular pieces of the functionality can be implemented among the actors when different DRM technologies are being used. Quite often the solution makes use of intermediaries or translation services in order to move content and licenses from one DRM regime to another. Certainly this type of analysis is beneficial; however, so far it appears to be most useful in the study of macro-scale DRM interoperability. As an analogy, this is similar to studying how one might execute the programs from one computing environment, call it X , in a completely different computing environment, say Y . This can be accomplished by constructing a “virtual machine” for X that runs on Y . Partitioning DRM functionality along architectural lines produces an alternative view that leads to new insights in the analysis of DRM interoperability. Furthermore, it should be noted that the procedure-oriented and architecture-oriented views are synergistic—they simply provide different views of DRM. For certain situations one view may prove more useful than the other, and in fact we will demonstrate later how a DRM use-case can be overlaid onto a layered DRM architecture.

Another important benefit of the architectural view is that it also allows one to compartmentalize certain functionality that should *not* be standardized. This last point is extremely important. Everyone agrees that standards must be established if DRM systems are to be made interoperable. To do this, important points of interaction must be identified, and standards must be developed at these interfaces. However, a vital point that is generally overlooked in the discussion of DRM standards is that it is just as important to specify the places where standards should *not* play. The reasons for this were eloquently described by Clark et al. with reference to “tussles” in cyberspace [3]. Specifically, these authors noted that standards and innovation often work at cross purposes. Thus, it is important to clearly identify those locations within a system where standards will not be defined, and therefore vendors are free to innovate. These so called “tussle spaces” provide areas where competition can occur as vendors attempt to differentiate themselves from one another. This does not lead to non-interoperable systems as long as the boundaries of the tussle space are well-defined, and standards exist at these boundaries. Of course different vendors will have vested interests in defining the dimensions of the tussle spaces differently. How these spaces are ultimately decided will be determined by many issues related to technology, market dominance, litigation and standardization efforts. In any event, we believe it is very useful to view these tussle spaces from an architectural perspective.

An overview of this paper is as follows. In Section 2 we briefly review the previously proposed layered DRM framework, and point out areas within the architecture where protocols (and hence standards) can be defined. Section 3 discusses previous research on DRM interoperability, as well

as the direction that it is likely to evolve. This is followed by a detailed explanation of thinking about DRM interoperability from the perspective of a layered framework. In particular, Section 4 explores the different degrees of interoperability that can be achieved by making specific layers of the DRM framework independent of the rest of the system. We also explain the implications of this independence in terms of required standards and protocols. Section 5 provides a case study of Microsoft DRM 10 architecture, where we overlay the architecture onto our layered DRM framework and study standards and protocols required to achieve device- and operating system-level independence. In Section 6 we provide some useful concluding remarks.

2. A LAYERED DRM FRAMEWORK

We previously proposed a framework for DRM architectures that was motivated by the design principle of breaking up the functionality of a complex system into layers [9]. That is, we partitioned the functionality of a generic DRM system into layers according to the types of services that should be provided within each layer. We previously noted the main advantages of this approach. In particular, it provides an abstraction mechanism that allows developers to more effectively deal with the complexity of the system. Because only those interactions that occur between layers need to be considered, this approach simplifies the analysis, design and development of DRM systems. Furthermore, it reduces complexity since we only need to worry about the services provided by a layer, and how to communicate with the layers above and below, and with peer layers on other systems. It should also be noted that this approach promotes interoperability since with appropriate handshaking protocols, vendors can independently focus on technologies that address different parts of the system. The Internet is an example of another system that evolved around these same principles.

From the perspective of interoperability, one of the advantages of a layered architecture is that the specification of the layers themselves helps to fix the architecture to a certain extent, thereby providing enough stability so that standards can be developed around how the layers interact. Thus, determining what layers should exist in the architecture is a vitally important precursor to the development of DRM standards that will support interoperability. We proposed a specific set of layers for DRM architectures in [9]. In this paper we take the logical next step of elaborating the interactions between layers on the client side, and between peer layers on the client and server sides, with particular emphasis on how interoperability is supported by this approach. The basic tenet is that protocols and procedures at one layer should be independent of those at other layers.

First let us briefly review the layers in the previously proposed framework. These are shown in Figure 1. The DRM layers in this figure involves a pair of protocol stacks, one for the content distributor (DRM server) and one for the content consumer (DRM client). We previously coined the term *rights enforcement* to refer to the process of enforcing specified digital rights within a DRM environment [5]. Because rights enforcement technologies are generally not implemented on the server side, the complete set of layers only appears on the client side. We can partition the layers on the client side into three groups. The upper layers consist of the Application and Negotiations Layers. These

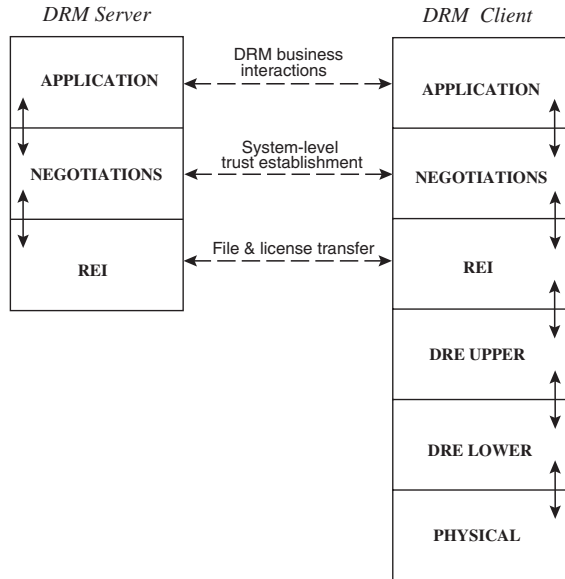


Figure 1: The interactions between the client and server sides in a layered DRM framework. Dashed arrows correspond to inter-system communications, while solid arrows correspond to intra-system communications.

layers create the high-level services that are frequently used by applications that involve DRM. Thus, we also refer to them as services layers. As the name suggests, the Application Layer is concerned with applications that make use of rights-managed content. This layer is responsible for providing the interface that users interact with while using content. Thus, the Application Layer is responsible for content selection, personal trust and monetary transactions. System-level trust is established in the Negotiations Layer. Accepting requests for licenses and assessing the capabilities of rendering devices are particular functions that involve system-level trust. The results of negotiations regarding system-level trust may influence the type/value of content delivered by the content distributor, and will likely involve certification of client-side technologies.

The lower layers, responsible for implementing rights enforcement technologies, include the DRE Upper, DRE Lower, and Physical Layers. The DRE Upper Layer consists of high-level rights enforcement technologies such as encryption, watermarking, and digital certificates. Thus, the DRE Upper Layer is responsible for license management, access control (to content), and cryptographic protocols. The DRE Lower Layer is concerned with technologies that exist at the operating-system level, including device drivers and secure containers. Thus, the DRE Lower Layer is responsible for low-level rights enforcement functionalities that include device driver authentication and prevention of illegal memory access by malicious programs. Microsoft’s Secure Audio Path (SAP) technology is an example of a DRE Lower Layer technology. Finally, the Physical Layer consists of items related to the hardware on the client side. For example, this includes the file system where content and licenses are stored, along with the memory in which content

is rendered. Therefore, this layer is responsible for DRM technologies that are built directly into the hardware of the rendering device.

The all important middle layer in Figure 1 includes those technologies that are used to link DRM application-level functionalities with rights enforcement technologies. The specific technologies involved in the middle layer are rights expression and rights interpretation, hence the name REI Layer is used. The goal of this layer is to provide the minimal set of DRM services necessary to “connect” the upper and lower layers. The services provided by this layer are the minimal ones required by any real DRM application. That is, there must be rights in a DRM application, and this is the layer where they are expressed. Thus, the REI Layer serves the purpose of shielding the complexities of the service layers above from the intricacies of the rights enforcement layers below. The functionality placed at this layer includes the creation of licenses and protected content, and the delivery of rights-managed content and licenses. More specifically, the rights expression functionality is responsible for gathering information from upper layers regarding the rights that need to be supported in order to provide DRM functionality within a particular application. Rights expression languages (RELs) appear to be the preferred implementation approach [2, 6, 7]. Rights interpretation functionality is necessary since different rights can have different interpretations depending upon the environment, e.g., a “read-only” right has one interpretation in a general-purpose computing environment where printing or copying is readily supported, and another interpretation in a special-purpose environment printing is not supported. Furthermore, different RELs may have different sets of rights, and the rights interpretation functionality is responsible for any necessary translation of licenses, reformatting of rights, and repackaging of content and licenses. The MPEG-21 Rights Data Dictionary, which is intended to allow for the exchange of rights expressions between different DRM applications, is an example of an REI layer technology that performs rights interpretation [8].

In summary, a major advantage of considering interoperability from the perspective of a layered architecture is that it allows us to consider the levels within a system at which certain functionality should be provided, and then to address how the functionality between layers should interact in order to provide larger DRM capabilities. The communications that occur between layers, both within a single system and between two communicating systems, are the places where protocols can be defined and possibly standardized. Thus, they provide focal points for studying and addressing interoperability in DRM systems. In Figure 1 we define two types of protocols. First there are *intra-system* protocols (solid arrows) that deal with communication between the layers within a single system, and second there are the *inter-system* protocols (dashed arrows) that deal with the communication between peer layers on different systems. Furthermore, in Figure 1, we describe the generic responsibilities of the inter-system communications that occur between the various peer layers in the architecture. Although we have drawn these dashed arrows directly from one peer layer to another in this figure, in subsequent sections we will see that this communication can in fact involve intermediaries.

3. INTEROPERABILITY IN DRM

It seems that everyone has a notion of what interoperability means, which generally revolves around the idea of “things” working together. A slightly more formal definition related to technology is: “The ability of one technology to interact with another technology in order to implement some useful functionality”. It is possible to make nearly any two DRM technologies work together in a manner that satisfies this definition. Specifically, by building translation services, it is often possible to make one DRM regime work with a different DRM regime. At the current stage of development of DRM markets, this approach to interoperability makes sense. Furthermore, a number of recent papers have provided interesting solutions addressing interoperability at this level [10, 14, 15]. However, in order to facilitate the continued development of DRM markets, more detailed notions of interoperability of DRM technologies must be developed. In this sense, the real issue is not interoperability per se, but rather the level of interoperability that allows better DRM solutions to be created. Specifically, these will be DRM systems that are widely accepted by end users and thus support a sizable and therefore viable DRM marketplace. Before addressing this level of interoperability, let us briefly review problems identified by previous researchers, as well as some of the solutions they have developed.

3.1 Previous Work

Koene, et al. proposed three ways to approach interoperability in DRM systems [10]. The first of these is *full format interoperability* in which all participants use the same predetermined data representation, encoding, protection scheme, trust and key management, etc. The next level they proposed was *connected interoperability* in which online intermediaries exist for the purpose of transforming the functionality provided by DRM regime so that it can be used in a different DRM regime. The final level considered is *configuration-driven interoperability*. In this approach, it is assumed that the consumer’s device can dynamically configure itself in order to accommodate new formats and protocols. These authors argue that full format interoperability should be applied wherever possible and transformation services should be employed to bridge areas of non-interoperability. The key areas where they suggest applying full format interoperability include transport formats, compression, content key distribution, security services and trust management services. This approach seems realistic for restricted DRM markets, for example the authors mention that full format interoperability occurred with respect to DVDs; however, achieving this type of interoperability across the broader DRM market seems unlikely. Moreover, full format interoperability appears to conflict with the design principles proposed by Clark, et al. who note that rigid designs tend to break [3]. These authors instead advocate the idea of “design for choice”, where design is modularized along tussle boundaries. Furthermore, it is important to define standards in such a way that they provide enough room for DRM “players” to express their preferences [1]. This also seems difficult to support with full format interoperability.

A number of authors have further expanded the idea of connected interoperability. For example, Schmidt et al. proposed tackling connected interoperability using intermediated DRM [15]. The intermediary has to carry out four tasks to achieve interoperability, including: content and rights

re-formatting, data management, condition evaluation, and dynamical state evaluation. In addition, Safavi-Naini, et al. explored the idea of connected interoperability by proposing different translation architectures [14]. The requirements that need to be addressed by the translation architectures are transport format translation, license format translation and trust establishment with the rendering terminal. A common thread in all of the aforementioned lines of research is that they attack the DRM interoperability along process lines—the issues associated with this perspective were discussed in Section 1. Studying DRM from the perspective of processes (i.e., the use-case view) is very helpful in understanding what types of functionality need to be supported, and the types of communication that need to occur. That is, in this case we view DRM as a process involving a sequence of events that accomplishes some goal. Thus, the process-oriented approach is helpful in allowing others to understand what is being achieved within a particular DRM scenario. The architectural perspective, discussed in more detail next, allows us to consider DRM from a different viewpoint that we believe is useful in dealing with interoperability and standardization efforts. We note that this is analogous to use cases versus architectural views in the Unified Modeling Language (UML). In UML, software engineers find use-cases helpful in eliciting requirements and in understanding the process by which a system is used. The architecture view is used to show how the pieces of the system fit together, and where certain functionality should reside.

3.2 Future Efforts

Let us consider two extremes with respect to DRM interoperability. This first is a system in which a single vendor supplies all of the DRM functionality, from license management and content packaging, to client-side rights enforcement and the implementation of DRM business rules. Since all of the components are provided by a single vendor, the ability to produce a stable and reliable system is greatly enhanced. This is precisely the manner in which early DRM systems were developed—let us call them first generation DRM systems. The major problem with these systems was that they were inflexible. Thus, they were difficult to adapt, even slightly, in order to address different DRM business models. Furthermore, this inflexibility made it nearly impossible for other vendors to insert new technologies into the DRM value chain, which had the effect of stifling innovation and competition.

The other extreme would correspond to a DRM system that could be constructed by a systems integrator by easily piecing together the various components necessary to precisely suit a particular client’s business needs. Furthermore, in this case, end users would also be able to select from a variety of client-side DRM technologies that best suited their needs, and these technologies would seamlessly integrate with content providers’ delivery modalities.

The first extreme is where the DRM industry has already been, and the second is where we hope this industry will eventually go. Obviously, we cannot move from one extreme to the other in a single step, and there are likely to be at least a few more generations of DRM solutions before we approach the latter extreme. As we have previously mentioned, the connected interoperability types of DRM solutions that are already being developed constitute a logical evolution, and

we refer to them as second generation DRM systems. We expect subsequent generations will give vendors the ability to provide successively smaller pieces of an overall DRM system, and that these pieces will integrate better (i.e., more tightly and more easily) from one generation to the next. In addition, more specialized DRM systems, serving new and emerging needs, will develop as a result of more readily available DRM technologies. We envision that subsequent generations of DRM systems will begin to benefit from the development of standards, and the lessons learned from previous generations. Thus, to conclude this section, we provide a definition of DRM interoperability that takes these ideas into account. “Interoperability refers to the ability to use a technology developed by one party, that addresses a specific area of DRM functionality, by integrating though well-accepted interfaces with DRM technologies developed by other parties”. In most cases, in order to address a specific area of DRM functionality, the technology involved will have to be trusted by the other technologies that it integrates with. Thus, we see the development of mechanisms and protocols for establishing trust between independent DRM components as an important area that will need to be carefully considered. In the following section we use this definition, and the trust relationships it implies, to explore DRM interoperability with respect to a layered architecture.

4. INTEROPERABILITY & DRM LAYERS

The layers in the architecture presented in Section 2 correspond to one partitioning of the specific areas of DRM functionality described in the interoperability definition given at the end of the previous section. These are large areas of functionality, and we envision sublayers developing within each layer in order to address more specific functionality within a given layer. How these sublayers, and in fact the layers themselves, are ultimately decided will be determined by tussles in the DRM industry. Nevertheless, it is possible to perform some useful analyses by considering the layers as we have currently defined them. To do this, we will assume that the space inside a particular layer can be changed at will in order to address the particular functionality, but that the interfaces to adjacent and peer layers have been standardized. We will consider the effects of assuming that this type of interoperability exists at three different levels of the architecture shown in Figure 1.

Consider first the case of making the Physical Layer interoperable, as shown in Figure 2. In this and subsequent figures, the layer that is free for innovation is shown in white, and we also note the intra and inter-system communications that would need to be standardized in order for the layer to be made interoperable. In the case of Figure 2, a DRM system would be able to include different rendering devices, as long as they conformed to the interface with the rights enforcement technologies provided in the DRE Lower Layer. That is, the hardware will need to be able to directly support the hooks provided to allow for authentication of device drivers, integration with secure containers, and other device driver and kernel-level technologies. This corresponds to the solid arrow shown in the figure, and standardization of these hooks (or at least readily available documentation in the case of widely used operating systems) will be essential to support easy interoperability of rendering devices

The ability to make hardware interact with different operating systems through the use of operating system-specific

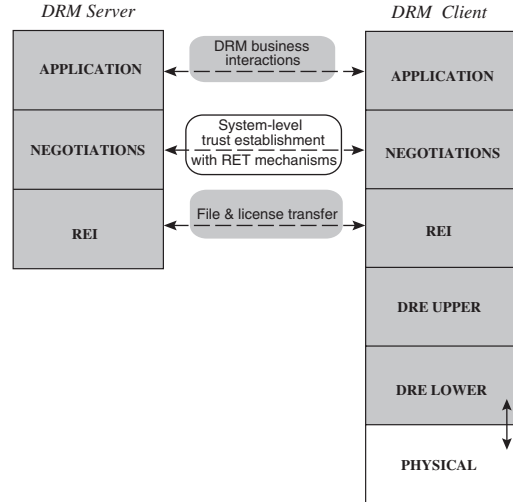


Figure 2: The communications within a layered framework that are necessary to achieve a low level of interoperability at the Physical Layer.

device drivers is very common today, and generally the device driver is provided by the same manufacturer that developed the hardware. From a DRM perspective, this situation can still be supported, but the security demands on the device driver must become more rigorous. To see why, consider the following. Because the establishment of trust between the hardware and the rest of the DRM system is now made explicit, we thus show inter-system communications between peer Negotiations Layers as something that needs to be standardized. The purpose of this interface is to allow the server side to make inquiries about the capabilities of the hardware, including the type and level of security it can provide. This can be used to allow the server side to dynamically determine the level of trust that it has in a particular client, and then make decisions about the type of content that will be supplied. For example, content may not be delivered at all, low-resolution or trial versions may be delivered, or high-resolution versions may be delivered. Other DRM-related decision may also be made according to the established trust. In any event, the inter-system trust establishment protocols will most likely require some standard and secure way to query the underlying hardware. We presume that this will involve the use of digital certificates in order to verify the credentials and authenticity of the hardware device, as well as protocols to express its capabilities. Obviously, these must be standardized if the device is to be used interoperably. That is, there must be some standard way for higher level DRM functionality to query the lower level DRM functionality.

We mentioned in Section 2 that Microsoft’s SAP is an example of a DRE Lower Layer technology. The SAP operates under varieties of the Windows operating system, and therefore supports computer hardware as a rendering device. Given the prevalence of Windows, the case can be made that the SAP provides a “standard” way for hardware to interface with DRE Lower Layer technology on a widely use operating system. Two things are missing in order to make this a more functional DRM environment. First, it

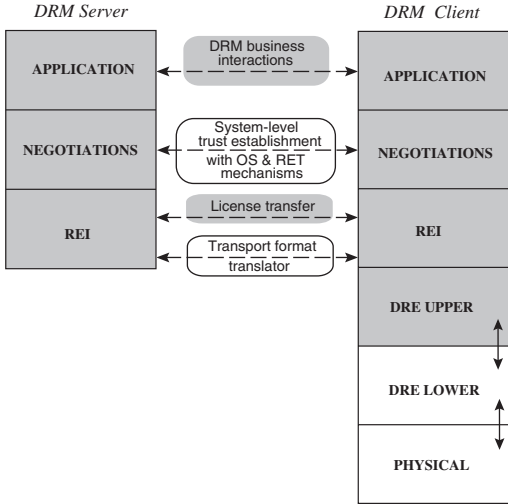


Figure 3: An intermediate degree of interoperability achieved at the DRE Lower Layer.

appears that only one level of security is provided, i.e., the SAP is either functioning or not. It would be useful if it could distinguish between the different levels of security that might be provided by the underlying hardware. In conjunction with this, protocols for establishing trust between the server-side and client-side hardware should be provided. By providing these two pieces, it would be possible for hardware to more easily interoperate within richer DRM environments, albeit we still regard this as a low level of interoperability.

Thus, we see that by freeing up space for innovation in the Physical Layer, interoperability protocols at two different interfaces in the overall DRM architecture need to be established. Nevertheless, this appears to be one of the most immediate ways to start providing interoperability. The degree of interoperability is low, but it is relatively easy to achieve since the issue of data handling does not arise. The next two levels of interoperability that we will discuss are more difficult to achieve, mainly because data handling becomes an issue, but they offer a higher degree of interoperability. In addition, it seems that they should arise naturally as a progression from the low-level form of interoperability we have just discussed.

Specifically, consider next the challenges to interoperability provided by freeing the DRE Lower Layer for innovation, as shown in Figure 3. Notice that we also show the Physical Layer as being free for innovation. This is an immediate consequence of standardizing the interface between the DRE Lower and Physical Layers that was required in order to completely free the DRM Lower Layer. Of course we could also see the situation where this interface is not standardized, and the DRE Lower and Physical Layers are then bound together as a whole. While this lowers the degree of interoperability, it simplifies matters since there would only be one interface on the client side that would need to be specified.

The DRE Lower Layer includes low-level rights enforcement functions implemented in the operating system of the device. This level of interoperability might mean for ex-

ample that a user is free to use the operating system and device of his choice to render content from different service providers. It also means that operating system or DRM vendors are free to implement innovative rights enforcement technologies at the DRE Lower Layer. One such example is the SAP just discussed, which includes kernel level authentication of all the components in the path to the sound card. Let us now consider the implications of making the DRE Lower and Physical Layers interoperable with rest of the system.

First there are compatibility issues. If interoperability is to be achieved at the operating system level, then there should be standardization in the way content is handled. Figure 3 shows the use of transport format translators operating at the REI Layer. Transport format compatibility methods include mechanisms by which compatibility in transport (i.e., file) formats is achieved between the sever and the client. This is related to the notions of full-format and connected interoperability as suggested by Koenen et al. [10]. However, the manner in which we are proposing the use of this type of functionality is more limited, i.e., we see it as a small piece in the set of protocols used to achieve interoperability. It should also be mentioned that a full-format interoperability approach is used in the Windows Media DRM 10 for devices system in which it is specified that the operating system of a device must support the ASF file format if it is to be compatible. Other compatibility issues may be specific to a particular vendor. We will discuss some of these in Section 5 when we consider the Windows Media DRM 10 for devices system.

Security is another issue that must be addressed. Ideally the DRM server would be able to query the client in order to determine the security of its operating system. Thus, there should be mechanisms by which the DRM server can verify if the client system satisfies certain security requirements. Protocols dealing with establishment of this type of system-level trust between the DRM server and the DRM client are implemented in the Negotiations Layer, as shown in Figure 3.

Finally, there needs to be standardization in the interface provided by the operating system to the DRM vendors' rights enforcement technologies. If these standards are well defined, then operating system vendors can follow them to make their system interoperate with the products of respective DRM vendors. For example, to provide interoperability at the operating system level, Windows Media DRM 10 for devices requires that the operating system running on a device implement functions to perform standard file I/O calls. These standards are shown as intra-system interoperability protocols between the DRE Upper and DRE Lower Layers in Figure 3. Thus it can be observed that the process of making the Physical Layer and the DRE Upper Layer interoperable imposes the the following requirements on the different layers of the system:

1. Introduction of protocols at the Negotiations Layer to establish system-level trust with devices and operating systems-level rights enforcement mechanisms.
2. A mechanism for the server to check the transport formats supported by the device operating system through the peer Negotiation Layer.
3. Introduction of transport format translation services which will operate at the REI Layer.

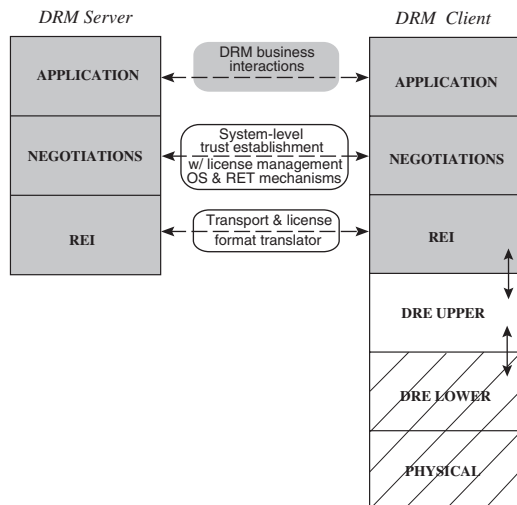


Figure 4: A high degree of interoperability achieved at the DRE Upper Layer. The hashed layers are viewed as a single block, but these could be separated in a more detailed analysis.

- Standardization in the interface provided by the operating system to the DRM vendors' rights enforcement technologies, introduced between the DRE Upper and DRE Lower Layers.

The next level of interoperability we consider is at the DRE Upper Layer, as shown in Figure 4. This is the high-level type of interoperability that is often envisioned when people use the term “interoperability” with respect to DRM. Interoperability at this level means that rights enforcement technologies can be developed independently. In this case, the free space created includes the rights enforcement mechanisms in the device and operating system, as well as higher level DRM mechanisms such as license management. This means that developers of these technologies can independently develop their products as long as they conform to the standards at the interface points.

Such a high level of interoperability creates a large number of compatibility and security issues. These compatibility issues include the ones discussed in the previous paragraphs, as well as new ones introduced as a result of making high-level rights enforcement mechanisms interoperable. One of the major issues in this area is license format compatibility. Taking a broader view, compatibility in terms of rights expression is one of the major areas of concern in DRM interoperability. A significant amount of research is concerned with rights expression interoperability [13, 14, 16]. Security issues also become complex when high-level rights enforcement is made interoperable. The DRM server must be sure that the high-level rights enforcement technologies running on the client system, along with the license management system, are capable of specifying and enforcing specific rights. The security and compatibility issues in the DRE Upper Layer impose added responsibilities in the Negotiations and the REI Layers. The Negotiations Layer must now also include a mechanism by which the DRM server can establish system-level trust with the high-level rights enforcement and license management mechanisms running on the client

system. For rights expression compatibility, the REI Layer must now also include protocols for license format translation if a connected interoperability type of approach is used, or agree on a standard REL. In terms of intra-system interoperability, the interface between the rights enforcement technologies running on the client system and the rest of the system must be well defined.

We have shown some of the possible ways of introducing interoperability into a layered DRM system—note that this type of analysis could also be applied to the sublayers within these layers, or for a DRM architecture where the layers are defined differently. The key point is that thinking about DRM as a layered system allows us to separate groups of DRM functionalities. Each layer in the layered framework represents such a group of DRM functions. Then it is possible to study in detail each of the layers separately and identify the mechanisms that should operate at the respective layers. Once these functions are identified, the effects of making a particular layer interoperable can be studied. In the next section we will overlay the Windows Media DRM 10 for devices system onto a layered framework, and then consider interoperability and standardization issues.

5. CASE STUDY: THE MICROSOFT DRM 10 ARCHITECTURE

The Windows Media Rights Manager provides DRM functions that include packaging of media files and protection of rights. A packaged media file is generally encrypted using a key. The key is then put in a license which is then distributed to legitimate users. Figure 5 shows the workflow for Windows Media Rights Manager [11]. The complete DRM process can be divided into five parts, namely, packaging, distribution, establishing a license server, license acquisition and playing a media file. Let us briefly look at the process of how DRM is handled by the Windows Media Rights Manager.

- **Packaging:** This is the process of encrypting the media file in order to create protected content. The encryption is generally accomplished by using a symmetric key encryption process. Windows Media DRM uses the method of encrypting the license and content separately. The protected media file also contains some additional metadata, which is included in the header of the file. The header, which is not encrypted, also contains the URL for license acquisition.
- **Distribution:** The protected media file is then distributed to the consumers. The distribution process is determined by the business model adopted by the distributor. The media can be distributed by means of a web server, a streaming server or optical media such as CDs and DVDs
- **Establishing a license server:** A license server is responsible for distributing licenses to legitimate customers. A license transaction takes place by first authenticating the users. Financial transactions are generally handled by the license server. A license contains information about usage rules and a key to encrypt the protected content.
- **License acquisition:** Microsoft uses four methods of license delivery, including non-silent, silent, non pre-delivered and pre-delivered. Silent delivery of a license

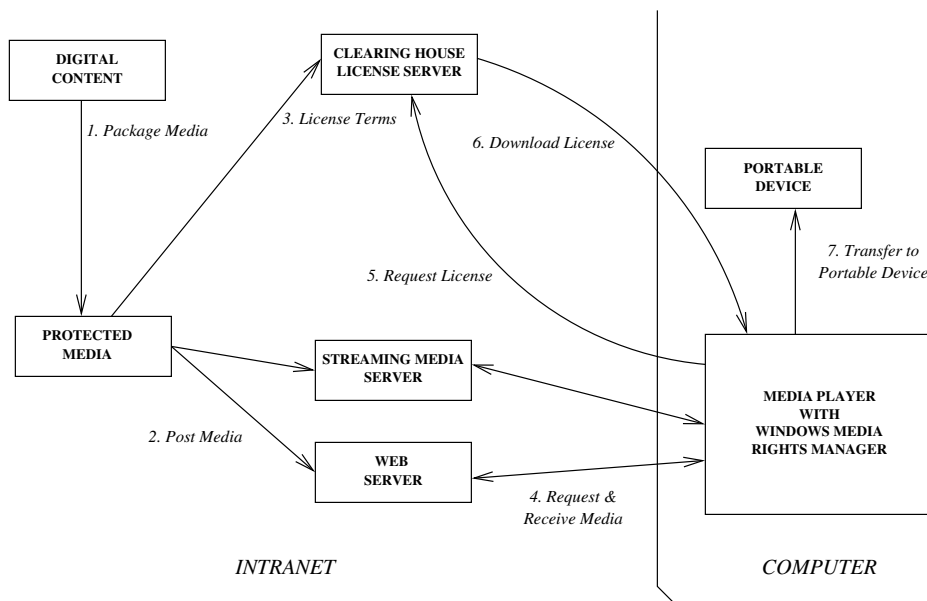


Figure 5: Workflow of the Windows Media Rights Manager.

is transparent to the user, whereas the other three methods require some action on the part of the user to acquire the license.

- Playing a media file: The Windows Media Rights Manager ensures that the media file is played in accordance with the rules specified in the license acquired by the consumer. Licenses are non-transferable and are bound to the rendering device, unless specified otherwise in the license.

Windows Media DRM 10 for portable devices provides a porting kit which allows portable devices to play content protected using the Windows Media Rights Management SDK. The porting kit provides ANSI C code to build functions which are necessary for managing content on portable devices. Licenses for portable devices do not allow the user to edit or copy content. Since the source code for portable devices is in ANSI C, it can be compiled on devices running non-Windows operating systems. A device has to meet certain requirements in order to play Windows Media protected content. Hardware requirements specify storage space requirements. Also, if time-bound licenses are to be supported, then device should have a tamper-proof real-time clock. A device should also securely store a unique device serial number. In addition, the following software requirements are specified [11, 12]:

- The operating system of the device must support standard file I/O calls.
- The operating system must be able to obtain the system time and convert between file time and system time, if the device supports time-bound licenses.
- If the device obtains licenses indirectly, then it must support the Microsoft Media Transport Protocol.
- The device should have an XML certificate, digitally signed by Microsoft.

- The device must be able to parse the ASF file format.

Microsoft DRM 10 for portable devices also specifies five levels of increasing security for audio and video output protection rights.

5.1 Overlaying Microsoft's DRM onto a Layered DRM Framework

The Windows Media Rights Manager flow can be overlaid onto the proposed DRM layered framework. In its simplest form, the selection of content by the user takes place at the Application Layer. The user can select the content from the Internet by using a standard web browser. The creation of protected media which includes encoding and encryption of data will take place at the REI Layer, which is also responsible for the distribution of the content. The webserver and the streaming media server will therefore operate at the Application and REI Layers.

The license server will operate at the three upper layers, i.e., the Application, Negotiations and REI Layers. The functions of the license server, which represent the method adopted by Microsoft for user authentication and performing monetary transactions, occur at the Application Layer. The functionality of license server that involves accepting requests for licenses operates at the Negotiations Layer. Device authentication is performed by the license server, where the rendering device is required to provide an XML certificate digitally signed by Microsoft. Protocols for authentication of devices and rendering platforms will be implemented in the Negotiations Layer. The piece of the license server that actually creates licenses and delivers them to the user machine operates at the REI Layer. The different license acquisition protocols implemented by Microsoft DRM 10, discussed earlier, actually represent various license services created in the Negotiations and Application Layers using the basic license creation services in the REI Layer.

Windows Media Rights Manager running on the client system will operate at all of the levels. The piece of the Rights

Manager which performs the tasks of obtaining licenses from the license server and providing information about the client platform operates in at Negotiations Layer. The part of the Rights Manager that actually receives the content and license from the server operates at the REI Layer.

Windows Media Rights Manager is also responsible for enforcing the rights included in the license on the client system. Management of licenses on the client system and controlling of access to the content will be performed in the DRE Upper Layer. Functions performed by the Rights Manager such as maintaining the play count and license store garbage collection will also be performed in the DRE Upper Layer. Operating system level rights enforcement mechanisms, such as device driver authentication and mechanisms to prevent memory snooping, operate at the DRE Lower Layer. The rendering device itself represents the Physical Layer. Rights enforcement mechanisms provided by the device, such as a secure clock and the SAP, are located at the DRE Lower and Physical Layers.

5.2 Standards, Interoperability & System-level Trust

Currently, interoperability provided by the Windows Media DRM 10 for devices is at the device operating system level. This means that Windows Media DRM 10 for portable devices can interoperate with any device running any operating system. In particular, Microsoft does not put any restrictions on the type of rendering device and its operating system, as long as they conform to the standards specified by Microsoft. This is one of the reasons why the porting kit provided by Microsoft is ANSI C compatible, so that devices running any operating system can actually use the content created using the Windows Media SDK. In our layered DRM framework, we note that an important point of interoperability is at the REI layer. One of the major reasons for this is that REI forms the minimal requirement for DRM systems and providing a minimal service at this layer will make it possible to separate DRM services from rights enforcement technologies. This is the type of interoperability envisioned by researchers. In Windows Media DRM system, interoperability is provided at the operating system level. Our layered framework for DRM systems places operating system functions in the DRE lower layer. Hence, Windows Media DRM 10 provides interoperability at the DRE lower layer.

As discussed previously, whenever a certain component of a system is made independent of rest of the system, there are associated standards, which must be developed in order to make the component work with rest of the systems. As long as the standards are followed, product manufacturers are free to introduce innovative technologies into their products. The same device and operating system can also download content from a service provider other than Microsoft, by running the RET of the other vendor.

It is important to consider what levels and what form of standardization is required by Windows Media DRM 10 to offer interoperability at the DRE lower layer. Broadly speaking, device and operating system manufacturers must conform to the standards specified by Microsoft, there must be a mechanism by which Microsoft verifies this. In addition, there must be a well-defined standard interface at the operating system level. Let us discuss how these goals are achieved in the current Microsoft DRM setting.

To support interoperability, devices have a minimal memory requirement of approximately 45 KB of RAM and 200 KB of disk space. To support transport compatibility, the device operating system must support the ASF file format. There is no need for license format compatibility because the licenses are managed by the software provided by Microsoft itself. These are the requirements specified by Microsoft for compatibility. Another requirement of interoperability is the trust with the device rendering the content. To achieve this, Microsoft defines five security levels for rendering devices in order to play content with varying levels of rights. Each rights level is associated with a security level that the device must conform to in order to exercise the corresponding right. These security levels are: unprotected, obfuscated, encrypted low, encrypted medium and encrypted high. In order to support time-bound licenses, the device should have a secure clock and the operating system must be able to get the system time and convert between file time and system time. These are the security and compatibility requirements that devices must satisfy in order to use content protected with Microsoft Windows Media DRM technology.

As mentioned earlier, the creation of an independent space at the Physical and DRE Lower Layers affects the other parts of the system in terms of the standards and protocols defined over the boundary of this space. The first requirement is that of a mechanism by which the license server is able to verify that the rendering device does indeed satisfy the security and compatibility requirements. Microsoft achieves this by requiring the device to have an XML certificate digitally signed by Microsoft. Whenever a license is issued, the Windows Media License Server requests a Microsoft-signed XML certificate for the device. Similarly, before giving a time-bound license to the device, the server ensures that the device has a tamper-proof clock. The request for a certificate and its verification constitute a process of establishing system-level trust with the user's system and these protocols are implemented in the Negotiations Layer. These are the inter-system interoperability protocols are shown in Figure 2. Thus it can be seen that making the Physical Layer independent imposes a requirement of introducing a certificate verification protocol at the Negotiations Layer.

To make different devices and their operating systems work with the Windows Media for DRM 10 Porting Kit, it is necessary to standardize the way the porting kit interacts with the operating system. Microsoft achieves this by specifying software requirements for the operating systems. They require the device's operating system to implement functions to perform standard file I/O calls such as close file, read file, write file, set file pointer, etc. These functions are called by the Microsoft DRM 10 for Portable Devices functions. The prototypes for these functions are provided by Microsoft in their documentation. These standards actually represent the intra-system interoperability protocols shown in Figure 3. Thus, to make the DRE Lower Layer independent, standard interfaces must be defined at its boundaries with adjacent layers. In this case, the Physical Layer (device) and the DRE Lower Layer (device operating system) as a whole are made to interoperate with rest of the system. Standards are therefore introduced at the boundary between the DRE Upper Layer, where the Microsoft DRM 10 for Portable Devices license management system operates, and the DRE Lower Layer, where the device operating system resides.

We have now looked at how Microsoft offers interoperability at the device and operating system levels with respect to DRM. Even though the Microsoft Media Rights management specification is flow based, it is possible to overlay it onto a layered framework. One advantage of thinking about DRM from the perspective of a layered system is that it helps to provide a “vocabulary” for talking about DRM systems, as we have just demonstrated. Furthermore, we have seen that it is possible to separate certain DRM functionalities in order to make them interoperable and then study the implications on other layers. This section provided one such detailed study. Highly interoperable DRM systems will be extremely complex, and studying such systems from the perspective of a layered framework can greatly facilitate analysis and design.

6. CONCLUSIONS

In this paper we considered DRM interoperability from the perspective of a layered architectural framework. We acknowledge the fact that the path interoperability ultimately takes in the DRM space will be determined not only by technological considerations, but also by current and emerging economic and legal issues. That is, as with other complex technologies, interoperability in DRM systems will evolve over a period of time as it is influenced by many factors. Architectures and standards will certainly play a role and will have to evolve in order to help guide the development of interoperable DRM systems. For this to happen, open standards must be developed along with an open architecture that can accommodate change. A layered architecture greatly facilitates this development by separating the different functionalities associated with a DRM system. This allows specific layers to be extracted so that the interactions between these layers can be considered in isolation. By reducing the complexity in this way, it makes it much more manageable to study the implications of interoperability in terms of the standards and protocols required to define interfaces and system-level trust.

That is, by thinking about DRM processes in terms of a layered system, it is possible to create spaces within DRM systems where innovation can be allowed. These “no standards zones” must exist in order for DRM vendors to create solutions that differentiate themselves from one another. Thus, these zones are necessary in order to foster competition and ultimately innovation in the DRM markets. We can also study the implications on the rest of the system in terms of standards and protocols which would be necessary to make a particular layer interoperate with other layers. A layered framework by its nature allows one to easily create these spaces. Furthermore, we have shown that as we try to make more of the lower layers independent, additional protocols must be implemented in the upper layers, and in particular at the Negotiations and REI Layers.

The current Microsoft DRM architecture for portable devices allows for interoperability in terms of rendering devices and the operating systems running on them. We have shown that the protocols and requirements specified by Microsoft to achieve this type of operating system-level independence is coherent with our discussions about interoperability at the DRE Lower Layer in terms of a layered framework. Thus, this approach to analyzing interoperability is helpful for understanding the protocols and standards that need to be defined in order to make certain components of existing DRM systems interoperable.

7. REFERENCES

- [1] H. Alverstrand. The role of the standards process in shaping the internet. *Proceeding of the IEEE*, 92(9):1371–1374, 2004.
- [2] C. N. Chong, R. Corin, S. Etalle, P. H. Hartel, W. Jonker, and Y. W. Law. Licensescript: A novel digital rights language and its semantics. In *Third International Conference on the Web Delivery of Music*, pages 122–129, Los Alamitos, CA, 2003.
- [3] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden. Tussle in cyberspace: Defining tomorrow’s Internet. In *SIGCOMM*, pages 347–356, Pittsburg, Pennsylvania, USA, Aug. 2002.
- [4] Coral consortium. www.coral-interop.org.
- [5] H. Goldstein, G. L. Heileman, M. D. Heileman, T. Nicolakis, C. E. Pizano, B. Prumo, and M. Webb. Protecting digital archives at the Greek Orthodox Archdiocese of America. In *Proceedings of the Third ACM Workshop on Digital Rights Management*, pages 13–26, Washington, DC, Oct. 2003.
- [6] R. Iannella. Open digital rights language (ODRL), Version 0.7. <http://odrl.net/ODRL-07.pdf>, Oct. 2000.
- [7] International Standards Organization. Information Technology – Multimedia Framework (MPEG-21) – Part 5: Rights Expression Language. ISO/IEC 21000-5:2004.
- [8] International Standards Organization. Information Technology – Multimedia Framework (MPEG-21) – Part 6: Rights Data Dictionary. ISO/IEC 21000-6:2004.
- [9] P. A. Jamkhedkar and G. L. Heileman. DRM as a layered system. In *Proceedings of the Fourth ACM Workshop on Digital Rights Management*, pages 11–21, Washington, DC, Oct. 2004.
- [10] R. H. Koenen, J. Lacy, M. MacKay, and S. Mitchell. The long march to interoperable digital rights management. *Proceedings of the IEEE*, 92(6):883–897, 2004.
- [11] Microsoft Corporation. *Architecture of Windows Media Rights Manager*, May 2004. www.microsoft.com/windows/windowsmedia/drm/.
- [12] Microsoft Corporation. *A Technical Overview of Windows Media DRM 10 for Devices*, Sept. 2004.
- [13] N. Rump. Can digital rights management be standardized? *IEEE Signal Processing Magazine*, 21(2):63–70, March 2004.
- [14] R. Safavi-Naini, N. P. Sheppard, and T. Uehara. Import/export in digital rights management. In *Proceedings of the Fourth ACM Workshop on Digital Rights Management*, pages 99–110, Washington, DC, Oct. 2004.
- [15] A. U. Schmidt, O. Tafreschi, and R. Wolf. Interoperability challenges for DRM systems. In *IFIP/GI Workshop on Virtual Goods*, Ilmenau, Germany, 2004. <http://virtualgoods.tu-ilmenau.de/2004/program.html>.
- [16] X. Wang. MPEG-21 rights expression language: Enabling interoperable digital rights management. *IEEE Multimedia*, 11(4):84–87, Oct./Dec. 2004.