# Security Attributes Based Digital Rights Management*

Jordan C. N. Chong[†] René van Buuren[‡] Pieter H. Hartel[†] Geert Kleinhuis[§]

February 3, 2002

## Abstract

Most real-life systems delegate responsibilities to different authorities. We apply this model to a digital rights management system, to achieve flexible security. In our model a hierarchy of authorities issues certificates that are linked by cryptographic means. This linkage establishes a chain of control, *identity-attribute-rights*, and allows flexible rights control over content. Typical security objectives, such as identification, authentication, authorization and access control can be realised. Content keys are personalised to detect illegal super distribution. We describe a working prototype, which we develop using standard techniques, such as standard certificates, XML and Java. We present experimental results to evaluate the scalability of the system. A formal analysis demonstrates that our design is able to detect a form of illegal super distribution.

## 1 Introduction

Annual losses to the film and music industry due to online piracy amount to billions of dollars annually [1]. Digital Rights Management (DRM) provides a potential solution to the online piracy problem. DRM systems manage copyrights on digital content in untrusted cyberspace. Commercial DRM platforms are available from SealedMedia, InterTrust, IBM, and Microsoft. Some music and movie industries have changed their online business models to subscription-based music sales and pay-per-view, such as Sony Music Entertainment (see `sonymusic.com`) and Universal Music Group (see `umusic.com`). These systems are proprietary or include key components that are proprietary. We propose an open system, and analyse how effective it is.

Section 2 introduces our design of a DRM system. Section 3 gives a user scenario to illustrate how the system works from the user's point of view. Section 4 discusses the state of the art and related work. Section 5 describes our prototype system, a performance evaluation of the implementation, and a SPIN model of the system. Finally section 6 concludes the paper and briefly explains future work.

## 2 Delegating responsibilities

Most real-life systems delegate responsibilities to different authorities. The authorities co-operate to provide an efficient service. We apply this model to DRM. We elaborate the role of the authorities in Section 2.1 and establish their relationships in Section 2.2.

### 2.1 Authorities

We consider a client-server setting in which a user downloads content, where her rights on the content are carefully controlled, say by the content providers or by DRM service providers. We assume that each user has a unique identity. A standard public key certificate can then be used to bind the identity to a public key. The public key certificates are issued and distributed by a Certificate Authority (CA) [2]. We assume that the server can establish the validity of the public key certificate, so that the server can identify and authenticate the user.

*This work was supported by KPN Research and Senter, both of the Netherlands under contract nr ??

[†]Dept. of Computer Science, Univ. of Twente, The Netherlands, Email: `{chong,pieter}@cs.utwente.nl`

[‡]Telematica Institute, Enschede, The Netherlands, Email: `Rene.vanBuuren@telin.nl`

[§]KPN Research, Groningen, The Netherlands, Email: `G.Kleinhuis@kpn.com`

Additionally by using the server public key certificate, the user is able to authenticate the server.

For maximum flexibility, we assume that different security attributes can be associated with each identity. Security attributes are information, other than cryptographic keys, that is needed to establish and describe the security properties of a user in the system. These security attributes may include role, group membership, time of day, location to access resources, password etc. A security attribute is encapsulated in an attribute certificate. The latter has a similar structure to the public key certificate but does not carry a public key. The Attribute Authority (AA) [3], is responsible for issuing, assigning, signing, distributing and revoking attribute certificates. Note that the CA and AA could be the same authority, but for maximum flexibility the CA and AA would be different parties. An attribute certificate is signed with the AA's private key to ensure the integrity of the attribute certificate.

Having established the identity of a user, and her security attributes we are now able to decide

1. What content is accessible to the user, and
2. What rights the user may exercise on that content.

The separation of deciding *what content is accessible* from *what can be done with the available content* is a key aspect in our system; it creates flexibility, and at the same time may ease the implementation. For example consider a document that contains both aggregate and detailed business data. Then users with appropriate security attributes may obtain the detailed data, whereas others may only obtain the aggregate data. Establishing the particular rights (i.e. the ability to view, print, save etc) for any of these users holding their respective attributes is still a separate, orthogonal issue.

We use a digital license to capture the rights of a user on particular content. The digital license carries information about the content, the license holder, the payment status (if payment is involved), as well as other terms and conditions of using the content. The license provides fine-grained control over the content. The Clearing House (CH) is responsible for issuing, managing payment, and distributing and revoking digital licenses. Again the CH could be the same as one of the CA or AA but would be different from either for maximum flexibility.

The licensing mechanism we have sketched above is not vastly different from other DRM mechanism. However, there are two innovations:

1. Identity, attribute and rights are decoupled to allow for maximum flexibility;
2. Digital licenses are generated on demand, after the identity and the security attributes have been verified. This is the earliest moment when the system is able to decide which (partial) content must be delivered, and what the associated rights should be.

One may ask what the difference is between the public key certificate and attribute certificate and why we also use a digital license. Lets look at an illustration in real life here: A Malaysian resident wishes to enter The Netherlands and she likes to stay in the country for more than a year. She needs a passport; a visa and a residence permit for doing that. The Malaysian Passport Service (at the Immigration Department) issues the passport. The Visa Service (at The Netherlands Embassy) grants the visa. The Alien Police Department in The Netherlands distributes the residence permit. Three trusted authorities are thus involved in this process.

A public key certificate can be seen as a passport - it identifies the owner, it tends to be valid for a long period, it is difficult to forge and a strong authentication process is used to establish the owner's identity.

Our attribute certificate is like an entry visa. A different authority issues it, and in most cases a passport has a longer validity than a visa. As a consequence, acquiring the entry visa becomes a simpler procedure. The entry visa will refer to the passport as a part of how that visa specifies the terms under which the passport owner is authorised to enter the country. Once the passport owner is identified and authenticated, the visa may authorise her to enter the country.

Once in the country, the traveller may apply for a residence permit, which is then an analogue to our digital licence.

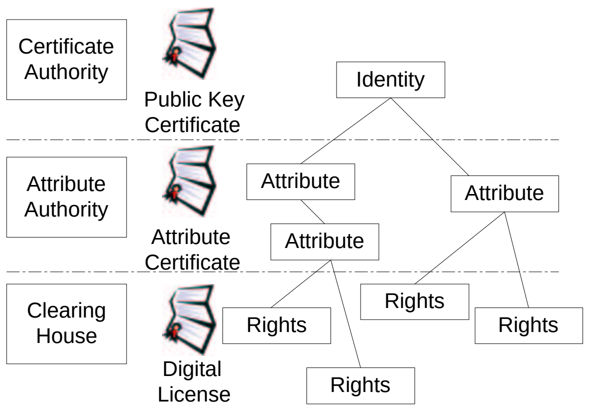

Figure 1: The hierarchy of the authorities and associations between digital proofs.

## 2.2 Association of authorities

There must be a relationship between authorities to establish the chain of control. For example the visa is a stamp in the passport, therefore the link between visa and passport is difficult to break. In the digital world, the public key certificate, attribute certificate and digital license are linked using cryptographic techniques.

Figure 1 shows that an identity may be associated with several attributes, that an attribute may be associated with other attributes, and that an attribute may be associated with several rights. For example, Alice (*identity*) is an editor (*attribute*) and she belongs to an administrative group (*attribute*). The editor is only allowed to enter the system at a given time (*attribute*). The editor can print (*rights*) an annual report and also she can also view (*rights*) some confidential document.

Because of the interposition of attributes between the identities and the actual rights we call our system a security attribute based digital rights management (SABDRM) system. The association between these three properties identity, attribute and rights represents a chain of control. The distinctive features of SABDRM are:

1. The use of public key certificate, attribute certificate, content identification and a source of randomness to generate a secret, unique, personalised content key. This should allow multiple use of the same key to be detected with high probability.
2. A hierarchy of authorities that provides for flexibility in the licence management.

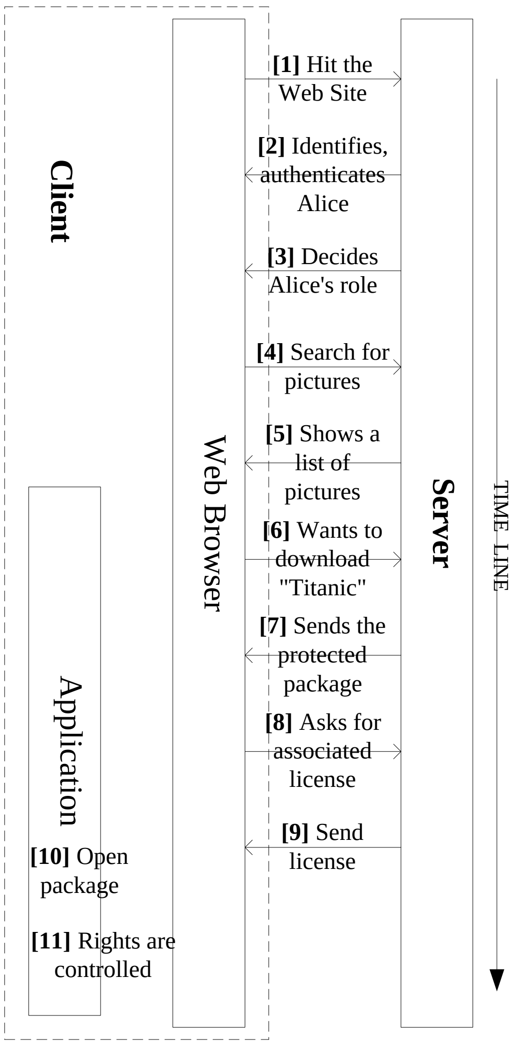
Client
Application
Web Browser
Server
TIME LINE
[1] Hit the Web Site
[2] Identifies, authenticates Alice
[3] Decides Alice's role
[4] Search for pictures
[5] Shows a list of pictures
[6] Wants to download "Titanic"
[7] Sends the protected package
[8] Asks for associated license
[9] Send license
[10] Open package
[11] Rights are controlled

Figure 2: A user scenario of the system.

# 3 User Scenario

To illustrate the system from the user's point of view, Figure 2 shows how the system evolves in a number of steps as follows:

1. Alice enters the system using a Web browser,

she clicks on a Web site to download a digital poster, example: `someposters.com`.

2. The server identifies and authenticates her, by her public key certificate.
3. The server grants the role 'Guest' to her for entering the system.
4. She looks for digital posters.
5. Using her role 'Guest', the server shows a list of digital pictures she is allowed to access.
6. She decides to download a digital poster of the 'Titanic'.
7. The digital poster is encrypted, packaged, and sent to Alice.
8. Alice needs the associated license 'Guest's Titanic license' to access the protected digital picture. So she asks for it from the server.
9. The server retrieves the rights she has according to her role as a 'Guest' by doing a database lookup. The server embeds the rights 'View' and the digital content key in the license generation and sends the license to Alice.
10. Alice has the protected digital picture and the associated digital license. She is now able to open the picture with the appropriate application.
11. She is prohibited to print the picture, because the license only allows her to view the picture.

## 4 Related Work

Having presented the core of our idea, we will now look at the state-of-the-art and some related work. DRM is a relatively new area and the business realities are not fully developed yet. To focus the discussion we use the following definition (See `XrML.org`):

> DRM refers to approaches to establish a foundation of trust that enables digital content trading, from digital content preparation, transaction to distribution.

The main purpose of the DRM is to prevent online piracy and unintended distribution of the digital content and to govern the usage rules of the digital content on networked systems.

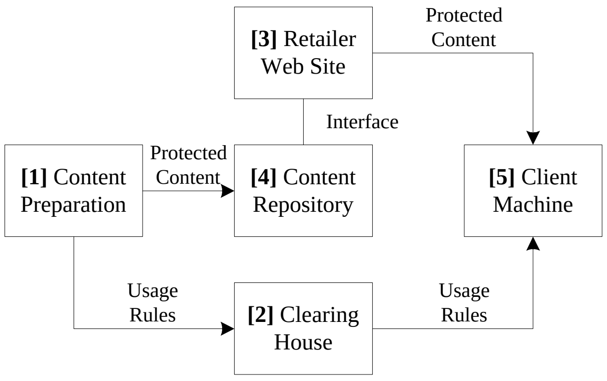

Figure 3: The common architecture of a DRM systems.

### 4.1 Commercial systems

There are several DRM platforms on the market: SealedMedia Enterprise License Server (ELS) (`sealedmedia.com`), Microsoft Windows Media Technologies (`microsoft.com/windows/windowsmedia`), IBM Electronic Media Management System (EMMS) (`ibm.com/software/emms`), InterTrust Rights|Systems (`intertrust.com`), and ContentGuard RightsEdge (`contentguard.com`). From the white papers on the various Web sites, we could glean only limited information. The architecture of these systems conforms to the generic architecture shown in Figure 3. We can identify the following major components of the architecture:

1. An application for the content creator or publisher to prepare the protected digital content and to set the digital rights or usage rules;
2. A server for license and payment management;
3. A system or interface with the user for digital content distribution management (could be a retailer Web site);
4. A secure repository for the protected digital content; and
5. A client application for accessing the protected digital content.

Table 1 shows a comparison of some of the currently available DRM platforms in terms of *appli-*

*cation* (the tools provided for accessing the protected content at the client side); some characteristic *features* of the platform; and the *content* type the DRM platforms support.

All DRM platforms provide a Software Development Kit (SDK) for developers to create and customize an application that is able to interpret the structure of specific protected digital content. The DRM systems of SealedMedia, Microsoft and IBM provide a secure environment at the client side for preventing malicious users from intercepting digital content. IBM and InterTrust deploy special hardware products for protecting the content keys and to provide a tamper-resistant environment for accessing the protected digital content. Most of the DRM platforms, including Microsoft, SealedMedia, InterTrust and IBM provide a service for updating or upgrading a service to counter possible advanced attacks. All except IBM and Microsoft support a variety of content type formats. The last entry in Table 1 lists some of the features of the prototype (SUMMER), that we have built. We will discuss the prototype in more detail later.

## 4.2 Literature

Horne, Pinkas and Sander [4] have invented a novel exchange protocol for peer-to-peer file sharing system for content distribution, which provides higher quality of service and restricts unlicensed distribution. Kwok and Lui [5] have proposed a license management model to provide peer-to-peer sharing domain. They implement two types of services, one at the server side and one at the client side, to handle consumer registration, payment, and license issuing processes and to deal with licensing in peer-to-peer distribution model.

Just regulating the distribution of digital content cannot solve the online piracy problems alone; it is also necessary to check that rights violations do not happen. Techniques to achieve this include watermarking and fingerprinting [6]. Dittman et al [7] present a technology for combining collusion-secure fingerprinting schemes based on finite geometries and a watermarking mechanism with special marking points for images. Fridrich [8] introduces the concept of key-dependent basis functions and discuss the applications to secure robust watermarking. Brin and Davis [9] describe their copy detection server which identifies copies of digital content, even for partial copies. In our system we are able to detect unaccounted key/license distribution due to the key personalization.

Silbert et al propose a self-protecting container, which they call a DigiBox for protecting the digital content by providing a cryptographically protected environment for packaging content and enforcing rights. The approach proposed by Durfee and Franklin [10] investigates trustworthiness of the distribution chain (i.e. from the Producer to the Consumers), which includes middlemen. Their approach allows trusted middlemen to alter rights but prohibits attackers from tampering with rights.

Last but not least, proprietary client-side DRM is another problem that has been researched. The main goal is to achieve application- and platform-independence, i.e. using different software applications to access the same protected digital content. Mourad et al [11] implement an application, namely WebGuard that enables existing Internet Explorer Browser and the browser's plug-ins to handle protected content. They use a technique dubbed 'Windows sub classing', which bypasses the Windows message passing within the operating system and application. We share their objective for application-independence but we work at the application level instead of the operating system level.

## 4.3 Standardization

Standardization organizations, such as the World Wide Web Consortium (`w3c.org`), Internet Digital Rights Management Forum (`idrm.org`) and the Motion Picture Experts Group (`mpeg.org`) are considering standardisation of digital rights languages. Such languages are mostly based on XML, to describe specifications of rights, fees and conditions for using digital content. At this moment, several rights language have been proposed and the drafts have been submitted for review. The eXtensible rights Markup Language (`xrml.org`), the eXtensible Media Commerce Language XMCL (`xmcl.org`), and the Open Digital Rights Language (`xmcl.org`) are some of the rights languages being proposed.

| DRM | Application | Features | Content |
|---|---|---|---|
| SealedMedia | Unsealer(a browser plug-in). | • Categorizes the digital content; one digital license can be used for digital content in the same category.<br>• Supplies a secure runtime environment for the application to prevent low-level application hacking.<br>• Implements trusted clock at the application, synchronized with license server. | HTML, GIF/JPEG, PDF, MP3, QuickTime video. |
| Microsoft | Windows Media Player. | • The licenses are described using XrML.<br>• End-to-end persistent protection: licenses and keys "lock" media files.<br>• Secure audio path to prevent an unauthorized program from capturing traffic between the player and the sound card. | Audio (MP3), video and movie. |
| IBM | EMMS player SDK. | • Provides a tamper-resistant environment for accessing the audio and managing a library of EMMS-formatted content.<br>• Use secure containers to thwart unauthorized use of the protected digital music.<br>• Uses a cryptographic coprocessor to protect the keys.<br>• Produces logs for all transactions. | Audio. |
| InterTrust | Toolkits. | • Updates the system automatically to install bug fixes.<br>• Provides a DRM add-on chip for secure microprocessor architecture.<br>• Supports multiple cryptography standards. | Any type of digital content. |
| ContentGuard | SDK. | • Produces the rights label during content preparation. The rights label is used to generate the license when the license is requested.<br>• Produces accounting reports and transaction audit logs. | Any type of digital content format. |
| SUMMER | Plug-in to Adobe Acrobat. | • Produces the protection package of digital content and generates the license on-demand.<br>• Personalizes the content key to the user. | At the moment, only PDF. |

Table 1: Brief comparison of DRM platforms with the prototype SUMMER.

## 5 Prototype: SUMMER

In this section, we describe the prototype that we have built to demonstrate the SABDRM idea presented earlier. We give an overview of the architecture in Section 5.1, followed by a summary of the implementation of the system in Section 5.2.

### 5.1 Overview

We have developed a prototype of our DRM system in the context of the Secure Multimedia Retrieval (SUMMER) project. The main objective of SUMMER is to design a secure distributed multimedia database management system for efficient multimedia retrieval from distributed autonomous sources, including database management systems, text, XML, and video or audio. Figure 4 shows the the SUMMER architecture. The present paper focuses on the right part. The left part (IAA-QM) is described only briefly below.

#### 5.1.1 IAA-QM

The Identification, Authentication and Authorization (IAA) module decides the identity of the user and then establishes the security attributes for that identity. Therefore, the IAA begins the chain of control of identity-attribute. As described in section 2, attribute certificates and public key certificates, which are generated and distributed by AA and CA respectively, are used as means to achieve IAA.

The Query Module (QM) enforces access control in the system. The QM acts as a filtering and monitoring environment on requests from the client and results from the server. The requests and the results are described using XML. A security policy file and the attribute of the client are fed in to the environment. The result displays the list of digital content accessible according to the attribute of the client. Further information on IAA-QM can be found in Damian it et al [12, 13], Bertino et al [14], and Kudo and Hada [15].

#### 5.1.2 DRM

The DRM part (right half of Figure 4) is composed of four components, namely the Content Module, the License Module, the Content Renderer and the Plug-in. The DRM part completes the chain of control started by IAA-QM, by linking the digital rights to identity-attribute. The Content Module is composed of the following sub components:

1. Content protection: generates, personalizes, and stores the digital content key in a secure fashion. We hash the public key certificate, attribute certificate, content identity and some random data to generate a key that is unique with a high probability.

2. Raw content repository: securely stores the unencrypted digital content.

3. Secure key storage: stores the content key generated by the Content Protection module.

In the prototype, the License Module implements the CH (Clearing House). The License Module generates digital licenses, which encapsulate rights, terms and conditions of using the digital content. The module also retrieves the associated digital content key from the key storage, encrypted by using the client's public key, and embeds in the digital license. The security attributes of the user, the identity of the content are fed into the License Module for license generation. The License Module needs the security attributes to retrieve a list of access rights the user possesses for the content. A digital license is generated on-demand and stored. The digital license is signed by using the server's private key.

The Log Store is used to record all transactions. The Log Store is assumed secure. The Log records:

- Content protection and download on the server side.

- License generation on the server side.

- License interpretation by the Plug-in on the client side.

The Log Store:

1. Provides a complete overview of all actions the client has exercised on the digital content;

2. Achieves non-repudiation, for audit trailing, and if necessary for billing, and

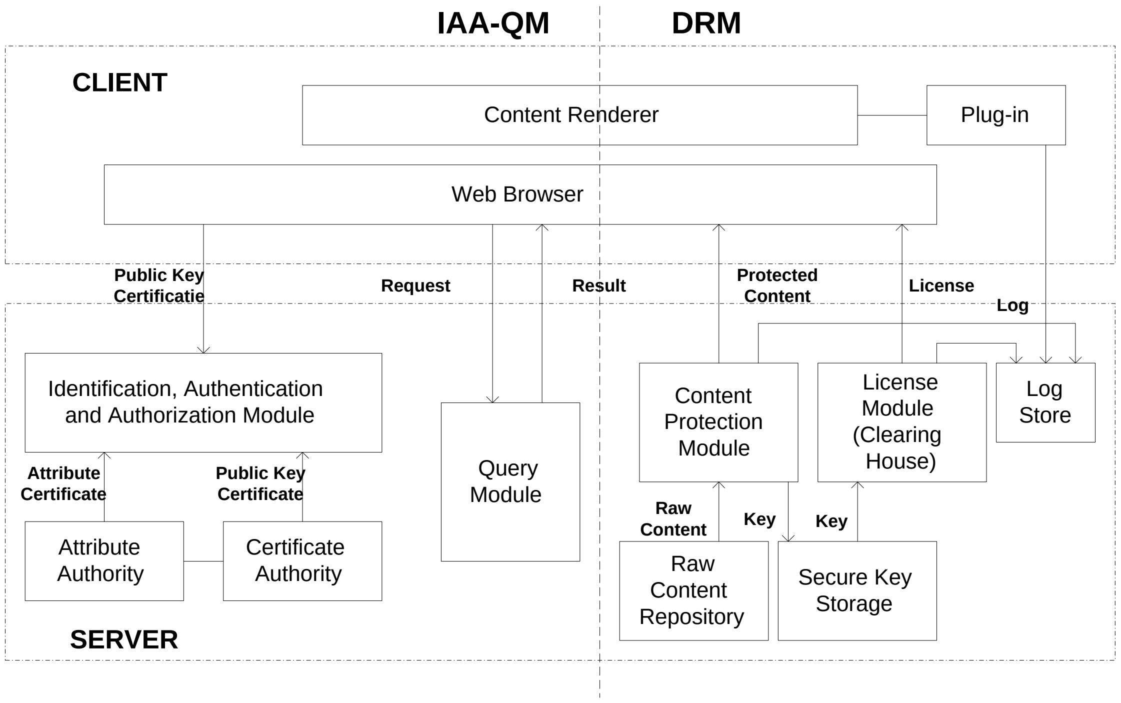

Figure 4: The overview architecture of SUMMER.

All records from the Plug-in are signed by using client's private key to prevent the client from denying any of the messages.

At the client side, a Web browser is used as an interface to the communication between the rendered and the server. We use Adobe Acrobat (`adobe.com`) as our Content Renderer to access the digital content in the form of PDF files.

We have created a renderer specific plug-in to interface our server with Adobe Acrobat as a proof of concept. For future work we plan to build a format/renderer independent client side application that interfaces to arbitrary renderers via much smaller renderer specific plug-ins. The development of the plug-in to the Adobe Acrobat gave us first-hand experience of application customisation and extension. The client side code plays a very important role on:

1. Understanding and interpreting the protected digital content structure.
2. Validating the digital licenses (signature verification and time validity etc.).
3. Decrypting the digital license.
4. Decrypting and retrieving the digital content key from the digital license.
5. Upholding and enforcing the client's access rights (triggering on/off the functions on the renderer application) on the digital content according to the digital rights stated on the digital license.
6. Logs all the client's actions exercised on the digital content.

The renderer and plug-in run in an insecure environment, which makes it possible for unencrypted content to be leaked. This problem occurs with most DRM systems. The problem can be solved to an extent by using tamper resistant hardware at the client side, and with watermarking techniques (See Section 4).

## 5.2 Implementation

This section presents a brief summary of the implementation of the prototype DRM system. All modules at the server side are written in Java (using servlet [16] and JSP [17] technologies). Cryptographic libraries used include the Java Cryptography Extension (JCE) [18], the IAIK Cryptography library – a Java cryptography library that conforms to JCE (`jcewww.iaik.at`), the XrML library (`xrml.org`), and the XML Xerces and Xalan libraries (`xml.apache.org`).

The public key certificates conform to the X.509 standard [19]. The structure of the attribute certificates is defined by the IETF PKIX Attribute Certificate Profile [20] [21]. The digital licenses are described by using XrML. We chose XrML as the rights language in this prototype because XrML is the only finished rights language at the time of implementing the prototype. Additionally XrML provides a detailed syntax for encapsulating fine-grained control information on a digital content.

Adobe provides a free C++ software development kit (SDK), which contains an API for users to develop customizable plug-ins with which to extend the functionality of the Adobe Acrobat. The cryptography library being used in the plug-in is the open-source, free and reliable OpenSSL cryptography library (`openssl.org`). Moreover, the Xerces-C (`xml.apache.org`) library that conforms to the Java version of Xerces is used for parsing and interpreting digital licenses in the plug-in.

We list the security features of the prototype in section 5.3. Thereafter we evaluate the performance of the prototype by investigating the efficiency of the server in section 5.4. We analyze the protocol of the prototype by constructing a SPIN model in section 5.5.

## 5.3 Security Features

The security features of the prototype are as follows:

1. The digital content is kept protected (encrypted) at the client-side. Only when the user has the proper digital license and certificates, the protected digital content can be decrypted and accessed by Adobe Acrobat (using the plug-in).

2. The digital content key is personalized to the user (as described in Section 5.1.2). Therefore the key can be traced back to the user. The owner of the key may thus be held responsible if a rights violation is detected – detecting such violations is outside the scope of this paper. The content key should never leave the confines of the user's machine.

3. The server (CH) is the only entity able to sign a license. Every client can verify the signature on a License.

4. Clients are not anonymous.

5. Clients can exercise rights multiple times. To control this, secure timers, counters and the like may be needed. This is an area of future work.

## 5.4 Performance Evaluation

We have performed some experiments to measure the overhead at the server side due to the on-demand content encryption and licence generation. The measurements would allow us to determine the limitations of the server, and extent to which the system is scalable.

We used Apache (`apache.org`) on a Pentium III 650 MHz, 128 MB RAM machine as a server. We have installed the Jakarta Tomcat (`jakarta.apache.org/tomcat`) servlet container to interface with Apache. The client is a Pentium III 850 MHz, 256 MB RAM, it is connected to the server via a intranet 10Mbps wavelan. The content encryption algorithm is Blowfish [22] with 128 bits key size. We have varied the block size for encryption to see how that influences performance. The two block sizes we have chosen are 1KB (1024 bytes) and 8KB (8192 bytes) (which are also the block sizes tested by IAIK - see `jcewww.iaik.at/products/jce/test/`). We have not tested with larger block sizes, because a larger block size makes it harder to hide the patterns of the plaintext securely [23]. Timings were generated by executing the target code block in a tight loop and by using `System.currentTimeMillis()` to capture (wall clock) timing information.

We have run 3 sets of tests on the prototype, measuring:

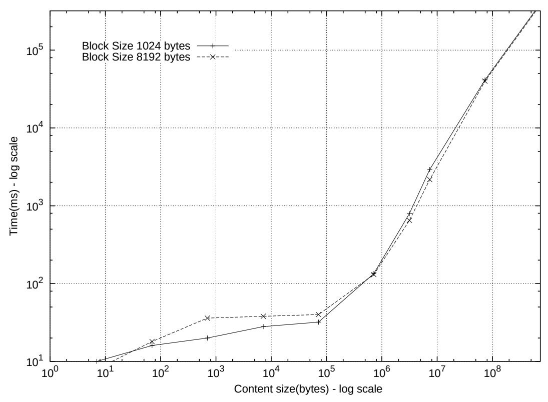

Figure 5: Time needed for content encryption as a function of content size. Two different block sizes are sketched. The x-axis and y-axis are in logarithmic scale.

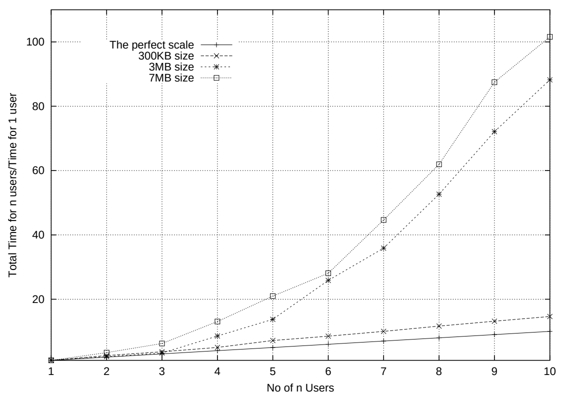

Figure 6: Scaled time needed for content protection as a function of the number of user activating the content encryption simultaneously.

1. *time as a function of content size* for content protection, as shown in Figure 5,

2. *time as a function of the number of users* for content protection, as shown in Figure 6, and

3. *time taken license generation as a function of the number of users.*

A typical video is 700MB (this just fits on a 700MB CD-R; a typical MP3 is 3MB. All other digital document (TXT, PDF, DOC, and so on) and digital pictures (BMP, JPEG, GIF, and so on) can be as small as 1KB, or as large as 10MB (or even bigger). To cover a significant variety of sizes of digital content, we have chosen the range 7B, 70B, 700B, . . . 7MB, 70MB, 700MB.

We assume a 10 Mbps network (typical cable modem for home user). An MP3 can then be downloaded in about 2s; a video takes 9 mins.

As can be seen in Figure 5, the time taken to encrypt the content with 1KB block size and 8KB block size are nearly parallel. We conclude that varying encryption block sizes does not show a significant difference. In the same figure, the curve from 7B to 70KB is nearly flat. The curve leaps drastically after 70KB, which we cannot explain. However, we can see that the time spent for an MP3 is approximately 1s, which is 50% of the time needed to download the content. Figure 5 shows that a typical 1 hour video of 700MB takes about 5 mins to encrypt; this represents a 55% time overhead. However, by overlapping encryption and downloading using streaming techniques [24, 25], the time take to encrypt can be completely hidden from the user.

Figure 6 shows how the performance of the server degrades when there is more than one concurrent user. For each of three file sizes 300KB, 3MB and 7MB, and for each of $n$ = 1,2, . . . 10 concurrent users we have made a number of measurements, and plotted the average of these measurements in the figure. We have scaled the averages, dividing the average time for $n$ users by the average time for one user. The line indicating perfect scalability is also shown. The performance of the server is not affected by small content sizes, but it becomes overloaded if the content size increases and/or if the number of concurrent users increases beyond a certain point.

To avoid degrading the service beyond the point where content encryption time can no longer be hidden from the user, the server could to decide when to accept and when to reject further demands for content could on the basis of these measurements.

The size of a digital license is around 10KB, and it can be generated in 60±20ms per license. This is negligible with respect to the download times for small content size.

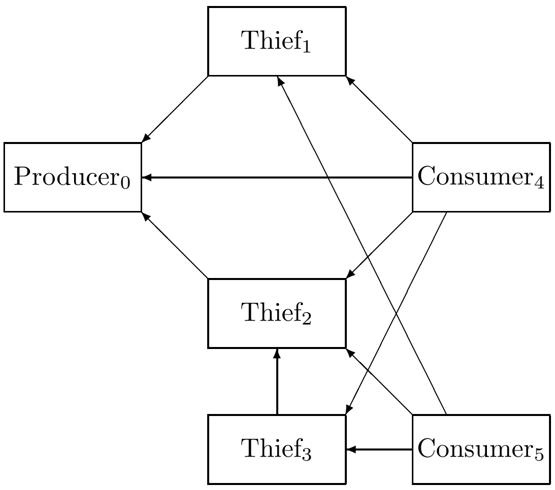

Figure 7: Network with a Producer, three Thieves and two Consumers showing arrows in the direction of Request message flow; License and Content messages flow in the opposite direction.

## 5.5 SPIN model

Designing correct security protocols is notoriously difficult [26]. Many tools and systems are available to help the designer to analyze the protocols [27]. Our system includes some moderately complex protocols. Therefore, we have used the SPIN model-checking tool [28] to help us explore a key property of the protocol: preventing the illegal re-distribution of content. SPIN is able to explore the state space of a model looking for undesirable states. An example of such an undesirable state is one where content is intercepted by a Thief, and redistributed.

Figure 7 shows a network of six processes, where the server (labelled as Producer) supports five clients. Three clients play the role of Thieves; the remaining clients (labeled Consumers) are honest, the Producer is also honest. Only the Producer is able to create original Packaged Content and Licenses. Thieves acquire Packaged Content and a License from the Producer or from each other. Thieves would do this in order to earn money: purchasing content for a certain price from the Producer, then selling it for less money to many Consumers would enable Thieves to earn arbitrary amounts of money. Consumers and Thieves can either download Packaged Content and License from the Producer or from one of the Thieves. Thief 3 demonstrates that another middleman can be involved in a transaction. Of course there are many networks involving any number of Producers, Consumers and Thieves. We believe that the network show in Figure 7 represents the essence of all such networks.

The processes shown in Figure 7 exchange three different types of messages:

- Request messages identify the desired content.
- Packaged Content messages contain the actual content encrypted under a unique content key.
- License messages contain a content key which is signed by the Producer. The signed key is then encrypted under the public key of the process requesting the license, i.e. a Consumer or a Thief.

Request messages flow in the direction of the arrows, the other messages flow in the opposite direction. In response to each Request message, a Packaged Content and a License message are returned. For example in the simplest scenario, Consumer$_4$ sends a Request message to the Producer, which returns the appropriate Packaged Content and License messages to the Consumer.

To keep the model simple and yet to make it sufficiently expressive, we have made the following simplifying assumptions:

- A private key is truly private, and public keys are universally known.
- A Thief must re-encrypt a License because a every client requires that a License be encrypted with her public key.
- The Producer and Consumer are honest; the Thieves are not. No parties collude.
- The Producer never reuses a content key. This allows the Consumer to check keys received for duplicates. A duplicate content key indicates the involvement of a Thief.
- Encryption is modelled by including the key in a message. Decryption then becomes comparing keys.

The model is limited and could be extended in the number of ways:

- It is possible to eavesdrop on a channels; to model this is future work.
- The Thieves are really middlemen [10] with bad intentions. A generalization to middlemen with good intentions would be valuable.
- Billing is not taken into account, this would rely on non-repudiable logging.

The key property of any DRM system is to prevent illegal super distribution: Anyone other than the Producer trying to redistribute content will eventually be caught. To ensure this we rely on two assumptions: that all content keys are unique, and that customers when receiving content keys are able to check that the key received is fresh. We will show below that our SPIN model satisfies this property. The SPIN model is of course an abstraction of the real system, and as such provides no guarantees about the prototype. However, the fact that the model satisfies the prevention of illegal super distribution property demonstrates that the design of the system is sound.

The SPIN model is included in the appendix of this paper. We have run the model checker twice; once with the assertion by the Consumer that every key is fresh taking effect and once with the assertion taking no effect. No assertion violations were found with the assertion on fresh keys commented out. With be fresh keys assertion effective, the scenario of Figures 8, and 9 was found.

Figure 8 shows how a Consumer sends a Request message to a Thief, which in turn sends a Request message to the Producer. The Producer responds with Packaged Content and License message to the Thief, which caches the Packaged Content and License, then sends it along to the Consumer. The Thief takes care to re-encrypt the License with Consumers public key.

Figure 9 shows the Consumer asking for the same content a second time. The Thief has no choice but to resend the cached Packaged Content and License again, and thus gets caught. The price we have to pay for this level of security is that the Clients have to be able to cache all license keys. This requires in principle an unbounded amount of store. However with an appropriate hashing technique the

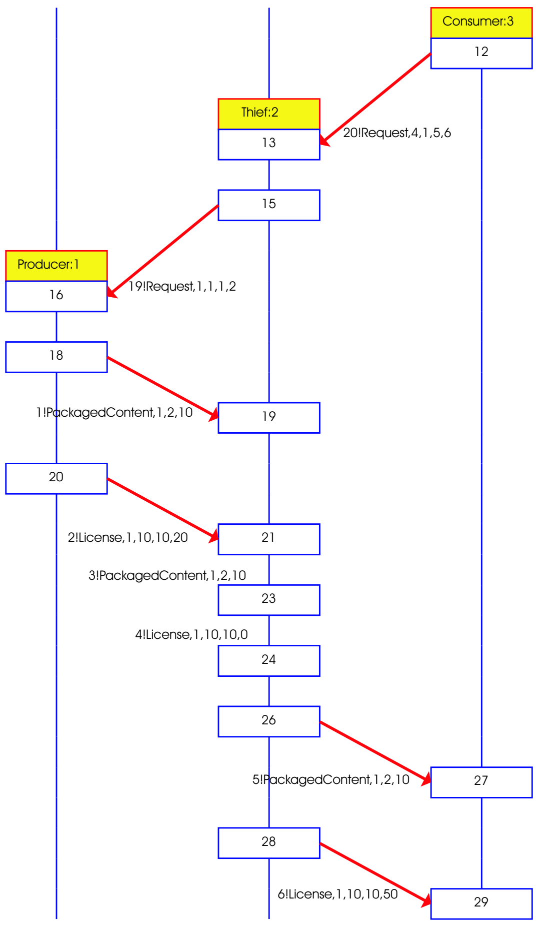

Figure 8: Top half of the message sequence chart, showing that the Consumer acquires some content, using the Thief as middleman.

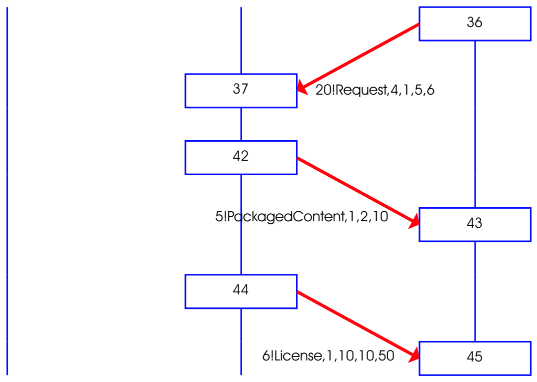

Figure 9: Bottom half message sequence chart, showing the Consumer requesting the same content again.

storage requirements could be curtailed, as hashing the same key twice is guaranteed to give a collision. The question then remains how to deal with falls positives. This is an area for future research.

## 6 Conclusions and Future Work

We propose a Digital Rights Management system for multimedia content that separates the authorities involved in three categories: Identity, Attribute and Right. This separation creates flexibility because each authority has a significant degree of autonomy. The link between the authorities is established using cryptographic means to build a chain of control (*identity-attribute-rights*). Hashing the public key certificate, attribute certificate, licence identity and some random data generates the content key.

We also describe a prototype implementation of the system, with a performance evaluation from a users perspective. We have measured the average time needed for content protection and license generation when simultaneously serving several users, and the average time needed for content protection while varying the size of the content. The measurements indicate that up to a certain load the system scales.

A SPIN model of the system is used to verify that super distribution can be detected when content keys are unique.

The server side DRM of our prototype is document and renderer independent. The prototype uses Adobe Acrobat to render content. An Acrobat specific plug in is responsible for the client side DRM. The DRM client functions as a validator for the chain of control constructed by the server, as a detector for illegal super distribution.

We plan to replace the Acrobat specific client side DRM with a format and renderer neutral DRM client that interfaces conveniently with any application (or plug-in of an application) at the client side. An interesting alternative might be to embed the content in an agent [29]. We also plan to use smart cards to provide tamper-resistant licence store at the client side.

Finally, we contemplate establishing a chain of control on partial digital content, as opposed to the whole content. By doing so, the flexibility of the DRM system can be increased. We believe that this can be achieved by integrating feature extraction for picture, movie and music as well as by implementing the idea proposed by [9] for document and book at the client and server side. In the meantime, we are collecting user's feedback using the system. We believe that the success of deployment of the system depends on the interface with the users.

## 7 Acknowledgements

We thank the SUMMER team for their support and insightful comments on developing the prototype SUMMER, with special thanks to Hiddo Hut for his review of our paper.

## References

[1] F. Hartung and F. Ramme, “Digital rights management and watermarking of multimedia content for m-commerce applications,” *IEEE Communications Magazine*, vol. 38, pp. 78–84, Nov 2000.

[2] D. Henry, “Who's got the key?,” in *Proceedings of the* 27th *annual SIGUCCS Conference on Mile high expectations*, (Denver, Colorado), pp. 106–110, ACM Press, 1999.

[3] J. Linn, "Attribute certification: An enabling technology for delegation and role-based controls in distributed environments," in *Proceedings of the 4th ACM Workshop on role-based access control on Role-based access control*, (Fairfax, Virgina), pp. 121–130, ACM Press, 1999.

[4] B. Horne, B. Pinkas, and T. Sander, "Escrow services and incentives in peer-to-peer networks," in *Proceedings of the 3rd ACM Conference on Electronic Commerce*, pp. 85–94, October 2001.

[5] S. H. Kwok and S. M. Lui, "A license management model to support b2c and c2c music sharing," in 10th *International World Wide Web Conference, Hong Kong*, pp. 136–137, May 2001.

[6] D. Sellars, "An introduction to steganography," tech. rep., University of Cape Town, Computer Science, May 1999. http://www.cs.uct.ac.za/courses/CS400W/NIS/papers99/dsellars/index.html.

[7] J. Dittman, A. Behr, M. Stabenau, P. Schmitt, J. Schwenk, and J. Ueberberg, "Combining digital watermarks and collusion secure fingerprints for digital images," in *IEE Electronics and Communications, London*, pp. 6/1–6/6, 2000. UK ISSN 0963-3308 - Reference No:2000/39.

[8] J. Fridrich, "Robust digital watermarking based on key-dependent basis functions," in *Proceedings of Second International Workshop on Information Hiding, USA*, pp. 143–157, 1998.

[9] S. Brin, J. Davis, and H. Garcia-Molina, "Copy detection mechanisms for digital documents," in *Proceedings of the ACM SIGMOD Annual Conference San José*, pp. 398–409, May 1995.

[10] G. Durfee and M. Franklin, "Distribution chain security," in *7th ACM conference on Computer and communications*, pp. 63–70, ACM Press, New York, 2000.

[11] M. Mourad, J. Munson, T. Nadeem, G. Pacifici, and M. Pistoria, "Webguard: A system for web content protection," in 10th *International World Wide Web Conference, Hong Kong*, pp. 142–143, May 2001.

[12] E. Damiani, S. de Capitai di Vimercati, S. Paraboschi, and P. Samarati, "Design and implementation of an access control processor for xml documents," in *Proceedings of WWW9 Computer Networks*, 1–6, pp. 59–75, 2000.

[13] E. Damiani, S. de Capitai di Vimercati, S. Paraboschi, and P. Samarati, "Securing xml documents," in *Advances in Database Technology - EDBT 2000, 7th International Conference on Extending Database Technology, Konstanz, Germany, March 27-31, 2000, Proceedings*, vol. 1777 of *LNCS*, pp. 121–135, Springer, 2000.

[14] E. Bertino, S. Castano, E. Ferrari, and M. Mesili, "Controlled access and dissemination of xml documents," in *Proceedings 2nd ACM Workshop on Web Information and Data Management (WIDM9́9)*, pp. 22–27, 1999.

[15] M. Kudo and S. Hada, "Xml document security based on provisional authorization," in *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pp. 87–96, 2000.

[16] J. Hunter and W. Crawford, *Java Servlet Programming.* O'Reilly United Kingdom, April 2001. ISBN: 0596000405.

[17] L. Pekowsky, *Java Server Pages.* Addison Wesley, June 2000. ISBN: 0201704218.

[18] J. B. Knudsen, *Java Cryptography.* O'Reilly United Kingdom, 1998. ISBN: 1565924029.

[19] International Telecommunication Union, Place des Nations 1211 Geneva 20 Switzerland, *ITU-T Recommendation X.509, Information Technology, Open Systems Interconnection, The Directory: Authentication Framework*, July 1997. http://www.itu.int.

[20] S. Farrell and R. Housley, *An Internet Attribute Certificate Profile for Authorization, IETF Draft.* PKIX Working Group, January

2001. http://search.ietf.org/internet-drafts/draft-ietf-pkix-ac509prof-09.txt.

[21] B. Gelbord, H. Hut, G. Keinhuis, and E. Kwast, “Access control based on attribute certificates.” Email: {b.s.gelbord, d.h.hut, g.kleinhuis, e.kwast}kpn.com, 2002.

[22] B. Schneier, “Description of a new variable-length key, 64-bit block cipher (blowfish),” in *Fast Software Encryption, Cambridge Security Workshop Proceedings*, pp. 191–204, Springer-Verlag, December 1994.

[23] B. Schneier, *Applied Cryptography, Second Edition*, ch. 15, pp. 357–368. John Wiley & Sons, Inc., 1996. ISBN: 0-471-11709-9.

[24] C. Shi and B. Bhargava, “A fast mpeg video encryption algorithm,” in *Proceedings of the* $6^{th}$ *ACM International Conference on Multimedia*, (Bristol, UK), pp. 81–88, ACM Press, 1998.

[25] F. LaMonica, “Streaming media,” *Linux Journal*, vol. 81es, January 2001.

[26] G. Lowe, “Breaking and fixing the Needham-Schroeder Public-Key protocol using FDR,” *Software Concepts and Tools*, vol. 17, pp. 93–102, 1996.

[27] P. H. Hartel, M. Butler, A. Currie, P. Henderson, M. Leuschel, A. Martin, A. Smith, U. Ultes-Nitsche, and B. Walters, “Questions and answers about ten formal methods,” in *4th Int. Workshop on Formal Methods for Industrial Critical Systems, Vol II* (S. Gnesi and D. Latella, eds.), (Trento, Italy), pp. 179–203, ERCIM/CNR, Pisa, Italy, Jul 1999. www.dsse.ecs.soton.ac.uk/ techreports/ 99-1.html.

[28] G. J. Holzmann, “The model checker SPIN,” *IEEE Transactions on software engineering*, vol. 23, no. 5, pp. 279–295, 1997. http:// cm.bell-labs.com/ cm/ cs/ who/ gerard/.

[29] J.-H. Morin and D. Konstantas, “HyperNews: a MEDIA application for the commercialization of an electronic newspaper,” in *ACM symposium on Applied Computing*, pp. 696–705, ACM Press New York, NY, USA, 1998.

## 8 Appendix - SPIN model

```
/*
Verification takes about 10 mins, at 96 Mbyte, Supertrace
*/


#define NumChaches      4
#define NumServers      4
#define NumRequests     10

#define mkcontent(n)    (2*n)
#define mkkey(p, n)     (NumRequests*(p+1)+n)

#define tprocess        byte
#define tkey            byte
#define tsignature      byte
#define tnumber         byte
#define tcontent        byte


mtype = {Request, PackagedContent, License,
         NoPackagedContent, NoLicense} ;

/* Server input channels (Producer or Thieves) */
chan    ch[NumServers] = [0]
        of {mtype, tprocess, tnumber, chan, chan} ;

#define pp  0


proctype Producer(tprocess mp) {
    tnumber     n ;
    tcontent    c ;
    tkey        dk ;    /* Content encryption key */
    tkey        lk ;    /* License encryption key */
    tsignature  sk ;    /* License signature key */
    tprocess    rp ;    /* Client process id */
    chan        rod ;   /* Content output channel */
    chan        rol ;   /* License output channel */
    byte        cnt ;   /* Generate unique content key */

    cnt = 0 ;
end:do
    ::  ch[mp]?Request(rp, n, rod, rol) ;
        if
        ::  cnt < NumRequests ->
                dk = mkkey(mp, cnt) ;
                cnt++;
                c = mkcontent(n) ;
                sk = mkkey(pp, 0) ;
                lk = mkkey(rp, 0) ;

                rod!PackagedContent(n, c, dk) ;
                rol!License(n, dk, sk, lk)
        ::  else ->
                rod!NoPackagedContent(0, 0, 0) ;
                rol!NoLicense(0, 0, 0, 0)
        fi
    od
}


proctype Thief(tprocess lp; tprocess mp) {
    tnumber     n ;
    tcontent    c ;
    tkey        dk1 ;   /* Content encryption key */
    tkey        dk2 ;
    tkey        lk ;    /* License encryption key */
    tsignature  sk ;    /* License signature key */
    tprocess    rp ;    /* Client process id */
    chan        rod ;   /* Content output channel */
    chan        rol ;   /* License output channel */
    /* input channel from server */
    chan    lid = [0]
            of {mtype, tnumber, tcontent, tkey} ;
    chan    lil = [0]
            of {mtype, tnumber, tkey, tsignature, tkey} ;

    /* Cached document and license queues */
    chan    qd = [NumChaches]
            of {mtype, tnumber, tcontent, tkey} ;
    chan    ql = [NumChaches]
            of {mtype, tnumber, tkey, tsignature, tkey} ;

end:    do
    ::  ch[mp]?Request(rp, n, rod, rol) ;
        if
        ::  qd??[PackagedContent(eval(n), c, dk1)] ->
                qd??<PackagedContent(eval(n), c, dk1)> ;
                ql??<License(eval(n), dk2, sk, _)> ;
                assert(dk1 == dk2) ;
                lk = mkkey(rp, 0) ; /* (re) encrypt */
                rod!PackagedContent(n, c, dk1) ;
                rol!License(n, dk2, sk, lk) ;
        ::  else ->
                ch[lp]!Request(mp, n, lid, lil) ;
                if
                ::  lid?PackagedContent(n, c, dk1) ->
                        lil?License(n, dk2, sk, lk) ;
                        assert(dk1 == dk2) ;
                        assert(sk == mkkey(pp, 0)) ;
                        assert(lk == mkkey(mp, 0)) ;
                        qd!PackagedContent(n, c, dk1) ;
                        ql!License(n, dk2, sk, 0) ;
                        lk = mkkey(rp, 0) ; /* re-encrypt */
                        rod!PackagedContent(n, c, dk1) ;
                        rol!License(n, dk2, sk, lk)
                ::  lid?NoPackagedContent(_, _, _) ->
                        lil?NoLicense(_, _, _, _) ;
                        rod!NoPackagedContent(0, 0, 0) ;
                        rol!NoLicense(0, 0, 0, 0)
                fi
        fi
    od
}


proctype Consumer(tprocess mp) {
    tnumber     n ;
    tcontent    c ;
    tkey        dk1 ;   /* Content encryption key */
    tkey        dk2 ;
    tprocess    lp ;    /* Server process id */
    tkey        lk ;    /* License encryption key */
    tsignature  sk ;    /* License signature key */

    /* input channel from server */
    chan    lid = [0]
            of {mtype, tnumber, tcontent, tkey} ;
    chan    lil = [0]
            of {mtype, tnumber, tkey, tsignature, tkey} ;

    /* Cached document queues */
    chan    qk = [NumRequests] of {tkey} ;

end:do
    ::  /* Non-deterministic choice of document number */
        if
        ::  n = 1
        ::  n = 2
        ::  n = 3
        fi ;
        /* Non-deterministic choice of server */
        if
        ::  lp = pp
        ::  lp = 1
        ::  lp = 2
        ::  lp = 3
        fi ;
        ch[lp]!Request(mp, n, lid, lil) ;
        if
        ::  lid?PackagedContent(n, c, dk1) ->
                lil?License(n, dk2, sk, lk) ;
                assert(dk1 == dk2) ;
                assert(sk == mkkey(pp, 0)) ;
                assert(lk == mkkey(mp, 0)) ;
                assert(c == mkcontent(n)) ;
                assert(! qk??[eval(dk1)]) ; /* Comment line out */
                if
                ::  full(qk) ->
                        break
                ::  nfull(qk) ->
                        qk!dk1
                fi
        :: lid?NoPackagedContent(_, _, _) ->
                lil?NoLicense(_, _, _, _) ;
                break
        fi
    od
}


init {
    atomic {
        run Producer(pp) ;
            run Thief(pp, 1) ;
                    run Consumer(4) ;
            run Thief(pp, 2) ;
                run Thief(2, 3) ;
                    run Consumer(5) ;
    }
}
```