HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

NONIUS: IMPLEMENTING A DRM SYSTEM

Ville Saarinen, Jari Anttila, Petri Lauronen, Olli Pitkänen

December 10, 2002

HIIT TECHNICAL REPORT 2002-3

NONIUS: IMPLEMENTING A DRM SYSTEM

Ville Saarinen, Jari Anttila, Petri Lauronen, Olli Pitkänen

Helsinki Institute for Information Technology HIIT
Tammasaarenkatu 3, Helsinki, Finland
PO BOX 9800
FIN-02015 HUT, Finland
http://www.hiit.fi

HIIT Technical Reports 2002-3
ISSN 1458-9451

Abstract

The paper describes experiences, ideas, and problems that were discovered while developing a DRM extension to X-Smiles XML browser as a part of Helsinki Institute for Information Technology's MobileIPR research project. The project succeeded in implementing a piece of software capable of demonstrating the most important DRM features. The most significant implemented features are restrictions related to an individual, time, and usage-counts.

The implementation is not perfect. Some of the interesting features that were intentionally left out are aspect and target constraints. The most obvious shortcomings are related to security: certificates and user data are stored in a user's hard disk in a human readable and modifiable format. Deliberately, there is no support for a secure encryption algorithm for protecting the content when it is transmitted over the network.

The most difficult tasks in implementing a DRM system are related to security and parsers. A secure DRM system requires the support of hardware devices. A DRM document parser depends profoundly on a flexible software architecture. Merging certificates and implementing an interface for creating certificates are also challenging. The ODRL specification, too, produces some challenges for implementation. The three most challenging features in ODRL specification are logical operators, documents' internal links to its elements, and the requirement that child elements may depend on ancestor elements' children.

Some general technical problems are discussed though largely left unanswered. How can we make secure software when the source code is available to anyone? In what layer should DRM be supported: application software, operating system, audio devices or even further? And how secure should the system be?

Many non-technical issues are also discussed. Creating a common, global DRM standard is a very hard political problem. If the immaterial products are sold in a global Internet shop, new questions will arise. Guarantees must be implemented somehow, and some solution must be found for the payment. Probably the biggest non-technical problem with DRM is how to sell it to the consumers. A DRM-featured version of a product is worse and maybe more expensive for a consumer. DRM will make technically impossible some actions that are allowed by copyright laws, like copying a product for personal use or reproducing a part of a product for educational purposes. The publishers must have an answer to the people who find out that they have lost rights they used to have.

But DRM is a radical change for publishers too. There is also a big question about the future role of the publishers since the main work is no longer producing and delivering physical products.

Contents

1. Introduction

Digital rights management (DRM) is a very interesting and timely topic. Traditional rights management is quite different than immaterial rights management, and immaterial products, like digitally stored pictures, are much easier to copy than traditional ones. In digital world it is very easy to make a perfect copy of an original product. A consumer may not even think the copyright issues at all when saving a picture from a browser to a hard disk. However, the consumer should consider copyright issues. One problem is that we cannot distinguish copies and originals in immaterial world, since a perfect copy is really a perfect copy.

The concept of digital rights management is ambiguous. In this report, DRM refers to technologies that control copyright in the digital world. The basic idea is that the content is encrypted and it can be safely distributed. But decrypting and using the product is allowed only if the consumer has a suitable certificate for the product. The certificate gives the right to use the content. A certificate can have various restrictions so that digital content can be published in totally new ways. For example, a consumer can buy instances of use or a limited usage-time for some product. In general, such restrictions are not feasible in the material world.

Helsinki Institute for Information Technology (HIIT) is the joint research institute of University of Helsinki and Helsinki University of Technology. It conducts internationally high-level strategic research in information technology, especially in areas where Finnish IT industry has or may reach a significant global role. HIIT works in close co-operation with universities and industry, aiming to improve the contents, visibility, and impact of Finnish IT research to benefit the competitiveness of Finnish IT industry and the development of the Finnish information society.

HIIT's MobileIPR research project studies problems related to the rights management of information products on the mobile Internet from legal, technical and economic perspectives. The project started in 2000 and it is planned to be continued until the end of 2003. The project leader is Dr. Jukka Kemppinen and the project manager is Olli Pitkänen. Mikko Välimäki, Ville Oksanen, and Tommo Reti are also working on the project. Significant part of project work is carried out at the University of California at Berkeley. The project is funded by Tekes – the National Technology Agency of Finland, Elisa Communications, Nokia, Sonera, Finnish Broadcasting Company (YLE), and a group of Finnish law firms.

In connection with MobileIPR, a student group called *Nonius* made a DRM extension to X-Smiles XML-browser. Chapter four gives more information of this two-semester project. This document contains some ideas and describes some difficult areas that we found in the Nonius-project.

The second chapter will describe in more detail what DRM really is, and what kinds of solutions are available. The third chapter contains technical details about our solution and the forth chapter is about the organization of Nonius project. Chapters from five to seven tell about problems in DRM and some ideas how to prevent them. Chapter eight describes how this technique involves different parties, like customers, retailers and content owners. And finally there are the conclusions. Terms and sources are in appendix one and two respectively.

2. Digital Rights Management

2.1 Special Issues with Immaterial Products

Industrial products can be divided into two categories: physical (or material) products and immaterial products. Immaterial products like books, music and movies are only information, and they can be presented in digital form. Today these products also include a physical medium like the paper of the book or the disk where the music is stored, but through a computer network it is possible to sell only the data itself. The fact that a product can be perfectly presented in the form of bits is both an advantage and the source of problems. The distribution of a digital product is very easy and cheap. Buyers can download the data using their network connections - no transport of physical medium is required. This saves the content producer from a lot of distribution costs, and also makes it very easy to advertise the product by giving out free samples. Free samples are a very good way to give potential buyers an idea about what the product is like, and since the distribution of a sample is practically free for the content producer, it can be used on a large scale. [9]

There is also a negative effect on the product being only information. When the product is given to a buyer, it is hard to avoid users making as many perfect copies of the data as they desire, and also do anything he likes with the copies, including giving or even selling them to other people. This can lead to massive piratism by average users if the copying is easy enough that anyone can do it. Also the job of professional pirates will become a lot easier. Most of the costs come from the creating of the product – when it is ready, it does not cost almost anything to make copies of it and spread them throughout the world. The pirates will have a much greater profit percentage for selling the product since they don't have to participate the creation costs of the product, and therefore will get a big price advantage.

2.2 What Is DRM

It is fundamentally impossible to prevent copying of digital data. The data is always just a stream of bits - ones and zeros. If you can read it, you can copy it. However, it is not necessary to make the copying impossible, a lot can be gained by only making it hard enough. A pirated version of the product may be cheaper than the original one, but if it requires hours of work to actually make it work, most of the consumers will select the easier way and buy the original product. [11]

The Digital Rights Management (DRM) scheme doesn't try to prevent the copying itself, but the actual usage of the pirated product. It separates the data from the right to view the data. The data itself, which might be a book, movie, music album or anything like that, can be copied freely, and the system relies on the fact that the data bits are transferred from one user to another. This takes part of the responsibility of data distribution away from the content producer or publisher to the consumers themselves. If the data is distributed between friends there is not so much need for a central data distribution server, which is expensive to operate and keep reliable. The content is distributed everywhere, but it cannot be viewed without a special certificate. The publisher must only operate a certificate server, which allows the consumers to buy the certificates required for viewing the content. The amount of data is dramatically smaller than if the publisher had to spread also the content

itself to all buyers. Also, the certificates must be somehow user or hardware-specific so that a certificate cannot be copied for use of multiple users.

Besides making copying harder, a good DRM system has also other functions that create possibilities for completely new business models. For example, a DRM certificate may limit the user's right to view the document so that he can only view it one or two times. This is very practical for giving out free samples of the product – user can see what the content is like but if he wants to continue using it he has to buy a new certificate. Other limitations that are good for free samples are limited period of time (for example the product can only be accessed during June 2002) or limited cumulative usage time (the user can view the content for one hour). The DRM system may also limit the usage by user, hardware, physical location or many other rules. [18, 19]

2.3 How to Implement a DRM System

The digital content must be protected so that it cannot be viewed without a valid certificate. This requires the use of strong encryption in a way that the certificate includes a secret key needed to decrypt the content data. The encryption algorithm must be strong enough that it won't be broken easily, but the data encryption is not the weakest link of a certificate-based system. The problem is that in order to be viewed, the encrypted data has to be decrypted by the legitimate customer. In an open environment like a standard personal computer the user will be able to do anything he wishes, including copying and spreading of the decrypted data. This problem must be dealt with by applying regulations in either software or hardware. [20]

The easiest way to control the copying of decrypted data is to create a special software browser or player for the material, so that it will decrypt and play the data on an analog device like a monitor or loudspeaker system, but it will not write the decrypted data anywhere in digital form, so that the user could make digital copies of the content. Analog outputs are not that big a problem, since analog copy is never the same quality as the original, unlike the perfect digital copies. If the quality is weakened even a little, it will make the copy less valuable than the original product and therefore not as big threat to the sales of the original product as if it was an exactly perfect copy. The special software browser is currently the most commonly used DRM implementation. It has been commercially used, for example, in the Adobe eBook reader. [18, 19, 21]

Software regulations are however usually easy to circumvent. By special debugger and disassembler software, crackers will be able to monitor the execution of the software player and get their hands on the decrypted content data. A hardware implementation of the copy prevention mechanism is a lot harder to break. A special hardware device that decrypts the data and only gives it out through an analog output is very hard to break since the decrypted digital content only exists inside the device. It will require a lot of resources and know-how to break a hardware-based copy prevention system.

A hardware copy prevention system does not necessary have to be a separate player device, it can also be done in a standard personal computer at some level. Some big computer industry companies like Intel and IBM have made a proposal to include special digital rights management schemes in computer hard drives and operating systems that would make it possible for a file to be marked as copyrighted material (by watermarking or other similar techniques), and all copy operations for the file would then be denied by the operating system. This proposal is known as CPRM (Content Protection for Recordable

Media). This kind of arrangement is problematic since all hard drive manufacturers and all operating system manufacturers would have to co-operate in order to effectively force the restrictions. Making all major manufacturers work together is a task close to impossible, especially since it would require a lot of new technology to be developed and the hardware manufacturers do not really have much to gain in the decreased piracy problem. Also open systems like the open-source operating systems are a problem. If the user has full source code for the system, he can modify it and bypass the restrictions. [12]

There is also a proposal for applying the digital rights management scheme closer to the user, in the output devices which make the actual conversion of digital data to some analog signal like sound or picture (monitors, loudspeakers, and other similar devices). Since the encrypted content data would be decrypted for example inside a monitor, there is no way a user could get his hands on the decrypted data with software. Only minimum co-operation would be required from the operating system, but there are still some disadvantages. If the content data is in encrypted form all the way to the output device, only an output device that supports the technology can show it. This would require all end-users to update their hardware, which is not an easy process. The new hardware will be a significant cost to customers, and a lot of opposition would be encountered. Even at best, it will take years before a significant number of potential buyers have the hardware needed to view the product.

The digital rights management for output devices can also be done in a more backwards-compatible way. The data itself doesn't have to be encrypted, the content producer only needs to mark it copyrighted through a watermarking scheme like the SDMI (Secure Digital Music Initiative) watermarking. All compliant output devices would then detect the watermark and apply whatever restrictions the content producer wants for the use of the material. Using this kind of approach, people would still be able to view the content without authorization using old output devices, but since any digital device gets old pretty quick the older devices would be pretty much history in five or ten years after all new devices started to include the required restrictions. The problem is that all new devices would have to include the digital rights management features in order to make them actually work, and it is not easy to include a restriction in all output devices by different manufacturers. There is certainly consumer demand for a player without the restrictions. So unless it is illegal, some manufacturer will surely create a player without the limits and make good profits.

Creating a completely unbreakable copy prevention system is impossible. The user, or at least some component in the user's hardware must have a proper key to decrypt the digital content in order to actually view it. And if digital data can be read and viewed, it can also be copied. If the security measures are done in software, it is easy for an experienced hacker to break the software and get the decrypted content data. And even if the security measures are done by hardware, some hacker or a professional pirate will have enough resources to analyze how the hardware works and modify it to bypass the security measures. But the professional pirates are not such a big problem as long as the average user pays for the product.

2.4 Rights Description Languages

In the technical implementation of a DRM system, some kind of language is needed for describing the limitations and other restrictions in the product. In our project, we added DRM

restrictions to an XML browser, and were instructed to use some existing XML digital rights description language. There are a lot of competing language standards, and we selected three of them for a closer look: XMCL (eXtensible Media Commerce Language), XrML (eXtensible rights Markup Language) and ODRL (Open Digital Rights Language).

All the candidates had enough DRM restrictions for our needs. They all have a lot of powerful supporters in the technology and content producing industry: XMCL is backed up by for example Adobe, America Online, British Telecom, EMI Recorded Music, IBM, Napster, RealNetworks, Sony Pictures Digital Entertainment and Sun Microsystems. The list of XrML includes Adobe Systems, Hewlett-Packard, Microsoft, and Time Warner Trade Publishing. Nokia and many smaller firms support the ODRL standard. [6, 7, 8]

The industry support was however not important for us since we had a more academic perspective to the problem. We finally decided to support the ODRL standard, mainly because it is an open standard and therefore more flexible than the other candidates that are created mostly for the needs of their creator companies (XMCL is developed by RealNetworks and XrML by ContentGuard. [3, 4]

Later, after we had already made our choice, XrML achieved a considerable victory over the others, winning the competition for rights expression language for MPEG-21 media distribution standard. [4, 5]. An overview of ODRL is presented in chapter 4.

2.5 The Role of Legislation

Legislation has an important role in enforcing the new restrictions. Most DRM technologies depend on the support of all hardware or software manufacturers, and it's pretty much impossible to get everybody to support the same standard unless it's required by law.

There is currently a draft for legislation called SSSCA (Secure Systems Standard and Certification Act) [15] in the United States that would force all manufacturers of digital devices by law to include the necessary digital rights management features in all digital devices. The DRM standard to be used will be selected by manufacturers and content producing industry, or if they fail to accept a common solution, the governmental authorities will choose a standard to be used. Forcing the DRM schemes by governmental regulation might be the only way to actually get them to work, but the SSSCA is still only a draft, and even if it were accepted in the United States, similar laws would have to be enacted throughout the world to actually make them work. [13, 14, 15]

Another role for legislation in the DRM scheme is to stop spreading information and tools that can be used to break the copy protection mechanisms. It is not a problem if a single hacker can break the protections and watch a movie for free, but if he makes a tool for breaking the protection and the tool is so easy to use that an average user can break the protection mechanisms with it, it will be a big problem for the content producing industry. Professional pirates will get the tools even if they are illegal, but the problem is dramatically decreased if the average users cannot break the protections. [11, 17]

There are currently laws that render the copy protection breaking tools illegal in both the United States and the European Union. The first law to ban break-in tools was the Digital Millennium Copyright Act in the United States, and the European Union followed with the EU Copyright Directive (2001/29/EC) in May 2001. The EU Copyright Directive forbids the manufacturing, commercial use, selling, advertising etc. of the circumvention tools. The

DMCA is a bit stricter; it also forbids the publishing of information of any such tools. However, similar laws must be enacted world wide in order to make them work. [10, 16]

3. Organization of the Nonius Project

The Nonius project started in September 2001 and was finished in April 2002. It was done as part of the Software Project course at Helsinki University of Technology. The goal of the project was to learn about working together in a large software project and create a product for the needs of our customer. Our customer (Helsinki Institute for Information Technology) wanted us to create an add-on-component to the X-Smiles XML browser that would enable DRM support in the browser.

Our project team included seven members. We divided the work so that 1-2 persons had the responsibility over project management and quality, 3-4 persons were responsible for the actual coding of the software, and 2 persons for the testing of the product. Our resources were limited so that each person should do a maximum of 200 hours of work at the course.

The course was mostly a success. We created the DRMTool, an extension to X-Smiles XML browser, which gives the browser support for DRM using ODRL digital rights expression language. We did not include all ODRL features in the product because of our limited time resources, but everything we and our customer considered as key features of ODRL are supported. We demonstrated the product to our customers and other people connected to the DRM research, and this document is based mostly on our experiences during the Nonius project.

We finished the project with the resources we were given. The project required a total number of 1218 working hours, which was well below our limit of 1400 hours. No person had to do over 200 hours of work. The organization, quality and other overhead processed required 512 hours of work, documentation 236 hours, coding the actual software 257 hours and testing the product 146 hours. The large number of overhead is explained mainly by the requirements of the course at HUT – the course was about learning the organization of big software projects so the organizing process was quite heavy. [22]

During the project we created 6229 lines of code: 4114 lines of program code and 2115 lines of comments. A lot of commentation was required since we won't ourselves be responsible for the improving of the product after the course. We also estimated how much the project would cost in a real commercial environment. We used 60 hours of our customer's time (estimated value 15900 euros), 1200 hours of our own time (estimated value 40800 euros), and the project would require computer equipment and software that would cost an approximate value of 23100 euros. So, the total cost of the project in a commercial environment would be about 80000 euros. [22]

4. Introduction to the DRMTool

This chapter introduces the insides of the result of the project, DRMTool. The DRM features that we implemented are discussed along with some alternative solutions and reasoning for the choices that were made. A short introduction to the X-Smiles XML browser, how DRMTool interacts with X-Smiles and the architecture of DRMTool is presented. Specifically, experiences and observations that were gathered during the implementation are not

analyzed in this chapter, but in a later chapter. A reader not interested in technical issues may move on to the next chapter.

X-Smiles is an open source project, a pure XML browser implemented in Java. It currently supports several XML technologies such as SMIL, XSL and XForms [1]. X-Smiles is continually developed further, for this project we chose to get a separate branch of the browser as base for DRMTool development. The version of X-Smiles used in the DRMTool is 0.5. Majority of the functionality of DRMTool was implemented as separate modules, only a couple of changes to the GUI code of X-Smiles itself were made. The development environment was running JDK 1.3.1 on Microsoft's Windows 2000 and Windows XP operating systems.

X-Smiles offers an easy-to-use interface for connecting tailored modules for handling different types of XML documents. These modules are called MLFCs (Markup Language Functional Components) in the X-Smiles terminology. Basically, to connect an MLFC to X-Smiles, one first writes the implementing class (a subclass of fi.hut.tml.xsmiles.mlfc.MLFC) and support classes for whatever functionality is wanted. After this, a mapping between the wanted XML document type (namespace) and the name of the implementing MLFC class is added in the X-Smiles configuration file. After this, the browser core automatically calls the module when that type of XML document is requested. In the case of DRMTool, an MLFC for handling ODRL documents was written.

The architecture of DRMTool consists of the following components:

- DRMMLFC, the interface to X-Smiles
- DRMStore, a database for DRM agreements (i.e. certificates)
- HandlerDRMOffer, responsible for displaying the asset
- HandlerDRMAgreement, responsible for displaying/saving an agreement in the database
- AssetDecode, responsible for enforcing the correct handling of rights and possibly decryption
- SecurityModule, implements security functions

What comes to implementation of ODRL, the most interesting module is AssetDecode, which essentially consists of the parser of ODRL documents. The parser will be discussed next, whereas the inner workings of the other modules are out of scope of this document.

One aim in the design of the DRM parser was that it should be highly modular. The architecture should allow a change of the underlying DRM language by implementing a common set of interfaces. The interfaces were designed using ODRL as a basis: the conceptual model for the architecture follows very closely what is presented in the ODRL 1.0 specification [2].

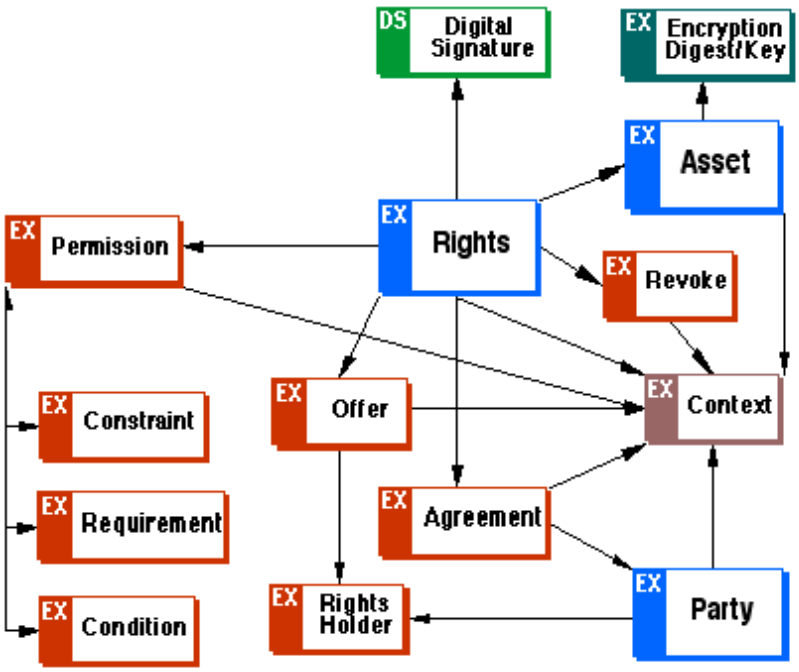

Figure: concepts modeled by ODRL in a broad context [2]

The figure above presents the most important concepts of ODRL and is explained in more detail in the ODRL 1.0 specification. These are more or less common in all DRM scenarios. An Asset is the product or work that DRM aims to protect. An Agreement is a contract between some parties that defines the terms under which a party may use the Asset. An Agreement is essentially what is called a certificate in many contexts. An Offer is like an Agreement with the difference that it doesn't specify any party that agrees to play by the rules. This makes it unusable as verifying DRM requirements, but an Offer may be freely distributed for viewing what kind of terms the use of an Asset would have.

In the implementation, each conceptual class was modeled as a Java interface with accessor methods for reading business logic related data and some standardized methods for constructing objects from general XML elements. After we had designed the interfaces, we wrote implementing classes with ODRL specific parser functionality to achieve the main goal of the project (a meaningful demo).

The concepts of the DRM language are presented in real world inside XML documents, as XML elements. The basic idea in the parser architecture was that each class knows what kind of term or restriction it represents in the DRM scenario. Each class represents one XML element in the DRM language, and knows what kind of information it should extract from the XML element when constructed. Further, each class needs to know what elements might be related to it and knows to pass the request for verifying a term to the relating child elements if it does not know how to verify it directly itself. An additional requirement was that the certificates should be viewed in a more humanly understandable way than XML: for this purpose, each class was assigned the responsibility to be able to represent the information it contains in a human language. These common features were implemented in one common interface, which was extended by all the other interfaces. The responsibility to create objects was given to separate factory classes when this was feasible, thus enabling easy development and rollover of new functions.

4.1 Assumptions and alternative solutions

At first, we planned that the opening of any Asset in the browser would trigger the validation of DRM rules that were related to it. This immediately proved impossible, since X-Smiles only allows us to map the execution of our module with certain XML documents, not all. Of course, this obstacle would have been possible to circumvent via hacking the X-Smiles code so that it would always call our module first. One of the goals of the project was to be able to implement DRM functionality with as few changes in the X-Smiles itself as possible, so this solution was rejected. Another problem is that each Asset needs to be uniquely identified, and a plain URL isn't enough for this purpose.

As a solution to this we thought of defining and implementing an own XML based language for DRMTool purposes. The idea was that the new language would contain metadata and execution instructions for opening ODRL documents. After we examined ODRL closer, we found out that an Offer document contains all the information necessary to implement the same functionality. Only four assumptions needed to be made, and they were quite reasonable, so this is the solution we chose.

The following assumptions were made:

1. The Asset is always accessible via a URL (which may be local as long as X-Smiles supports it) and points to an XML document that X-Smiles can display.
2. The displaying of an Asset is initiated by opening an Offer document
3. The Offer needs to contain a URL pointing to the Asset
4. The Offer needs to contain a unique identifier for the Asset

Now, once an ODRL Offer document is opened (by assumption 2, any asset protected by DRM means is opened this way), our module is loaded. Then it may identify the Asset that is being opened (by assumption 3). After this, it can search for the corresponding Agreement in the database. If an Agreement is found, the DRM terms are validated and, if everything is acceptable, the module can ask X-Smiles to open the Asset (by assumption 1) pointed to by the URL in the Offer (by assumption 3).

4.2 DRM features implemented in DRMTool

ODRL defines lots of DRM capabilities. The purpose of this project was not to implement them all. In the requirements capture phase, the most important capabilities and functionalities were chosen and prioritized. DRMTool has some general functionality not directly related to ODRL, such as user data management features or saving DRM documents in the user's system which are not discussed further here. The implemented features related to ODRL are:

- viewing the contents of a DRM document (Offer or Agreement) in a human understandable way
- restricted time of validity of certificate
- restricted individual (only specific user may use an Asset)
- restricted software (only specific instance of X-Smiles browser may use an Asset)
- restricted instances of use
- restricted accumulated time of use

- restricted geographical area
- any combination of these (AND –operator)

In order to implement these, a lot of the basic concepts defined in ODRL needed to be implemented, such as a general Permission, Constraint, Context and Party but these are not seen by the end user as clearly as the functionalities listed above. The chosen ones are the most commonly used when speaking about DRM, which was the main motivation for the choice.

As a contrast, some of the most interesting features defined in ODRL left out include:

- Different usage types (such as view, print, modify, sell): we implemented only display, since this is currently the only thing that X-Smiles supports
- Super distribution (X-Smiles doesn't support any such methods)
- Device constraints other than software (printer, CPU, screen etc.)
- Aspect constraints such as quality or watermark.
- Target constraints such as the purpose of use.
- Combination of constraints using OR and NOT operators.
- Secure encryption methods, such as public key cryptography. Security was not a focus area of this project.

5. Experiences from implementation of DRMTool

This chapter discusses the problems we faced and experiences we gathered during the implementation of DRMTool. The problems are viewed from a technical perspective. Some are specific to the technical environment, some relate to the architecture we chose and some derive from the ODRL specification.

5.1 Experiences from the programming process

The project team had not worked on DRM before. The project knew of no open source projects that could have been studied and asked for guidelines for DRM implementation. One goal of the project team was to design a scalable and modular architecture for the parser, one that might even allow a change of DRM language if necessary. Scalability was important for the process point of view: functionality was added in iterations and we had planned only two or three weeks for the implementation per iteration.

The team found that the ODRL specification was well written and relatively easy to understand. From it, business logic aspects of the architecture were quite easy to design and didn't require many modifications once designed.

On the other hand, other parts of the architecture such as the parser framework and utility classes were quite dynamic and hard to freeze. The inexperience of the project team in implementing parsers that create object structure from XML documents lead to that these changed quite a bit during the process. For example, the requirement to view a DRM document in a human understandable way lead to a new method in the common DRMComponent interface, and the corresponding implementation needed to be written in every parser class. Needs for new XML parsing utility functions were discovered one by one, such as a method for getting direct XML child elements with the given name, requiring at

least n such children to be found. Luckily these had been defined in a separate class that was easy to grow when new needs arose.

We didn't get to really test the modularity of the architecture by changing the DRM language used. Basically this should be a straightforward task as long as the new DRM language is XML based and its concepts can be unambiguously mapped to ODRL's concepts. Interpreting the interfaces correctly might require some knowledge of ODRL, though.

5.2 Shortcomings in the implementation

The implementation in itself is far from perfect: certificates are stored in the user's hard drive in a human readable and modifiable (assuming the user knows ODRL even a little bit) format. All user related data is there, too. This could be done in a more secure way by sealing the certificates. For example hash values could be used for verifying integrity.

Another problem comes from the implementation of the software constraint. For this purpose, DRMTool chooses a random device id (a DOI) when it is first needed. This, along with other user data, is stored in user's hard disk. If X-Smiles needs to be reinstalled, the device ID vanishes and certificates containing software constraint will not work, since the ID has changed. This could be implemented by storing the software ID in, for example, on a remote server, where it could be retrieved during reinstallation.

The only "encryption" method supported is gzip. The only purpose for this implementation is to make the documents unreadable for humans. Public key cryptography could be used, but this really should be supported by the A/V hardware for reasonable security.

Finally, the source code is available to anyone who knows Java. The open source problem is further discussed in another chapter.

These shortcomings are decisions that were deliberately made during the project implementation. For the most part, these do not lessen the product's usability for demonstrating DRM capabilities.

Two X-Smiles specific problems are related to the implementation of accumulated time constraint. Firstly, there is no way of stopping the displaying of a document once the viewing has started. This means that if the certificate allows for 10 seconds of accumulated time, the user may view the document for as long as he likes if he never shuts the browser or loads a different page. A workaround could be implemented, for example, with a specific timer thread that tells the X-Smiles to load a blank page after a certain period of time, but we did not consider this as good design and lasting a solution, and it was not implemented. Also, the DRMTool module starts the timer that counts accumulated time before it tells X-Smiles to load the asset, the timer ticks away while the document is loaded. This might take a long time over slow connections and/or if the document is big, which irritates users. Again, X-Smiles does not have an "afterLoad" event that could trigger the accumulated timer that could be used for this purpose.

One X-Smiles bug is that the document loading function that DRMTool calls is not able to display other than SMIL documents. This is extremely odd, since X-Smiles itself knows how to display many different types of XML documents, and the execution should follow the same route no matter who initiates the loading. This doesn't affect the usability of DRMTool for demonstration purposes: a SMIL document is the one that people most easily relate to DRM with its music, image, video and sound effect capabilities.

5.3 Difficult areas

ODRL specification states that if the software comes across some DRM restriction that it doesn't know how to validate, it shouldn't allow the action. With the DRMTool architecture, this is a challenging area. This has been implemented where possible. Easy places to add this kind of functionality are the factory classes. If a factory class gets an XML element that it has not got a mapping for, a DRMParseException with an explanatory message is thrown. When such an exception is thrown, the user will not be able to perform the action and an error message is presented. DRMParseExceptions are thrown also if there is something wrong with the data parsed (wrong type, missing information etc.). But, since DRMTool uses the XML DOM approach for parsing, when an object is being constructed from an XML element, it asks the DOM object for the information it needs. If there is an extra element in a place that some factory does not handle, it goes unnoticed. This violates the ODRL specification. A simple solution would be to write a custom XML schema file for the implemented subset of ODRL, and use that to verify documents. This was not implemented in the project.

DRMTool currently supports only one certificate per asset. No merging of certificates was implemented. This means that if the user buys another certificate for an asset, the latter automatically overwrites the previous one. Merging of certificates is no trivial task, especially if the certificates are complicated. Implementing it in software is very difficult. The biggest problem is that when doing a merge of two DRM documents one would need to consider the semantics of the XML elements. The merging problem has been studied for quite a long time in software configuration management, especially when implementing version control software. There exists dozens of algorithms for textual based merge (line per line comparison), and some for syntactic or semantic merge for a specific programming language [3], but studying and implementing one of these for ODRL would be a project in itself.

One challenging feature was adding support for incremental requirements. These are requirements that are not necessarily known when a user wants to use an asset but may come up later disallowing the use of the asset. We designed the framework for adding such requirements and implemented one: pay per view. As the result of the query to open a document passes on through the object structure, each DRM component may add its own additional constraint to the result. The additional constraints are implemented as a class that knows how to merge itself with another object of the same class. These objects gather together like streams into a river, and in the end, if there are additional constraints, the user will be asked to perform whatever he needs to perform before he's allowed to view the document. If the user doesn't agree with some term, the document will not be displayed. One problem here is the order of additional constraints: say, for example, that there are two pay-per-view constraints in the document. The user agrees to pay when the first constraint is displayed, but, when the second one pops up, he decides to back off. Now the first payment has already been made, and unless there is a rollback mechanism, it is forever lost. Surely he will never use the DRM software again… An easy solution to this would be to show all the additional constraints at the same time. The problem is that if there are very many additional constraints, all displayed at once, the user may be inclined to just accept them without reading like many users do today when installing a software with a twenty-page license agreement.

5.4 Challenges in the ODRL specification

The ODRL specification defines a huge amount of functionalities and places a lot of requirements for an implementation. We found that three requirements were more challenging than others. These should be carefully studied when starting to implement an ODRL based DRM software product. We implemented none of these features. In fact, all three are quite straightforward to implement. However, all of them would require some changes in the software architecture, and combined they would require a serious refactoring effort to keep the design complexity from rising exponentially. Also, implementing these would easily triple or even square the amount of work needed to run thorough tests. These requirements basically make it easier to write certificates, but they also make the implementation of a software tool that parses DRM documents quite a bit more complex.

The first requirement that complicates implementation is logical operators. AND is the easiest operator to implement with the architecture that we used: every object validates the action and if there's a conflict, an exception is thrown. AND is the only operator we implemented. NOT operator is the easiest, even though we didn't implement it as such, it can be implemented by the one who writes the certificates by inverting the rules. An OR is difficult. Even though in theory it can be implemented by the certificate writer by the rule A OR B == NOT A AND NOT B, this makes the job of writing the certificates very difficult. Our architecture doesn't bend easily to support OR, it would require some kind of transactional support: if one rule fails (throws an exception), another may still save the day by being valid and "roll back" the exception.

Another challenging requirement in the ODRL specification is documents internal references to its elements. The specification states that any element may be linked from any other place of a document. This is fine as long as the semantics of the element does not depend on the context it is in. The third requirement presented in the next paragraph destroys this assumption, thus making this requirement very hard to implement. A reasonable way to implement this would be to run the document through a preprocessor before handing it over to the parser. The preprocessor would expand internal links to full XML elements and the parser wouldn't need to know anything about the links.

The third challenge is that some deeper level element may depend on its ancestor elements' data. For example, "if a Requirement appears at the same level as a number of Permissions, then the Requirement applies once to all of the Permissions" [2]. From the parser point of view, this is one special case more: when parsing a Permission, the parser should also check the parent element if it has a requirement. If it has, then append it as it had been a child of the Permission. Alternatively, a preprocessor could be the solution to this one, too: move all Requirements that aren't under any Permission, under all the Permissions that they are on same level with.

6. Technical problems with DRM

6.1 Open source problem

The problem of open source software was mentioned earlier in the document. This is a difficult area: if the browser is distributed as open source, what prevents a malicious user from modifying the browser so that it will open all assets without even looking at the certificates? One means is to try and formulate the license agreements such that they forbid modifications that lessen DRM functionalities. This is a hopeless task: even if the restriction could be unambiguously formulated, surely no license agreements would hold back pirates. As long as the source code is open, it has to be assumed that anyone may modify it the way they want. Even if the source code was not publicly distributed, there exist many tools for decompiling and debugging executable code, inserting hooks and thus affecting what the software does. So, what can software be trusted with? What should be implemented in hardware? This is a very fundamental question. Unfortunately experiences from this project do not indicate anything unexpected: software is unreliable at best, and anyone might do anything with the code.

6.2 Combining different kind of certificates

This is a great challenge; in our project we did not implement this feature in any way. Combining certificates can be very tricky, for example let us think about following situation: a user buys a certificate that gives him the right to use the product five times in this month. Obviously every time he uses the product, the amount of allowed usage times must be reduced. But what if the same person buys two different certificates? The first certificate gives him the right to use the product for this month. The second certificate gives five instances of use. The latter is useless in this month, but possible to buy, or win in a lottery. These two situations are totally different, so we cannot combine these certificates directly. In the latter situation, instances of use cannot reduce in this month. This certificate-merging problem is discussed in a more technical level in chapter 5.3.

6.3 Supporting DRM

DRM should be supported by the operating system. That would be a much better approach than application layer. If the implementation is made on the application layer, we have some serious problems. For example saving the certificates: if somebody gets a five instances of use certificate, he can easily copy it. When instances of use have run out, he loads a new copy of the original certificate. This is possible because every program can equally use and manage the files. If operating system would support DRM, it would guarantee that no other applications could read or change saved certificates information. However before DRM will be supported by operating systems, there must be a widely supported standard. Somebody must also pay for these features, so the market must be ready and interested about DRM.

6.4 Security

This is always a problematic issue. The copyrighted material should be encrypted so that it cannot be illegally used. The main problem is that information should not be decrypted by the application itself, because then the decrypted information would reside in digital format in the memory of the operating environment. For example, if digital music is decrypted by the player application, another application can read the decrypted digital music data and create a perfect copy. If decryption is done by hardware, in this situation by sound adapter, is the solution then perfect? Unfortunately not: the sound adapted has a plug for loudspeakers and even a digital audio out interface. It is quite simple to record sound from sound adapter and bypass DRM limitations this way. If the decryption was done in the loudspeakers themselves, nobody could hijack the plain audio stream from sound adapter. This isn't a perfect scheme either, since somebody could use a microphone and recorder. But this is about as good as possible. And using microphone and recorder does not produce a perfect copy: the result will be an analog copy, which is not as good as the original.

Decrypted content is not the only security threat. Copying certificates is another, and quite complex to solve. This problem is discussed further in a chapter that discusses technical problems. It is quite difficult to save something onto a user's hard drive and still make copying impossible. It is very simple to move a hard drive to another computer with another operating system. And still the pirates should not be able to copy the data. Or how can we prevent the pirates making an identical copy of the whole hard drive? How can we detect an attempt to copy data? This is one reason why DRM should be supported by the hardware. However, this raises some new questions. If we have loudspeakers that support DRM, can we listen free material that has no DRM extensions? And the transmitting of certificates over a network should be secure. It is very simple to listen network traffic and record the certificate while it is being downloaded. Recorded data could be used and, instead of hacking client software, the pirates could bluff the client with a fake authority. [19]

7. Non-technical problems with DRM

This chapter introduces some general problems with digital rights management that are not of technical nature, but more like political, economical or psychological.

All problems with digital rights management schemes are not purely technical. A working DRM model would require a lot of co-operation from the hardware and software vendors. A common standard has to be agreed upon. If a product, for example a movie, worked only on the video cassette players of one vendor, the consumers would not probably buy it. Or, if the DRM scheme were constructed so that all non-supporting players could play it but would not support DRM restrictions, the users of non-supporting players would gain a big advantage. This would seriously damage the sales of players that actually enforce the DRM restrictions, and thus probably destroy the whole idea of DRM. Some common standard is required in order to create markets for the product, but it is not easy to get all manufacturers work together. In general, many companies are willing to create a common open standard. However, some companies consider that their competitive advantage requires them to create their own, non-compatible standards since that enables them to control the standards development and usage. By patenting some key elements of the standard a company can successfully control how their competitors use the standard, and collect license fees.

Another issue that we encountered in our project is how guarantees are implemented. Should a user have a right to get a new certificate for free if he loses or breaks the one he has bought before? Obviously this introduces severe technical, economical, and legal problems. In general, a user should not be able to get unmerited advantages by losing a partially used certificate and receiving a new unused certificate instead. Privacy is also an issue. In order to give free certificates for people who have bought certificates before, a retailer should keep a list of customers. Keeping lists of what people have bought involves important privacy concerns.

If the products are sold on-line, for example in a web shop, billing is always a problem. The retailers, banks and other groups connected to the situation must agree on a common standard for billing online transactions. Small payments are very problematical – many have tried to create a standard (for example the eCash project) but none has succeeded yet. Globalization also leads to new problems, like currency conversions, different regulations, and so on.

The DRM schemes are currently mostly being developed by the content producing industry. The content producers look the DRM from their own point of view – how to make more money from their content products. They are not necessarily interested in improving the products; instead, it can be a better business to imply new restrictions on how the consumers can use them. Currently, when a consumer buys a product like a movie or a book, he can use it as much as he wishes when he has paid for it once. Getting a payment for a product may be a good business, but what if the content producer could get more than one payment? Maybe a payment for every time the user uses the product? Maybe double payment if the consumer invites his friend to watch the movie with him? The business potential is obvious – forcing new restrictions is a much better policy than actually improving the product. This has resulted to the fact that a product with DRM features is always worse to the consumer than the same product without DRM. The consumer loses his right to use the product everywhere, or in any way he decides. The consumer in general has to pay more for the same product. The loss of freedom is so clear that most consumers can notice it, which in turn is a problem for marketing and sales of products. Of course, many legal restrictions exist even in conjunction with the non-DRM products. For example, unauthorized copying or public displaying of copyrighted works are in general unlawful. Currently, however, these restrictions are controlled by legislation, not technically as with DRM. On the other hand, a DRM system may also enforce efficient restrictions that are not supported by the legal system.

Developing new DRM features is expensive. It takes a lot of money and more income is needed to make the DRM profitable. The first thought might be to increase the price of a DRM product. However, from the consumer's point of view, the DRM featured version of the product is worse than the one without DRM. The user cannot use a DRM featured product as freely as a traditional one since it has usage limitations. If the product is worse and it costs more, marketing becomes challenging.

Digital rights management separates the product itself from the right to use the product. The right to use the product (provided by a certificate) is usually person-specific so that it would not be possible to copy the product and the certificate to other people. This creates problems in the situations where the owner of the product lawfully changes. If a consumer gets tired of his copy of a movie he is usually legally allowed to sell it to his friend. But if the certificate only authorizes the original buyer to use the product, the possibility to resell the

product no longer exists. Possibly, it will not take long before the consumers realize this and consumer organizations start complaining. Public libraries will also have problems – they have a legal right to loan a DRM-featured book to their customers but without a certificate the customers cannot use the book. Libraries are in general a bad business for the content producers so they might enjoy seeing public libraries cease to exist but that is not in the interests of the society.

Another problem rises from the copyright legislation. It is not always forbidden to copy a copyrighted work. For example in Finland everybody is allowed to make a copy of a copyrighted work for his personal use. The United States copyright legislation includes a ”fair use” provision that allows, for example, limited reproduction of copyrighted works in educational purposes. DRM can make all these rights futile leaving the consumers without the rights they legally have. It may be allowed to show one minute of a movie related to a educational presentation, but what if it is technically impossible because the publisher of the movie has included DRM restrictions to the product? Digital rights management creates many situations where the user has a legal right to perform an action with a product he has purchased, but the action is denied by the DRM technology. Digital rights management takes a lot of rights away from the consumer, and at some point they will surely find that out. And when they do, the content producers must have some kind of answer ready for them. Some solution has to be developed to support the user rights.

8. DRM from different points of view

At least as important as the technical difficulties are issues relating to the “human interface”. What are the advantages and disadvantages of a DRM model for the stakeholders? DRM offers different advantages to different parties. In this chapter, the viewpoints of content producer, publisher, distributor and the customer are discussed. There are some benefits and some disadvantages for each.

8.1 Content producer

For content producer DRM is a great idea. Producer will have his rights to the content, and illegal copies are hard to make.

But DRM could really be an advantage to small producers like, for example, university professors. They write a lot of research material, but these are not usually widely spread. Papers may be hard to find. DRM would make it easier to distribute this kind of material. And the producer could, for example, allow a limited time to read the material and force the readers to purchase more rights for more serious studying.

On the other hand, DRM makes the use of some kind of agents or publishers practically compulsory, since certificates can hardly be “home made”. This means that even in this era of digital products and global communication, we would still need some intermediate parties that would benefit from the content producers work. In traditional business, though, there have always been agents and other hands between the producer and customer. This would not be such a big problem if content producers could create certificates themselves. The publisher has this technical knowledge and is capable of doing certificates.

During the project, we discussed whether we could include this kind of functionality in the software, but came to the conclusion that it would be a project of its own. To analyse a

certificate is complex enough, implementing software for constructing certificates is much harder. For now, a text editor along with the ODRL specification shall suffice. This capability is compulsory when designing a fully-grown DRM system.

If certificates could be homemade, DRM would have much more features for the content producer. People could make a home video, put it on the web and give certificates for their friends. This would create a new type of business: some small products could be sold straight from the producer to customers. But if certificates could be homemade this could reduce the reliability of certificates and DRM, which is another technical challenge. [18]

8.2 Publisher

DRM opens a lot of new possibilities of publishing. With DRM a publisher can release a free sample, for example a music recording. Anyone can download the song and listen to it once, but the second time will cost something. This is a totally new way of distributing free samples. When a customer has the content already, it is more likely that he will buy a certificate. And it is very simple to create a one-week certificate to a product. This is done even nowadays without DRM, but not so efficiently. If we had a reliable system for digital rights even big companies like Microsoft could make these kinds of releases. This requires that buying is easy and fast. Delivery and distribution are not so heavy processes anymore, since content can be freely copied. This is a huge change. And products could have various kinds of licenses. For example it could be possible to buy a one-year license for software. And regional pricing would be very easy. Same program could have different price in different countries, and same web-store could easily sell and make suitable certificates. Regional pricing is widely used, depending on the financial situation of a country.

The role of publishers may change dramatically. Delivering products to stores is no more the main work. Instead, searching for new producers as well as finding, aggregating, editing, and marketing, products will be central. Pricing can be very complex, because of the many features certificates have. But why suppliers cannot deal with these issues directly and make publishers unnecessary? It is highly unlikely that suppliers would be widely trusted by the content producers and that suppliers could offer something better than publishers. And suppliers would need much new knowledge about producers and publishing.

In these schemes, the publisher will control the rights and market the product. Publisher has now more advanced tools for managing rights. But publisher needs to posses even more knowledge about digital rights and digital business overall. In a DRM scheme, somebody needs to produce certificates and this must be done reliably. Whether this will be the role of publishers, remains to be seen.

8.3 Supplier

DRM makes more products available, and they can be more easily distributed via for example a web-store. This is an advance, since web-stores have a larger potential customer base. DRM makes copying more difficult and makes selling new types of products possible. In many cases the content itself can be freely copied, since content is useless without a suitable certificate. So contents can be in ftp-servers mirrored all over the world. This makes access much faster. It is hard to imagine a movie seller who transmits movies through Internet to customers all over the world. The situation is much better when certificates are the

subjects in selling movies. It is much more efficient to transmit ten kilobytes of certificate than one gigabyte of movie.

But there is also the other side of the coin. Constructing certificates can be complex as discussed in chapter five. Methods for this must be secure and hacker proof. If certificates are easy to produce or copy, the whole system collapses. And when volume rises, prices will lower. So there will be more transactions, but smaller payments per unit. This can increase relative expenses.

8.4 Consumer

DRM is designed to someway restrict the use of content. So it is obvious that the consumers are critical about the idea. The customer must vote, with his wallet, for DRM or it will never succeed. What would make customer vote for restrictions? Customer wants some benefit from DRM. New technique is not enough; it must bring some concrete benefits. And a large amount of producers must accept the technique. DRM should be as transparent to the customer as possible.

The benefits for the customer are more freedom of choice and reduced prices. Prices can be reduced when copying is extremely hard. But will the prices drop? Nintendo games were hard to copy, but the price was considerably higher than the same game for PC. Open standards and competition can make products cheaper. Open source projects will not benefit from DRM. And this can be a problem. If DRM will be implemented in open source, it is possible (and likely) that somebody will make a release that bypasses DRM restrictions. But more choices for customer are guaranteed. People can buy different kinds of rights to the product and more products will be available worldwide.

When DRM is introduced to the public, DRM-products must be much cheaper than non-DRM options or they must provide some other benefits. Otherwise this new standard will never be supported. And it would be wise for publishers and suppliers that prices would be very reasonable at the start. If DRM gets widely adopted, and all products support DRM features, prices are bound to rise since the customers have no alternatives.

9. Summary and conclusions

In the digital world, immaterial products are cheap to distribute through fast, global data networks. This has a lot of business potential, but also raises a big piracy problem. DRM (Digital Rights Management) is a system that separates the product itself from the right to use it. With a DRM system, the content publisher can limit the usage of the product so that it can for example be used only for a specified period of time.

A digital rights management system can be implemented by hardware, the operating system or special player/browser software. All implementation methods have their advantages and disadvantages, but in general it is more secure to implement the control mechanisms using hardware.

So far there is no widely used standard for rights description languages or describing the DRM limitations. We studied three competing standard proposals and selected ODRL to be used in our project. Creating a global standard is hard because all major companies have

their own proposals. Legislation can be used to fight against break-in tools that would help users to bypass the DRM control mechanisms.

This paper is about experiences we had during the Nonius project – a software project that made an extension to the X-Smiles XML browser that enabled DRM support in the browser. The project lasted for a year, and we had 7 people creating the program that we named DRMTool. The project was a success. At the end of the project, we held a demonstration about the product with our partners and other people related to the DRM research.

The project succeeded in implementing a piece of software capable of demonstrating DRM features with the most important DRM features. The most important of the implemented features are restricted individual, time of validity, and restricted counts of use. Some of the most interesting features that were left out are aspect constraints (quality, watermark) and target constraints (e.g. the purpose of use).

We found that when implementing DRM software, the most difficult tasks are implementing security and the DRM document parser. In fact, a secure DRM system requires the support of hardware devices for security. Based on our experiences, when implementing a DRM document parser, flexible software architecture is a must, since the technical requirements are hard to capture all at once. We needed to make some assumptions to make the chosen architecture work. We kept the technical risk level low by proceeding incrementally, implementing a couple of functions per iteration.

The implementation is far from perfect. The most obvious shortcomings are related to security: certificates and user data are stored in a user's hard disk in a human readable and modifiable format. There is no support for a secure encryption algorithm for protecting the content when it is transmitted over the network. This reflects the low priority that the security issues had in this project.

Based on our experiences, merging different certificates into one and implementing an interface for creating certificates are among the most challenging tasks when implementing a DRM system. The ODRL specification, too, produces some challenges for implementation. These were discussed in more detail in chapter five. In our opinion three of the most challenging features defined in the ODRL specification are logical operators, documents' internal links to its elements and the requirement that in some cases child elements may depend on some ancestor elements' children.

In chapter six, some general technical problems were discussed, even though largely left unanswered. How can we make secure software when the source code is available to anyone? In what layer should DRM be supported: application software, operating system, audio devices or even further? And how secure should the system be? More secure means more costs and higher product prices, and low security may lead to piratism. When talking about security, we easily think about the security of the content. But we must remember that copying of certificates and manipulating them is also a hard security problem. There are also great challenges with combining certificates and they must be solved before DRM can be widely used. And every instance should support DRM. Every part of the business, software producers as well as hardware producers, should stand for one standard. To make this even harder, at least at first, customers are likely to choose products without DRM support since they do not like the new limitations.

All the problems with DRM are not of technical nature. Creating a common standard that could be used in all products globally is a very hard political issue. Especially big companies want to create their own standards so they can have more control over them.

If the immaterial products are sold in a global Internet shop, new problems will arise. Guarantees must be implemented somehow, and some solution must be found for the payment. Users in different countries have different currencies, and everything must work smoothly.

Probably the biggest non-technical problem with DRM is how to sell it to the consumers. A DRM-featured version of a product is always worse for a consumer than the same product without DRM, since DRM controls how the consumer can use the product and therefore takes away a lot of freedom from him. Developing DRM control mechanisms is expensive, and the publishers may want to get their money back by increasing prices. But if the DRM-enabled product is worse to the customer and it also costs more than a traditional product, it is a big challenge to actually sell it to the consumers.

Digital rights management also takes away some rights that users are used to have. The publisher can deny the user from selling the product forward to someone else after using it himself. Public libraries need special certificates with a lot of usage freedom, but are the publishers interested to give those? DRM will also make technically impossible some actions that are allowed by copyright laws, like copying a product for personal use or reproducing a part of a product for educational purposes. The publishers must have some answer to the people who find out that they have lost rights they used to have.

Digital rights management is designed for content producers and publishers. They have big wishes for this technique. At the end the meaning of DRM is to produce more money to publishers and content producers. Customers at current time don't know or care about DRM. But what happens when they find out? Will they fight for their rights or do they care at all?

DRM is a radical change for publishers since the main work is no longer producing and delivering physical products. They should obtain new knowledge and be capable of producing and distributing certificates. And in the air is a big question, are publishers needed anymore? Maybe suppliers could produce the certificates?

All in all, the project succeeded in its main goal of producing a piece of software for demonstrating DRM functions. It showed that a DRM system can be implemented in open source and open technologies, specifically using ODRL. DRM in itself is a very complex and largely controversial subject, and the results of this project do not give any hint or direction about the future of DRM. Anyhow, a lot of issues came up, and a lot of experiences from the implementation process were gathered. Hopefully these experiences are of value to future implementers of DRM systems.

10. Terms

CPRM

Content Protection for Recordable Media. A system that would force DRM control in computer harddrives.

DMCA

Digital Millennium Copyright Act. The new copyright law in the United States.

DOM

Document Object Model. One way of handling and parsing XML documents programmatically. Represents the document as an object model.

DRM

Digital Rights Management – system that separates a product from the right to use it

eBook

Adobe's electronical publishing format that includes DRM control mechanisms.

MPEG-21

International standard e.g. for describing digital rights in multimedia.

ODRL

Open Digital Rights Language. A language used to specify DRM parameters in XML document

SDMI

Secure Digital Music Initiative. An attempt to stop illegal copying of music by using watermarking technology.

SMIL

Synchronized Multimedia Integration Language. A language for presenting multimedia documents. Specified by the World Wide Web Consortium (W3C).

SSSCA

Secure Systems Standard and Certification Act. A proposal for law in the United States that would make DRM regulations obligatory in digital player devices.

XForms

An XML technology for presenting and transmitting HTML like forms over the network.

XMCL

eXtensible Media Commerce Language. A language used to specify DRM parameters in XML document

XrML

eXtensible rights Markup Language. A language used to specify DRM parameters in XML document

XSL

Extensible Stylesheet Language. An XML language used for defining stylesheet transformations of XML documents to some representation.

References

[1] XML Devices project at HUT. *X-Smiles browser, project web pages*, 2002, <http://www.x-smiles.org>

[2] Renato Iannella. *ODRL Specification*, 2002, <http://www.odrl.net/1.0/ODRL-10.pdf>

[3] J. Buffenbarger. *Syntactic Software Merging*, Seattle, June 1995, pp153-172.

[4] Robin Cover. *The XML Cover Pages: XML and Digital Rights Management (DRM)*, 13.5.2002, <http://xml.coverpages.org/drm.html>

[5] Neil McAllister. Freedom of Expression: Emerging standards in rights management, March 2002, <http://www.newarchitectmag.com/documents/s=2453/new1011651985727/index.html>

[6] *XrML standard homepage*, <http://www.xrml.org/>

[7] N.N. *XMCL standard homepage*, <http://www.xmcl.org/>

[8] N.N. *ODRL standard homepage*, <http://www.odrl.net/>

[9] Hal R. Varian and Carol Shapiro. *Information Rules*, 1999, Harvard Business School Press

[10] *Testimony of Boniie J.K. Richardson, MPAA,* <http://www.mpaa.org/legislation/press/2001/2001_05_22.htm>

[11] Bruce Schneier. *Futility of digital copy prevention*, 2001, <http://www.counterpane.com/crypto-gram-0105.html#3>

[12] John Gilmore. *What's wrong with copy protection*, 2001, <http://www.linux.it/GNU/articoli/whatswrong.shtml>

[13] *EFF's page about the SSSCA*, 2001, <http://www.eff.org/alerts/20010921_eff_sssca_alert.html>

[14] Declan McCullagh. *Politech's summary about the SSSCA*, 2001, <http://216.110.42.179/docs/hollings.090701.html>

[15] *Security Systems Standards and Certification Act Draft*, 2001, <http://cryptome.org/sssca.htm>

[16] *The EU Copyright Directive*, 2001, <http://www.ivir.nl/legislation/eu/copyright-directive.html>

[17] Bruce Schneier. *Crypto-Gram*, August 15, 2001, <http://www.counterpane.com/crypto-gram-0108.html>

[18] Andrea Pruneda / Microsoft Corporation (white paper). *Windows Media Technologies: Using Windows Media Rights Manager to Protect and Distribute Digital Media*, <http://msdn.microsoft.com/msdnmag/issues/01/12/DRM/DRM.asp>

[19] Microsoft Corporation. *Features of DRM*, <http://www.microsoft.com/windows/windowsmedia/WM7/DRM/features.asp>

[20] Intertrust Corporation (white paper). *Understanding DRM Systems*, <http://www.intertrust.com/main/research/index.html>

[21] Adobe Corporation. *Adobe eBook Reader 3.2*, <http://www.adobe.com/products/ebookreader/overview1.html>

[22] Ville Saarinen, Jani Poikela. *Nonius-projektin loppuraportti*, 2002, <http://jt7-332.tky.hut.fi/nonius/lu/loppuraportti/loppuraportti.doc>