DRM interoperability and reusability through a generic software architecture

By: Sam Michiels, Koen Buyens, Kristof Verslype, Wouter Joosen and Bart De Decker, Dept. of Computer Science, K.U.Leuven, Leuven, Belgium

Abstract: The domain of digital rights management (DRM) is currently lacking a generic software architecture that supports interoperability between 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 will be able to support the diversity of devices, users, and media, not to mention the wide variety of requirements concerning security, flexibility, and efficiency.

Keywords: technical analysis – DRMS, interoperability, software architectures

Challenges to DRM development

Systems that provide digital rights management (DRM) are highly complex and extensive: 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, which provide the context of this article: fragmentation of individual solutions, limited reuse of and interoperability between DRM systems, and lack of a DRM software architecture 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 DRM solutions (e.g. for music, for pictures, or for digital TV) and makes it very difficult to complete the complex and extensive DRM picture.

The second challenge, limited reuse and interoperability, is partly caused by in-house developed solutions that are incompatible with similar systems produced by other parties. Currently, for instance, an access service implemented by Apple cannot easily be reused in a Microsoft DRM system, even if both parties would like to do so. Although various DRM developers 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 services is crucial to allow (small scale) projects to contribute to the development of particular DRM services (Jamkhedkar and Heileman, 2004).

The third challenge, lack of a DRM software architecture, is typical for complex
software systems in evolution, and providing a software architecture is often a sign of growing maturity of the application domain. A software architecture can be seen as a generic structure that identifies the main service components and shows how they can interact. Having available such generic structure helps to evolve towards a complete set of interoperable DRM service components.

The challenges of integrating independent service components are well-recognized and are being addressed in other application domains than DRM, such as network communication, web services, or graphical user interfaces. The Internet architecture, for instance, convincingly demonstrates how a properly chosen software architecture can shape the evolution of a complex system across vast changes in technology, scale, and usage. 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.

**A layered DRM architecture as solution**

This article describes an academic study that argues for a layered DRM architecture that supports DRM developers in producing complete and interoperable systems (Michiels et al., 2005). The architecture is approached from both a functional and a security perspective. The functional 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 study presents a layered architecture and identifies the key DRM services of each layer. The main contribution of this study 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 proposed architecture in a nutshell**

The study presents the main system requirements from three application viewpoints: the content consumer’s, the content producer’s, and the content publisher’s. In addition, it identifies for each viewpoint the core functional services that are needed in a complete DRM system to provide this application-level functionality. In this way, seven key DRM services have been identified (see Figure 1): the Content Service (e.g. search for content), the License Service (e.g. issue licenses), the Access Service (e.g. authenticate consumers), the Tracking Service (e.g. produce usage statistics), the Payment Service (e.g. billing), the Import Service (e.g. submit content to the DRM system), and the Identification Service (e.g. reveal abusers). Next to functional services, the study identifies the hot spots in this architecture that require specific security services (such as authentication, confidentiality, and anonymity), and the cryptographic primitives needed to implement them (e.g. watermarks, digital signatures, certificates, and encryption). Remark that a single security service can be implemented by multiple cryptographic primitives depending on the requirements. For example, user authentication can be implemented by using digital signatures; yet, if user anonymity is required as well, other techniques such as zero-knowledge proofs must be used instead. The functional and security services are combined and presented in an architectural overview as shown in Figure 1. The model consists of a distributed view and perspectives from the side of the consumer, the producer, and the publisher, a layered architecture for each party, and identification of components in each layer.
Validation of the approach

By way of validation of the proposed approach, the study maps state-of-the-art DRM technologies onto the architecture and discusses how it supports the three main challenges formulated earlier. The validation is based on six DRM technologies on which technical information was publicly available: Windows Media DRM, Lightweight DRM, EMMS, Helix DRM, Aegis DRM, and the OMA specification.

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

<table>
<thead>
<tr>
<th>DRM tech/ Service</th>
<th>Content</th>
<th>License</th>
<th>Access</th>
<th>Tracking</th>
<th>Payment</th>
<th>Import</th>
<th>Identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMDRM</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>LWDRM</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EMMS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Helix</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Aegis</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OMA</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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 the introduction (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 of and interoperation between individual service components.
- Secondly, the study 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.
Thirdly, since judging from the study 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.

**DRM architecture and Internet architecture compared**

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. 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. In other words, although service implementations may vary (for example, the access service implementation on a mobile phone will clearly be totally different from a version for a desktop computer), the architecture in which a service component is embedded and the interfaces it provides to other service is always the same. Until today, many different companies and organizations extend the TCP/IP architecture with protocols for quality-of-service, wireless communication, 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 have to be aware 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. 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. In this case, applications should be able to fine-tune the underlying system by injecting specific policies. This is a generic problem that has already been explored in other application domains than DRM.

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 on the publisher's side to allow for content producers to associate with their content a wide variety of business policies. Yet, from a content consumer's perspective, this layer will typically 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.

**DRM Architecture and DMP compared**

The Digital Media Project (DMP website, 2005) proposes a similar approach to increase interoperability of DRM services. It defines users (e.g. consumers, producers, or publishers) as entities that perform so-called **primitive functions**, which represent the underlying DRM services that handle digital content. The primitive functions can be related to the functionality of the service components (e.g. revoke user), the rights expression and interpretation layer (e.g. represent rights expression), or the security components (e.g. represent key). The DRM architecture we have presented structures the domain by locating the set of primitive functions (components) in three layers: the service components layer, the rights expression and interpretation layer, and the security layer. Both approaches focus on interoperability by providing functions (components) with well-defined responsibilities.

**Bottom Line**

The presented model has confirmed the potential benefits of applying software architectures to inventory, analyze, and discuss research in this field, and has proven to be
useful to set the agenda for the future. 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. In our opinion, the next steps to be taken in this research field are (1) to refine the interaction interfaces of the identified service components, and (2) to apply and validate the proposed architecture in a case study to reveal additional issues driven by non-functional requirements (e.g. efficiency of content distribution, content personalization, or context awareness).

Sources
► The Digital Media Project (DMP) web site (2005): http://www.dmpf.org
► Michiels, Sam; Verslype, Kristof; Joosen, Wouter; De Decker, Bart (2005): Towards a Software Architecture for DRM. In: Safavi-Naini, Rei; Yung, Moti, (Eds.): Proceedings of the 5th ACM Workshop on Digital Rights Management (DRM'05), Alexandria, VA, USA. ACM Press, pp. 65-74

Acknowledgements: This article is a summary of (Michiels et al., 2005) and presents a study which brings us one step closer to a generic software architecture for DRM. Research for this article is part of the E-PAPER project, funded by the Interdisciplinary institute for BroadBand Technology (IBBT). The project is being carried out in collaboration with a consortium of companies: Philips, De Tijd, Belgacom, Hypervision and I-Merge. The authors are very grateful to Ann Heylighen for her valuable comments and for proof reading the text.

About the authors: Sam Michiels is a postdoc researcher at the Computer Science Department of the K.U.Leuven. He received his Ph.D. in computer science in 2003. His main interests include software technology and network middleware. Koen Buyens is a researcher in the Computer Science Department of the K.U.Leuven. His main interests include secure development of applications and middleware. Kristof Verslype is a Ph.D. student at the Computer Science Department of the K.U.Leuven. His main interests include secure software and anonymity. Wouter Joosen is professor at the computer science department of the K.U.Leuven. His research interests include software architectures for distributed systems, aspect-oriented and component-based software development, and secure software. He received his PhD in computer science from K.U. Leuven. Bart De Decker is professor at the Computer Science Department of the K.U.Leuven. He received his PhD in computer science from K.U. Leuven. His research interests include secure software, privacy, and anonymity.

Contact: sam.michiels@cs.kuleuven.be / bart.dedecker@cs.kuleuven.be / koen.buyens@cs.kuleuven.be / kristof.verslype@cs.kuleuven.be / wouter.joosen@cs.kuleuven.be

Status: first posted 25/01/06; licensed under Creative Commons

URL: http://www.indicare.org/tiki-read_article.php?articleId=167