PERSISTENT PROTECTION IN MULTICAST CONTENT DELIVERY

Malek Barhoush

A thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy

Concordia University

Montréal, Québec, Canada

May 2012

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Malek Barhoush**

Entitled: **Persistent Protection in Multicast Content Delivery**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Dr. A. Aghdam _____ Chair

Dr. K.C. Almeroth _____ External to Program

Dr. M. Debbabi _____ Examiner

Dr. H.F. Li _____ Examiner

Dr. L. Narayanan _____ Examiner

Dr. J.W. Atwood _____ Thesis Supervisor

Approved by _____

Dr. V. Haarslev, Graduate Program Director

November 23, 2011 _____

Dr. Robin A.L. Drew, Dean

Faculty of Engineering & Computer Science

Abstract

Persistent Protection in Multicast Content Delivery

Malek Barhoush, Ph.D.

Concordia University, 2012

Computer networks make it easy to distribute digital media at low cost. Digital rights management (DRM) systems are designed to limit the access that paying subscribers (and non-paying intruders) have to these digital media. However, current DRM systems are tied to unicast delivery mechanisms, which do not scale well to very large groups. In addition, the protection provided by DRM systems is in most cases not persistent, i.e., it does not prevent the legitimate subscriber from re-distributing the digital media after reception.

We have collected the requirements for digital rights management from various sources, and presented them as a set of eleven requirements, associated with five categories. Several examples of commercial DRM systems are briefly explained and the requirements that they meet are presented in tabular format. None of the example systems meet all the requirements that we have listed. The security threats that are faced by DRM systems are briefly discussed. We have discussed approaches for adapting DRM systems to multicast data transmission.

We have explored and evaluated the security protocols of a unicast distribution model, published by Grimen, et al. that provides "persistent protection". We have found two security attacks and have provided the solution to overcome the discovered attacks. Then we have proposed a more scalable architecture based on the modified model. We call the resulting architecture persistent protection in multicast content

delivery. We present and formally validate the protocol for control and data exchange among the interacting parties of our proposal.

Acknowledgments

I am extremely thankful to God for helping me and guiding me to the next stage of my life.

I want to express my deepest appreciation and thanks to my supervisor and professor, Dr. J. W. Atwood, for his support and patient guidance during my entire research period. He helped me to successfully complete this stage of my life.

Also, I want to thank Dr. Mourad Debbabi for helping me overcome many difficulties in my research.

Also, I want to express my great thanks to the committee members for their feedback and for making my work stronger. A special thanks goes to Dr. Suleiman Hussein Mustafa who encouraged me to complete my studies. A special thanks goes to Yarmouk University for giving me the chance to continue my degree and for supporting me.

I cannot forget to thank my friends who love me and help me move forward.

My special thanks goes to my father, parents in law, wife and kids for their unconditional love, support, and encouragement and for praying for me to finish this thesis.

Contents

List of Figures

List of Tables

List of Acronyms

AAAS	Authentication, Authorization and Accounting Server
AVISPA	Automated Validation of Internet Security Protocols and Applications
CEK	Content Encryption Key
CP	Content Provider
CS	Content Server
DEK	Data Encryption Key
DRM	Digital Rights Management
DRMA	DRM Agent
EEMD	Encrypted Encoded Media Document
EU	End User
FI	Financial Institution
GCKS	Group Control and Key Server
IETF	Internet Engineering Task Force
LP	License Provider
MG	Mobile Guard Agent
MP	Mobile Protection Agent
MR	Merchant
MSA	Multicast Security Architecture
MSEC	IETF Multicast Security Working Group
MSP	Mobile Security Provider
NSP	Network Service Provider
PPMCD	Persistent Protection in Multicast Content Delivery
PS	Policy Server
SS	Security Server
VOD	Video on Demand
VS	Viewer Software

Chapter 1

Introduction

1.1 Motivation for the Research

The phrase "Digital Content Distribution" describes the distribution, using the Internet, of intellectual property from a Content Provider to one or more End Users. Most contemporary Content Distribution systems establish a one-on-one (unicast) relationship between the Content Provider and the individual End Users. In addition, the delivery of the intellectual property is usually done "on demand", i.e., delivery happens when the End User wants it to happen.

However, the Internet is a very open, insecure medium. Few owners of intellectual property will be interested in using this medium for distribution if their intellectual property can be copied and re-distributed during delivery. For this reason, Digital Rights Management (DRM) systems have been developed to manage the distribution, and protect the rights of the Content Provider against the actions of malicious intruders and the malicious actions of legitimate End Users.

DRM systems need to provide two kinds of protection:

1) Within the delivery medium, it is necessary to ensure that a random malicious intruder cannot access the digital content. This is "protection before delivery".
2) Within the End User's device, it is necessary to ensure that the media stream cannot be captured and subsequently shared with the End User's friends or clients. This is "protection after delivery".

Protection before delivery is normally provided by cryptographic mechanisms. The details of such mechanisms are outside the scope of this thesis. Protection after delivery is normally provided by DRM systems. Ideally, this protection will be "persistent", i.e., it will last beyond the point in time where the End User has finished using the intellectual property. A more precise definition of our use of "persistent protection" will be given later in this thesis.

The one-on-one nature of contemporary Content Delivery systems is resource-intensive for the Content Provider. As the number of customers grows, the effort required to manage them and to deliver the intellectual property increases linearly, eventually reaching the point where it is difficult to service all of the requests. For certain events (e.g., world-level sporting events such as the Olympics, soccer tournaments, cricket tournaments, etc.), it may become impossible to provide the expected Quality of Service to the End Users. In addition, the Network Service Provider may have difficulty providing sufficient bandwidth to the Content Provider to service all of the requests.

If we take as an example the delivery of video, as the number of End Users demanding a particular video increases, the disadvantage of making them wait a short period of time before the presentation starts gets smaller. If the video is for a real-time event, then of necessity all End Users need the identical video stream. In these two cases, multicast data distribution (one-to-many relationship) can provide significant resource savings to both the Content Provider and the Network Service Provider. However, standard Internet Protocol multicast has neither security nor DRM.

The initial focus of our work was to answer the questions:

1) what requirements for protection of digital intellectual property are difficult or impossible to meet when multicast data distribution is used?
2) what approaches can be used to mitigate these difficulties?

However, we discovered that there appeared to be no list of requirements for enforcing Digital Rights Management in the multicast systems. In addition, the available lists for unicast-based systems were narrowly focused. Therefore, we will begin our journey by studying in detail many DRM systems such as Microsoft DRM and Open Mobile Alliance (OMA) Digital Rights Management (DRM) 2.0. Next, we will collect and adopt a list of requirements to achieve the desired benefits for content media protection in the unicast case. Then we will find out what requirements are difficult or impossible to meet when multicast data distribution is used. Throughout this thesis the term "content media" is used uniformly to refer to "digital intellectual property", "media document", "digital content" and "digital multimedia", which are terms used by other authors to refer to the same concepts.

1.2 Problem Statement

The main goal of this thesis is to design a flexible mechanism, architecture and protocols, for scalable, scheduled and persistently protected content media delivery.

1.3 Objectives

The main objectives of this research are the following:

- Harden and increase the security of the protection applications that are used to control the content media access after it was been delivered.

- Improve the performance and scalability of current DRM systems by changing the underlying distribution mechanism from unicast into multicast.

1.4 Contributions

We have the following contributions:

- We provide a comparative study of the state-of-the-art proposals in secure and scalable content delivery.
- We provide a requirement analysis for DRM in multicast content delivery.
- We identify the sufficient requirements to enforce persistently protected content delivery.
- We provide security analysis of Grimen, et al.'s model.
- We elaborate an extension to Grimen, et al.'s model that fixes a vulnerability.
- We propose and validate an architecture that assures flexible, scalable and persistently protected content delivery.

1.5 Thesis Organization

In Chapter 2, we provide more details about the content media distribution model, beginning with an introduction to the related benefits of the public content media distribution. Then, we discuss the security issues with content media delivery, followed by a description of the DRM system's basic architecture. Next, we study different types of DRM systems and their limitations. Persistent protection and some information about Java mobile code are included in this chapter. Finally, we present the

DRM attack model. Chapter 3 describes how multicast content media distribution works by providing IP multicast as an example. We show the advantages and disadvantages of IP multicast, analyze IP multicast scalability factors, and describe two secure multicast models: Multicast Group Security and Multicast Security Architecture. We propose the problem statements in Chapter 4; there, we show the objectives of our work, and then we draw the road map to address the DRM and secure multicast problems. In Chapter 5, we start our road map to the proposed solution for the persistent protection in multicast content media delivery by proposing the requirements for securing both unicast and multicast distributions with persistent protection of the delivered media. Before proposing our solution, we go through a simple proposal by Grimen et al. in Chapter 6. It illuminates a promising solution for achieving persistent protection for delivered content media. We validate their proposal and find a hole, for which we propose a solution, and then, based on the model of Grimen et al., we propose and validate an improved persistent protection model. In Chapter 7 we propose the solution for secure multicast distribution with persistent protection; also, we propose the interaction protocol between main roles of the proposed solution, and then validate the protocol. Finally, in Chapter 8, we propose an architecture for a more distributed, scalable and presistent protection of the delivered content media. We finish our discussion with our conclusion and future work in Chapter 9.

Chapter 2

Content Distribution Model

If the Internet is to be used to distribute content media, it is necessary to provide control over this distribution, to ensure that only legitimate End Users can access it, and only in permitted ways.

In this chapter, we will discuss the services needed to protect the information, and the architectures of Digital Rights Management (DRM) systems that claim to provide this protection. We will focus our attention on the idea of persistent protection, which is one of two areas of concentration for the work of this thesis. Finally, we will give the attack model for security threats against DRM systems.

2.1 Public Content Distribution

Using the Internet for distributing multimedia contents without any control over the distributed media is called public distribution. Any user can use public digital content anytime anywhere as long as the link to the digital content is provided and s/he has the suitable device that is able to render the digital content [Sys, Che08]. The meaning

behind free content is: there are no restrictions on the use of free content after being delivered, so the end user has the right to copy the content and reuse it an unlimited number of times and s/he can redistribute it to his/her friends, or worse, he may be able to sell it without getting the permission from the content provider or content owner [Mar].

In the past, producing physical content media was done by a lot of expensive technologies. The next stage is to distribute these media. The way of distribution was through physical media, which gives the publisher some ways of control to protect physical media, but in the same time it increases the cost of goods, as well as, the beneficiaries of the goods get paid less. The enormous development and progress in the field of networks, as well as providing the possibility to convert many non-digital products to digital products, and the spread of the Internet and the possibility of buying and selling via the Internet, all have led many investors to think about using the Internet to promote the way of distrbuting content media [BA].

Public network which works well with free content distribution is not a good choice for the investor. They are looking for suitable controlled environment for distributing the content media. Digital investors are looking for controlling the media distribution as well as controlling how the media are used. Cryptographic technologies are the efficient way to control media distribution using a public network, Section 2.2 will give a good description on that. Controlling the content media to reduce the number of times to read the content media, restrict the operations allowed on the content media, or select a time to use the content media cannot be achieved by using encryption technologies only. Section 2.3 will describe these issues.

2.2 Security Issues for Content Delivery

The role of the content provider (CP) is to convert analog content media into digital media. The end user (EU) is the customer who is interested in receiving content media, and the distribution mechanism is the Internet. The CP needs to control the delivery of content media to those who are eligible to receive it. The EU eligibility is determined by the ability of a customer to pay for the use of the content. One of the most efficient ways to achieve such control via the public network is to use cryptographic technologies.

Cryptography may be defined as a way of encrypting or hiding plain text using mathematics to produce protected information or ciphertext. The encryption process requires key(s). The strength of the encryption technology depends on hiding the key, not the algorithm. At the same time, if an unauthorized person tries to decrypt the cipher text without knowing the key, we call this process cryptanalysis. Cryptography technology covers secrecy, authenticity, integrity and non-repudiation services [BP82, BEM+07].

A secrecy or confidentiality service allows two or more persons to securely exchange information without unauthorized disclosure. An authenticity service guarantees the user that the received message and its source are reliable; at the same time, this service assures the sender that the end user is reliable as well. An integrity service provides both the sender and the receiver with confidence that the received message was not altered along the communication path. A non-repudiation service provides all parties related to a specific transaction the assurance that none of them can deny having participated in the transaction [Sta03].

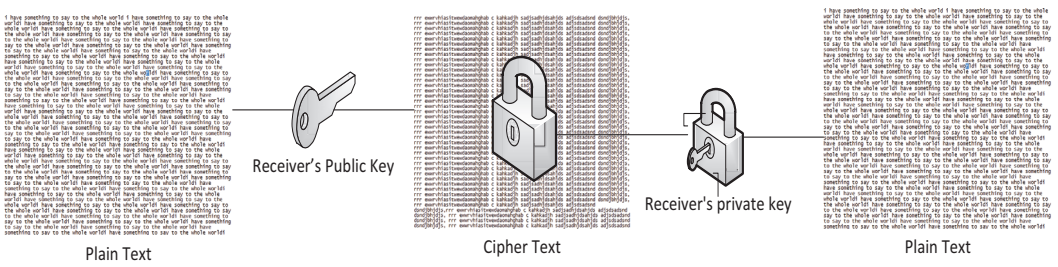

Figure 1: Asymmetric key encryption

2.2.1 Confidentiality and Secrecy

Confidentiality or secrecy is achieved when two parties are able to protect information exchanged between them without allowing others to know what has been negotiated, in other words, the secrecy property is concerned with preventing unwanted people from snooping around the channel established between two or more parties. In the Internet world, the secrecy property is achieved using cryptosystems; "the encryption and decryption [process] is called a cryptosystem" [BEM+07]. The sender encrypts the message using what is called an encryption key. Then, the receiver decrypts the protected message using the decryption key. The encryption and decryption procedures fall under two major models depending on what type of keys they use:

- Asymmetric or Public-key cryptosystems: In the Asymmetric encryption and decryption models, a public-key cryptosystems such as RSA, every user has two keys, one public key and one private key. Usually, the public key is known to the whole world, while the private key must be kept in a private place and should not be disclosed to anyone but its owner. In fact, the success of this model depends on keeping the private key hidden. To achieve secrecy, the receiver's public key is used to encrypt the message, and the receiver's private key is needed for the decryption process. Since the private key is only known by one person, its owner, the encrypted message can only be decrypted by the owner [Sta03], see Figure 1.

If Alice and Bob want to talk with each other using a public transporter, and Eve is listening to their talk, then, in order to prevent Eve from understanding their conversation:

- Alice produces and encrypts the messages she wants to send, using Bob's public key.
- Alice sends the encrypted message via public transporter.
- Bob receives and decrypts the protected message coming from Alice using his private key.
- Eve can receive the protected message, but she cannot see its content because she does not know Bob's private key.
- Since only Bob knows his private key, this fact partially provides the authenticity of the receiver.

Theoretically, the asymmetric model is computationally more secure than the symmetric model.

- Symmetric cryptosystems: In the symmetric encryption, both the sender and the receiver share the same key called secret key; both of them use the same encryption and decryption algorithm as well. This type of secrecy model depends on keeping the secret key protected from unwanted parties. In symmetric cryptosystems, the sender uses the secret key to encrypt the message he wants to send resulting in cipher data, and then sends the cipher data into the receiver. On the other side of the channel, the receiver receives the cipher data and uses the same secret key along with a corresponding decryption algorithm to decrypt the cipher data resulting in the original message [Sta03].

  If Alice and Bob want to talk with each other using a public transporter, and Eve is listening to their talk, then, in order to prevent Eve from understanding their conversation:

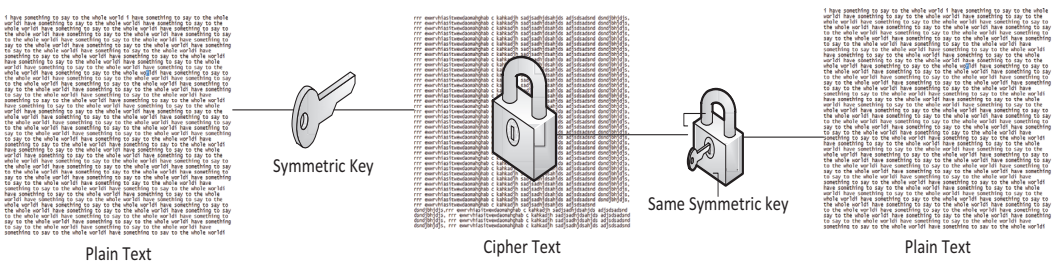

Figure 2: Symmetric key encryption

- Alice produces and encrypts the messages she wants to send, using the shared secret key.
- Alice sends the encrypted message via public transporter
- Bob receives and decrypts the protected message coming from Alice using the same secret key.
- Eve can receive the protected message, but she cannot see its content because she does not know Alice and Bob's shared secret key.
- Since only Alice and Bob know the secret key, this fact provides the authenticity of both the sender and the receiver.

The symmetric model is computationally faster and more efficient than the symmetric encryption model. The most important problem of this model is that the secret key distribution is not flexible, as well as, the symmetric encryption model is less secure than the asymmetric key. See Figure 2.

2.2.2 Hash Function and Authentication Service

Hash function is a one way function where its input is an electronic message regardless of its length, and its output is a message digest of fixed length. The digest length ranges between 128 bits and 160 bits. Theoretically, the result of a hash function is unique for each different message. The message digests could be used as a unique identifier for messages, and that helps the receiver to authenticate received messages. See Figure 3.

If Alice and Bob want to talk with each other using a public transporter, and they want to be assured that the talk integrity is not violated by Eve who is listening to their talk and can intercept and forward the talk, then, in order to prevent Eve from violating their conversation:

- Alice produces the hash value of the messages she wants to send.
- Alice encrypts the hash value with the secret key and then sends the encrypted hash value along with the message via public transporter.
- Bob receives the protected hash value along with the message and then decrypts the protected hash value using the same secret key.
- Eve can receive the protected hash value, but she cannot see its content and she cannot modify the hash value because she does not know Alice and Bob's shared secret key.
- Bob generates the hash value of the received message and then compares the result with the hash value received from Alice. If both are identical, then that is an indicator that the message integrity is not violated and that the sender is Alice. In this case, both the message and the sender are authenticated.
- Since Alice or Bob only can encrypt hash value using their secret key, then this kind of process provides the authenticity of the source of the message.

Another scenario using public key and hash value for the purpose of authentication service:

- Alice produces the hash value of the messages she wants to send.
- Alice encrypts the hash value with her private key and then encrypts the result with Bob's public key. Afterward, she sends the encrypted hash value along with the message via public transporter.

- Bob receives the protected hash value along with the message and then decrypts the protected hash value using his private key, then again, decrypts the result with Alice's public key.
- Eve can receive the protected hash value, but she cannot see its content and she cannot modify the hash value because she does not know Bob's private key.
- Bob produces the hash value of the received message and then compares the result with the hash value received from Alice. If both are identical, then that is an indicator that the message integrity is not violated and that the sender is Alice, as well as, the receiver is Bob. In this scenario, the message, the receiver and the sender are authenticated.
- Since Alice only can encrypt hash value using her private key, this process called signing the hash value, then this kind of process provides the authenticity of the source of the message.

In the previous two scenarios, if we need to apply confidentiality service, the message also needs to be encrypted as has been discussed in Section 2.2.1.

2.2.3 Nonrepudiation Service and Digital Signatures

This service is essential for electronic transactions where both ends, the sender and the receiver, need undeniable evidence that the transaction is complete and not forgeable. Digital signature is a mechanism achieved by encrypting a message, which needs to be signed, by the sender's private key. Since the private key is known only to the owner of that key, this gives a proof that the signature is done by only the owner [Lou00, Sta03]. In other words, "The recipient of a signed message has proof that the message originated from the sender" or "the recipient can verify that the message came from the sender" [RSA78].

Since encryption and decryption using public key infrastructure is a heavily loaded processes, and the result of a hash function is a fixed length message

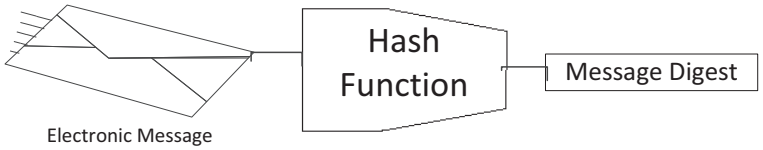

Figure 3: Hash function

digest (MD) that represents the original message and usually the MD is shorter that the original message, signing the MD is an efficient way to provide non-repudiation service. The receiver of the signed MD usually receives the original message that is encrypted via some symmetric-key algorithm, so s/he can produce the MD of the original message and validate with the received MD [Sta03].

2.2.4 Digital Signature Certificates

Digital signature certificates provide a flexible trust framework. Usually, digital signature certificates are issued based on public key infrastructure. The most important role in this trust framework is the certificate authority (CA), who is responsible for issuing and managing Digital Certificates. A digital certificate contains a customer's public key, date of issue, name of the customer, certificate expiry date and other information. The customer could be the end user, a company, the content provider, the merchant and so on. The certificate is signed by the CA, who is publicly known. The certificate authority's public key should be available; VeriSign organization is an example of a CA [MRB01].

One single trusted third party works as global or root CA, and the whole world trusts that entity. Of course, for availability purposes, that single trusted certificate

authority needs to be distributed. The global certificate authority may issue certificates for other certificate authorities and thus build a hierarchical framework for trust service.

Due to the fact that the CA's public key is universally known, customer's certificates can be easily validated by checking the expiry date of the certificate. The customer's public key can be extracted from the customer's certificate, which helps to build a trust relationship between customers and to secure the path between them.

In the case of exposing the customer's private key, the corresponding certificate needs to be invalidated; the CA periodically publishes the certificate revocation list (CRL).

2.3 DRM Systems

In the early years of producing content, the relationship between the content owners and the content consumers was based on physical objects, e.g., books. The content publisher, who was responsible for publishing these books, would try to prevent consumers from compromising this service and producing illegal copies. If s/he used special paper that prevented copiers from producing (illegal) high quality books, the protection of the content owner was somehow assured [Coy03].

The increasing reliability of the Internet and the advanced technologies used to generate digital multimedia have changed the distribution methods for multimedia content from physical forms into digital forms. Examples of digital multimedia content (e-audio, e-video, e-image, e-book, etc.), "e" stands for electronic. This new technology draws intelligent artists' attention, converting their "tangible" [PBW02] intellectual property into equivalent digital forms and then advertising their innovations to the whole world at small cost, knowing that millions of customers can easily

connect to the Internet and ask for the content media. The fact that many customers are attracted to get the content media using easy connections cheaply, will increase the demand on the content distribution service. Flexibility in this context means that the end user only needs to use his own machine to access digital content rather than going to a theater or a digital store to search for specific media and then watch it.

Compared with a content distribution service for "tangible" intellectual property, a digital service has the potential to increase the content producer's profit. However, it has the disadvantage that a person (paying subscriber or not) can get a copy of the content and start to re-distribute it. This has led to the idea of generating Digital Rights Management (DRM) systems, which are intended to protect the content producer's rights to distribute the content, and thus retain his/her profits.

Content distribution has traditionally been based on a one-to-one relationship between the content provider and the end user. These two parties agree (implicitly or explicitly) on the mechanism(s) to be used (in the content server, on the wire, and in the receiving host) to protect the digital content from various threats, whether they come during data transmission, or after the data have arrived at the receiver.

The management of the digital rights has been the direct responsibility of the content provider. The resources of the Network Service Provider have been used solely to "move the data". The only negotiation required between the content server and the network has been to ensure that the necessary resources are available to deliver the required Quality of Service, using, for example, the Resource Reservation Protocol (RSVP) [RB97].

Some rights are applicable for digital assets/roles and others are not. In the following lines, we will show examples of some digital assets/roles and their applicable rights:

- License (create, modify, distribute, redistribution).

- Digital content media (offline/online play, replay, modify, forward, super-distribution)
- Roles (CP, EU, LP, monitor).
- Digital document (view, write, print, delete, forward)
- Teleconference (join, leave, add members, delete members, authenticate)

2.3.1 DRM Architecture

The generic DRM architecture consists of three players: content provider (CP), license provider (LP) and end user (EU), see Figure 4. The CP is mainly in charge of generating the content media, then protecting the content media by encrypting it using well known encryption algorithms. In some cases, the CP may use a proprietary or closed encryption algorithm, and that may not be acceptable in the commercial world. Usually, for commercial use, cryptography technology hides encryption and decryption keys but not encryption and decryption algorithms. We strongly believe that in the military use, both encryption and decryption algorithms as well as encryption and decryption keys are hidden[Sav02]. CP generates meta-data, which contains some useful information such as the place where to get the encrypted media, which algorithm to decrypt the content media and where to obtain the decryption key, and so on. The CP attaches meta-data along with each encrypted content (protected content). The meta-data guides the consuming device to the location of the LP, i.e., where to acquire a license. The CP provides the LP with corresponding content encryption keys (CEK). The LP is mainly responsible for creating permissions (licenses), which include terms and conditions, as well as the CEK for enabling the consuming device to expose the corresponding hidden content. EU downloads the hidden content via local software called a DRM agent (DA), which is designed to enforce usage policies. The DA extracts the information pointing to the LP from the meta-data, negotiates with the LP for providing licenses according to user's payment amount, downloads the

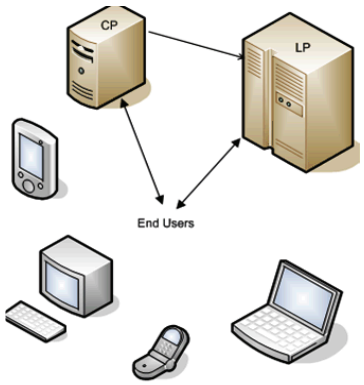

Figure 4: DRM generic interactions

license, checks the integrity and the validity of the license, interprets the license, extracts the CEK and enforces the terms and conditions [Ltd02, Ars, OMA08b, Cor08].

In most of the DRM systems, hidden contents can be publicly reached either from the CP or via another peer device (super-distribution). However, the license file that allows the completion of the rendering process for any distinct content must be paid for. Therefore, controlling and managing the license helps the content owners.

DRM technology is deployed in three levels (application, operating system and hardware [AH04]). In Section 2.3.2, we will talk about two successful DRM products belonging to Microsoft, which deploy DRM in operating system and application level: Windows media rights manager (WMRM) and the successor Windows Rights Management Services (RMS). Then we will go through a successful DRM system, mainly applied for mobile phones, Open Mobile Alliance (OMA), which deploys DRM at both application and hardware level.

2.3.2 Different types of DRM systems

Windows media right manager (WMRM) is a Microsoft tool that allows the CP to protect any content media, and it allows him to send and receive the protected content media using the public network. The CP, with the help of WMRM, encrypts

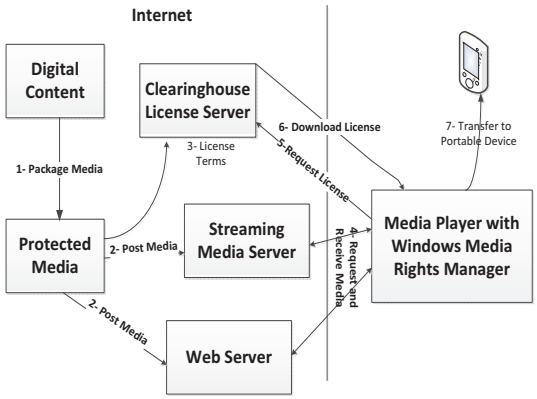

Figure 5: Windows media rights manager flow [Cor04a].

the content media with a selected key, then the encrypted media is packaged with useful information such as media version. The resulting protected content is ready to be distributed and delivered to any client via a distributor server. See Figure 5. The client who wants to watch any protected content, needs to get a license from the clearing house and licensing server, which contains license key seed and key id, for more information see [Cor04a]. Rights Management Services (RMS) is a product developed by Microsoft as well, there are some differences in the two products.

WMRM provides an economical and feasible solution for hiding the digital media; it does not need any special hardware to hide the content. In contrast, RMS may need such hardware. Both technologies (we will call them Microsoft DRM or MSDRM) support multiple access control models, such as:

- The number of times the end user is permitted to use the protected media [Cor04a].
- The starting time to use the content media [Cor04a].
- Is the end user allowed to copy the content media [Cor04a].
- "Document expiration" [MT08].

- "Access content programmatically" [MT08].

Microsoft released operating systems that allow Microsoft DRM to be run through a variety of devices such as personal computers, notebooks, PDAs, smart-phones and pocket PC [Cor04b, Cen]. WMRM as well as RMS are end-to-end DRM solutions. They used cryptographic mechanisms to securely distribute the content media. When a client receives any protected media, s/he cannot directly decrypt it, the client needs to use what is called DRM application, which is responsible for contacting the clearing house and license server and get a license, which is capability given to an eligible client to get access to a protected content. The application DRM afterward is capable of decrypting protected content and rendering it [Cor04b, Cen].

Microsoft builds a secure environment via software application; application uses proprietary mechanisms that encrypt and decrypt content media. Each end-user virtually has a unique instance of that application, this process is called individualization. The application uses a secure path to the hardware driver used to render the content media. If an instance of this proprietary application is hacked a revocation process is used to revoke that instance. This kind of individualization supports machine authentication [JPKJ06, Arn07]. There is no provided information that tells us that WMRM supports user authentication but RMS does [16w, Ros05].

For each enterprise, there is a certified RMS server used for registration purposes, the RMS system considers this server to be a root server. The registering RMS server signs up each client's device; it has the chance to register other servers. In the Microsoft RMS system, each client needs to use a DRM controller and his account certification in order to enable DRM. RMS may authenticate enterprise internal users as well as external users (users who do not belong to the same enterprise) as long as they use either active directory server or a .NET Passport account [Ros05].

MSDRM uses the "individualization" technique, which generates a unique instance of the software player, and binds each instance to a specific customer machine,

therefore, each player is supposed to work only on a specific machine. It also supports revocation service as counterattack if individualized software instance is compromised. More information on these two services is available on the Microsoft website [Mic].

The WMRM player is software and it is susceptible to modification or replacement attacks. These attacks are achieved by obstructing or modifying the enforcement part of the rendering code with an attacker-made code, and thus bypassing the checking points. Another attack was created by one software cracker. He analyzed the WMRM code and then produced a tool called "FreeMe", which tracks the location of encryption keys located inside the blackbox file (used to hide these keys) and then exposed hidden media files [Scr01, JM07]. WMRM does not provide real privacy preservation for end users; neither does RMS (R6). There is no reported attack against the RMS system but the behavior of the system indicates that it is susceptible to a software reverse engineering attack.

The Open Mobile Alliance DRM-2 (OMA-DRM-2) [All, Irw04] is a specification and standard designed for enabling the control of digital services on different mobile phones and personal players. OMA DRM2 architecture has three major components: content issuer (CI), rights issuer (RI) and a DRM agent (DA). The CI generates a content encryption key (CEK) either for each individual content medium or multiple contents, and may encrypt selective contents. It then packages each of them in a secure container; this container is grouped and packaged into the DRM Content Format (DCF). The DCF may contain more than one container, Figure 6 shows the DCF structure [All]. OMA-DRM-2 supports a small-size content DCF (picture, ring and small messages) as well as a large-size (audio and video content), they call it Packetized-DCF (PDCF). The CI negotiates the rules and constraints for DCF usage with RI. CI delivers DCF to customer machine via various transport mechanisms, s/he does not need to use a secure connection since the DFC is already secured.

When a customer browses a digital catalog, selects interesting media to play, reads

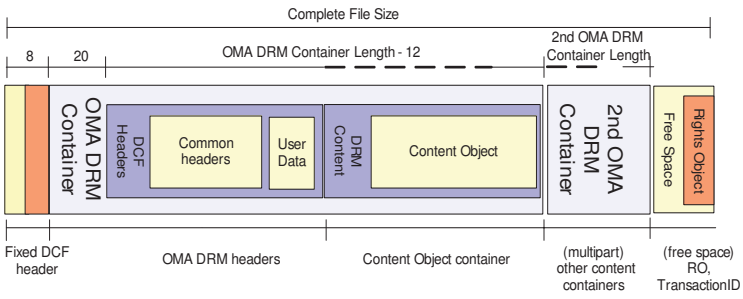

Figure 6: Data content format structure [All, Cor08]

and agrees on terms and conditions and the price for consuming that content, the SIM card that is attached to the mobile phone authenticates the user. Afterward, the DA, which plays the tamper resistant role residing in a mobile station, requests the protected content. DA downloads a DCF, checks its integrity and extracts the information that triggers it to send a rights request to the RI. When it receives the rights object (RO), it verifies the authenticity of RI and RO as well as RO's integrity, all authentication activities happen through rights object acquisition protocol (ROAP) [Irw04]. By then, DA extracts the keys (KEK) from RO and decrypts the protected contents within DCF/PDCF.

The LP creates a suitable rights object (RO) for each DCF. The RO works like a license. When the DA requests an RO, the LP authenticates the DA and protects the RO, which is achieved by encrypting the part containing KEK with target DA's public key and then signing the RO. This means if an adversary accesses this RO, s/he cannot access the KEK because of not having the corresponding private key. The RO is an XML file containing DCF encryption keys and expresses the rules and constraints for using the DCF as being expressed by the CI.

The DA enables the content rendering process and controls its usage rules. The DA is a trusted component in the mobile phone; it has a unique public/private key and a certificate [MST05, Med03], which helps the LP to authenticate the DA. The DA is designed in a way that it should receive both DCF/PDCF and the associated rights object in order to render protected content; it checks and governs the treatment

of the DRM content by enforcing the rights stated in the rights object. The keys in the Rights object are encrypted with the DA's public key. This process binds the rights object to a specific DRM agent and only that target agent can expand the encryption key out of the rights object. It is possible that DA redistributes a protected container to another friend's machine (super-distribution), the receiving machine's DA will start a new RO acquisition process for the received DCF. This super-distribution decreases the extensive overload on CP and improves the availability of the service.

OMA DRM-2 supports the domain concept, which allows the sharing of RO to a group of domain registered devices; this allows them to view the same contents using the shared RO.

Mobile phones have unique embedded proprietary hardware specifications, and each user has to use a special smart card, which supports the device with an address number, therefore, user/device identification, authentication and payment are reliable. Embedded hardware works as a tamper-resistant hardware and it provides the trust to the LP [KC04, MD03].

Internet Stream Media Alliance Encryption and Authentication (ISMACryp) is successfully used for trading by generating a controllable streamed service for high quality media content such as video and audio. The main purpose of this service is to preserve interoperability, especially when DRM is applied to the ISMA scheme. ISMACryp is being built in the application layer [Doe07]. The ISMA architecture consists of four parties: the mastering, key/license MGT, the sender and the receiver [ISM06, PJK06], see Figure 7.

The mastering is responsible for equipping the content for distribution; it has the option to encrypt the media content and specify the usage rights, which helps it to work as a clerk, or it may provide the encryption key to the sender and the sender will do the encryption part. ISMA uses "Advanced Encryption Standard Counter Mode" (AES-CTR) for protecting the content media [Hou04]. Mastering may get the

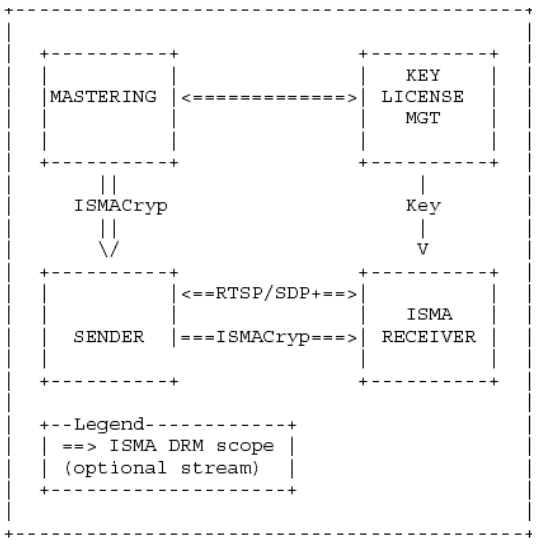

Figure 7: ISMA DRM architecture [ISM06]

encryption key from the key/license MGT or provide the key whenever it is needed. In the scenario where the encrypting of the media is done by the mastering entity, the sender is not aware of the encryption key; it has no job but to send the protected media whenever it is needed to the customers. Finally, mastering is responsible for advertising for the content media.

The sender is responsible either to encrypt the content media or receive the protected media from the mastering entity, and then stream it to the receiver; the distribution mechanism for the streamed protected content is the real-time transport protocol (RTP) [Doe07, PJK06]. The sender has the option to predict the encryption key by following the same procedures that are given to a user by the key/license MGT, and then generate the protected media. The media could be saved in a file before it is being streamed or streamed directly from the sender.

The key/license MGT is responsible for generating suitable licenses according to the granted rights; the license authorizes ISMA users to use the protected content media. It contains the two main components: the decryption key and the usage rights. For interoperability issues, ISMA tries to include all types of licensing schemes. If the responsibility for creating the key encryption is on the key/license MGT, then it

may generate that key depending on some properties of the receiving entity, which are used for authenticating the receiver. The ISMACryp uses the secure real-time transport protocol to authenticate protected content and uses existing key management standards, which provides the flexibility for the content provider to chose which key management he is going to use [ISM06].

The ISMA framework supports three types of receivers: ISMA-only-receiver, MPEG-receiver and IPMP-X receiver [ISM06]. The first type represents the receivers who play streamed MPEG media, the second represents the receiver that can play streamed data or stored files of type MPEG-4, and the last one represents the receiver that can parse and process Intellectual Property Management and Protection Extension (IPMP-X) format. The rendering software on the receiver side is responsible for contacting the sender and key/license MGT, authenticating both of them, acquiring and authenticating the license. ISMACryp uses the secure real-time protocol (SRTP) for integrity checking, accessing the proper decryption key for the protected content media and enforcing the usage rights. The receiver's rendering software has the option to decrypt, authenticate and check the validity of the control flow between the sender and the receiver. The receiver's rendering software enforces the usage rights [ISM06]. In the ISMACryp specification, the protected media are decrypted only just before they are decoded by the rendering software [Doe07].

ISMACryp supports super-distribution by providing the ability to OMA-DRM2 compliant devices to store streamed media into DFC/FDFC file format. Then, the compliant device can super-distribute the content to other friends' devices. That will increase the availability of the service as well as decrease the load on the sender server. Again, those devices should acquire a license in order to play forwarded content media [ISM06].

It is known that using a public key infrastructure for hiding data is robust, but it is inefficient, especially to protect real-time streaming media. For efficiency purpose,

the ISMACryp framework uses a symmetric algorithm for data encryption, authentication, and integrity purposes. ISMACryp has the option to change the mechanisms used to for encryption, authenticating and checking the validity of the message sent via this system [ISM06].

2.4 Persistent Protection

Encryption technology, alone, is not enough to protect content media [AH04, Mar]. The main purpose of the DRM system is to guarantee persistent protection for delivered media [LSNS03]; in another words, DRM systems are designed to guard intellectual property against digitally related criminal actions. DRM systems allow intellectual property owners to embed control within the delivered products.

DRM systems can be defined as cooperative and organized efforts among trusted entities and tools in order to achieve persistent control over digital products [PBW, Tec, Int]. The phrase "persistent control" is used to imply control over a sufficient period of time but not forever. After a certain period of time, the content may become available for free, not because the owners of the copyright have given it up, but rather, because they have stopped enforcing it. This is because the cost of satisfying demands for new copies is more than the content producers' revenues, and they have already achieved sufficient returns on their investments [PBW02, AH04, Hua07].

The word persistent means "refusing to give up" or "persevering obstinately" [Ans], which means that the persistency is valid at any time and any place. Persistent protection, in this context, means to obstinately protect the digital content from digital piracy at a place that is outside the content provider's ownership. However, persistent protection at any time and any place seems to be impossible. After some time, this persistency will decay due to the fact that the secret used to hide such content will not remain a secret forever. We will relax the definition of persistent

protection by providing the environmental conditions and the time period in which persistency is to be valid.

Liu et al. [LSNS03] suggest that persistent protection is gained by encrypting content media using a cryptography system; then, a distinctive identification needs to be assigned to the rendering device. This identification is used for authentication purposes. A license that embeds the decryption key(s), as well as the usage rights, needs to be issued to the rendering device. Finally, software-based or hardware-based protection must be used to enforce usage rights that are embedded in the acquired license.

Chin-Ling Chen [Che08] has expressed that persistent protection of delievered media content is achieved by individualizing the end user's machine and employing a digital certificate.

Most of the software-based and hardware-based protection used by many companies today does not support standard or common implementation. They use a proprietary implementation and design called a closed system [Che08].

2.4.1 Hardware Based Protection

Hardware-based protection is intended to protect software programs from piracy and tampering and to protect user's private digital information from unauthorized use and distribution, thus protect user's privacy. One example of a successful hardware-based protection system used to exchange digital products is smart cards. The smart card system is an integrated circuit used as a portable token that embeds a secure crypto-processor, random access memory (RAM), and a secure file system to protect cryptographic data such as a secret key. The design of the smart card is considered proprietary, and the secure file system contains private information about the end user for identification and authentication purposes [SSK04, Cle, Smab, Smaa].

Smart phones with OMA DRM2 specifications represent another example of the successful use of hardware-based protection. DRM Agents, as described in the OMA DRM Architecture specification [All06], embed unique private/public key pairs and certificates, which are used to identify and authenticate mobile devices and to individualize the acquired right objects for that device.

Trusted computing platform alliance (TCPA), or trusted computing group (TCG), provides a specification for trusted computing environments and protocols that is composed of trusted hardware, BIOS, trusted OS kernel, self encrypting storage, and trusted anti-virus software. TCG specification provides three access privileges:

- Privileged access[TCPA members only].
- Underprivileged access [platform owner].
- Unprivileged access [non-TCPA applications].

In the TCG, the following components are essential for enforcing DRM usage rights and security policies: Cryptographic operations, such as public and secret key encryption; key store; key management; and secure booting process [Gro07].

The main problem with hardware-based protection is that it is not easy to replace once it has been hacked.

2.4.2 Software-Based Protection

Software-based protection needs to be individualized in order to prevent it from working on more than one device. For example, each instance of Apple's Fairplay player embeds the hardware information of the device that is supposed to launch it; this is called individualization via binding hardware information. Microsoft media rights manager is another example of individualization via binding hardware information,

wherein the player with Windows Media Rights Manager uses DLL files, which are individualized for the distinct player that is supposed to run on a specific computer. The individualization process is achieved by generating a unique DLL file that is embedded with the computer hardware's unique identifier and private key. When the clearing house issues a license to a particular computer, it is encrypted with the related public key. Thus, the only machine that can use the license is the one with the right private key [LSNS03].

The license provider or clearing house, in turn, individualizes any acquired license by encrypting the media key with a specific DRM player's public key and then embeds the encrypted media key within the license. This process is called individualization via binding certificate [Che08]. The advantage of binding a license to a unique player is that it prevents the license from being transferable.

The individualisation process gives the content provider the power to make the digital content work under specific individualized DRM components [LSNS03].

The most important thing here is that the DRM enabled application must be tamper resistant in order to enhance the reliability of the playing machine. Rights enforcement is one part of the DRM enabled software that runs in the user's machine that needs to be protected from any modification, replacement or discarding. By employing a tamper resistant object, we make it more difficult for the software to be modified, replaced or discarded. Some companies use proprietary encryption and decryption algorithms, which are not tested and may not be trustworthy or efficient [LSNS03, LSNS03]; this is called security via obscurity. Other companies employ code obfuscation techniques in order to hide the logic of the obfuscated software.

The antiscreen capture program is another example of the successful use of software-based protection, which has the ability to prevent the end user from capturing sensitive information at the application level; this protection works at the operating system level [LSNS03].

The main problem with software-based protection is that it is susceptible to reverse engineering and dynamic and static analysis attacks by software experts.

2.4.3 Code Obfuscation

The code obfuscation process is used to make software codes hard to guess by human or reverse engineering processes, as well as harder to modify. It is important to note that code obfuscation does not alter original code functionality or logic; in other words, the code is obfuscated, but gives the same results as the original code. However, usually the performance of obfuscated code is degraded [EDB04, CT02, Low98].

The reverse engineering process dumps the executable binary code from a machines's memory, then interprets the code into assembly code, which called disassembly process, and then extracts a higher-level structure from the assembly code, which called decompilation process [LD03]. Figure 8 depicts the reverse engineering process.

The main goal of the obfuscation process is to add more complication and cost, as much as we can, to the static disassembling and decompilation processes. However, forever secure obfuscation algorithms are impossible.

Code obfuscation is categorized into three types [Low98]:

- Layout Obfuscation: this process is achieved by changing the variables' and functions' name into meaningless names.
- Data Obfuscation: this process may achieved by changing the scope of data structure variables, or changing the variable location, or data encoding or even data ordering.
- Control Obfuscation: this process is achieved by changing or hiding the control

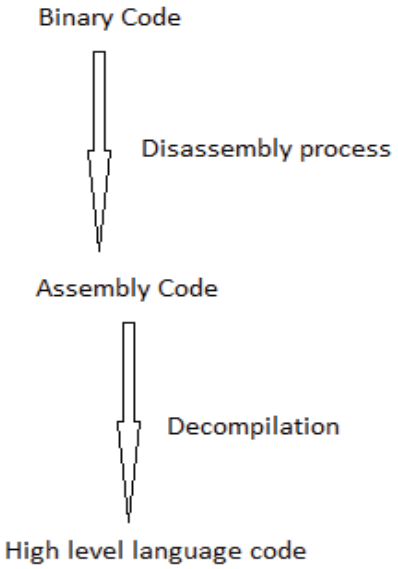

Figure 8: Reverse engineering process

flow and/or control computations of the original control flow graph.

Code obfuscation can be done using techniques used to increase instruction level parallelism such as software pipeline, and code unrolling technologies [HP07].

2.4.4 Mobile Code

Mobile code is a piece of software that has the ability to run on the target machine regardless of the platform. In the distributed system, this approach is called code migration, and it is useful for increasing performance, reducing the network bandwidth and improving load balancing. The following three examples show the benefit of process migration [TS02, RLMR00]:

- If a process wants to use resources that exist in another machine or wants to do heavy computation on a machine that does not have enough resources, then

migrating the process to the other machine with enough resources will lead to increase the performance.

- Sometimes, a process needs to use large amount of data that exists on another side of the network, and that process only wants to compute the summary out of this data. In this case, instead of moving the data from one place to another place which will overload the network, it is better to migrate the process itself to the other side of the network and then do the computation and return the summary. We assume that the size of the mobile process is small.

- If a machine is heavily loaded with many processes, and that machine is part of the distributed system, then migrating some processes from heavy loaded machine to another machine that is not loaded will help to improve the load balancing and the performance as well.

Mobile code technology allows a process to run on different hosts: Java applet is a good example and mobile agent is another example. The least necessary condition to run a Java applet program is Java enabled platform or Java enabled browser [GA98]. Java platform easily allows the Java class or applet migration from one host to another host; for this situation, a Java applet/class may be prepared in a way to circumvent target machine stability. For security purposes, Java applet/class needs to be verified, authenticated, authorized and controlled [RLMR00, DD00].

Any illegal changes of any system state sometimes considered an attack. In order to make sure there is no attack, we could employ integrity checker for the important file systems. The consequences of any system that is under attack may include system files modification or deletion or sending user's private information via network [GA98].

The main reason behind the ability of executing Java files on any devices is that the Java architecture supports Java code mobility and platform independence. The result of compiling Java source code is an intermediate language called Java bytecode,

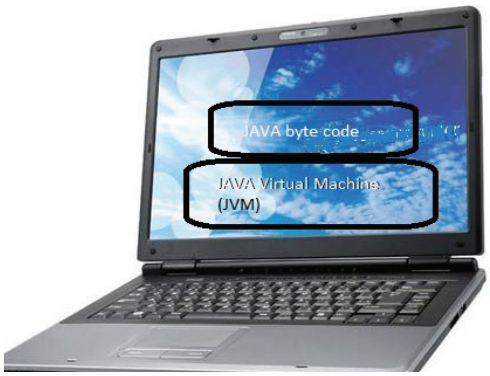

Figure 9: Java virtual machine

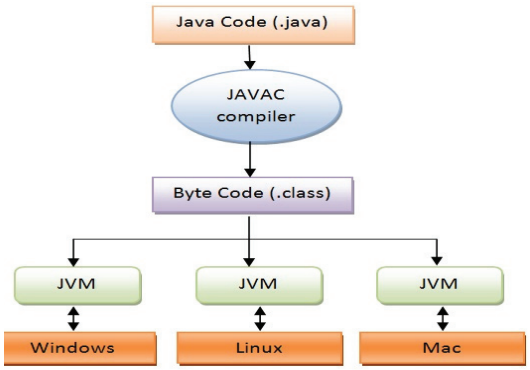

Figure 10: Java virtual machine [Pat08]

and any Java bytecode can be executed on any machine that has Java virtual machine (JVM). See Figure 9, Figure 10 and Figure 11.

As stated by Gritzalis and Aggeli [GA98], Java bytecode is an architecture independent object, which means that Java byte code can run on any platform that supports JVM. The main components of the Java virtual machine are [Pat08]:

- The Stack, which contains local variables, execution environment and operand stack.
- Garbage-collected Heap, which holds Java objects.
- Method Area, which holds Java bytecode.
- Program Counter.

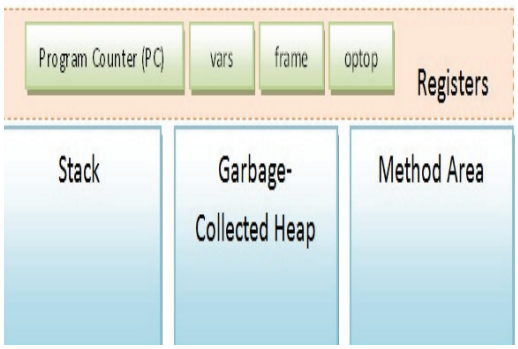

Figure 11: Java virtual machine components [Pat08]

- Registers.

One of the important issues that needs to be considered when transferring Java bytecode is security. Java bytecode needs to be authenticated and authorized before it runs on a specific machine. For that issue a trust relationship via signing model is built between Java code producer and consumer. The generator of a Java code digitally signs it, which enables the client to authenticate and check the Java code integrity [RLMR00, HB01].

A restricted environment called a sandbox is used. Sandbox is a construct used to control Java applets. The sandbox has a range of customizable access control starting from a simple sandbox that accesses the standard input/output and its memory space, to a sandbox that can access all resources in the target host [RLMR00, HB01]. The sandbox contains a bytecode verifier, an applet class loader and a security manager. The bytecode verifier checks the bytecode to see whether it is vulnerable to security attack or not. Applet class loader loads applet code and all related objects into the host memory. The security manager checks all applet operations that need to be perfomed by the host CPU, and if there is a security attack, then it stops the action [San].

Java 1.2 architecture is an extensible and comprehensive security framework and has many features and mechanisms, such as security policy, access permissions, protection domain and fine grain access control. A new service has been added to Java 1.2 called Java authentication and authorization service (JAAS), which works as authentication framework and it is responsible for authenticating users and authorizing their permission. In Java 1.2, every user has an identity based on some sort of evidence. Java 1.2 allows to define set of protected resources and terms or conditions, which are used to verify identified users to use them. JAAS policy used to state permissions to users. java.security.SecurityManager along with java.security.AccessController dynamically enforces access control [LG99].

The Java platform introduces APIs for main security services such as symmetric and asymmetric cryptography infrastructure, authentication and authorization services, secure communication services; ex, secure session layer (SSL), and fine grain access control. The Java platform supports interoperability and it is easily extendable to support new services [(SD05].

Now we come to the more challenging security problem, which is protecting mobile code from any malicious hosting target. In some cases, mobile codes may hold some private information such as a private key. This private information needs to be protected from being exposed. Mobile code may be subject to forwarding to another target. In this case it is subject to owner impersonation. Mobile code is subject to modification [RLMR00, HB01].

In summary, Java platform supports protection for hosting machine against mobile malicious code.

2.5 DRM Attack Model

Information security developers are concerned about countering threats. In this section, we will discuss threats that content distribution services are facing.

Since DRM developers have found that the easiest way to develop a DRM system is to consider that the end users' devices are trusted, therefore the most important role in the DRM system is the role of DRM agent (DA), which enforces the compliance to the content owner(s)/publisher(s) definition of legal activities on the content media. The DA controls the use of protected content media by burying secret keys used to decrypt that content, i.e., the DA works to provide a protection against piracy. The software and hardware attackers try to break the DA by exposing these keys and gain access to the clear content media.

Deploying a DRM software solution within generic machines is viable and cost effective, but it is susceptible to various attacks and it degrades the system performance. Normally, because DRM processes reside in customer-machines' memory, attackers can physically access to these memories, thereby, they can reverse engineer, disassemble and decompile binary codes inside these memories and then extract sensitive information used for content media hiding. Or, they can dynamically monitor process execution and follow the pointers to the location of secret keys [YKM+06]. In addition, a software solution may carry a serious attack against a customer, specially when it works as a virus spying for private information [FH06].

We consider DRM hardware solution as a black box, which hides sensitive secrets and prevents them from being released. This depends on the fact that the attacker has little knowledge about that box's internal structure, and the existing tools' capabilities are too limited to catch much useful information. Therefore, tampering with such hardware is too limited. This makes it a promising solution for future trusted computing [SV01]. Unfortunately, this solution is not economically feasible

Table 1: DRM Software and Hardware

	DRM SW solution	DRM HW solution
Attack	easy: tools are available	hard: need special tools
Fix	reinstall new version	replace or repatch HW component
other problems	Needs more computational cost	power surges & costly installation

Table 2: Attack Cost

Tools cost	Knowledge availability	Time availability	Attack cost
Low	Reverse engineering	Limited	Less expensive
Low	Reverse engineering	Open	Low
Low	Available	Limited	Medium
Low	Available	Open	Very Low
Expensive	Available	Limited	Very expensive
Expensive	Available	Open	Expensive
Expensive	Reverse engineering	Limited	Very expensive
Expensive	Reverse engineering	Open	Expensive

for existing PCs. Table 1 shows a comparison between DRM software and hardware [EDB04].

If the attacker has the right tools, a good experience and knowledge, as well as enough time, then s/he has a better chance to successfully attack the content distribution model. In contrast, if the tools used to circumvent the content distribution model are expensive enough, the knowledge about the technology used to protect digital assets is limited as well as the time to accomplish the attack is restricted by enforceable limitations, then s/he will have difficulty to attack the model. Table 2 shows attack costs for a variety of scenarios.

In software-based environments, the end user "owns" the environment where the rendering of the content takes place and the memory that holds the digital data as

well as the mechanism used to enforce the content owner's rights, which allows the end user to trace the place of the encryption key or modify the enforcement/ protection mechanism [BA, MVJDD05, GMM06b, GMM06c].

To conclude, the DRM systems seem to be reasonably strong, however they face the following threats [FH06, HB05b, EDB04]:

- All threats that are faced by transmitting data through insecure channels are applied to the DRM agent software, e.g., data eavesdropping, man-in-the-middle and modification attack.
- Reverse engineering of the DRM agent in order to deduce the location where the sensitive data reside.
- Monitoring the system behavior at run-time in order to observe the data changes in memory locations, so the attacker can predict sensitive secrets hidden inside the memory.
- Modifying the DRM software that is used to enforce usage policies, and then bypassing the enforcement point.
- Modifying the license in such a way as to allow customers to use fake rights.
- General operating systems suffer from many security holes. In MS-Windows OS, attackers have physical access to the machine's memory and disk, and are able to hide their spy-ware and dangerous files [HB05b], which helps the attacker to spy without being caught.
- DRM application may enable private information spying [FH06].

2.6 Summary

DRM systems control and manage the distribution of the content media. Mainly, it uses two levels of control, the first on the network level, and the second on the application level. In this chapter, we discussed the techniques used to provide the following services in the network level: confidentiality, authentication, integrity, non-repudiation and digital signatures. DRM systems use two approaches to provide protection on the application level, they are: hardware based protection and software based protection. We showed DRM architecture and some means to control the use of the content media after delivery. DRM system implementations are based on long-term same protection techniques and unicast distribution, which makes it susceptible to modification or replacement attacks, specially when software-based protection is used. A typical DRM system is not scalable when the number of customers exceeds the capacity of the NSP and/or the CP. We are searching for ways allow us to increase the difficulty of attacking protection programs and the scalability of the used DRM model. What we miss at this stage is to identify a comprehensive set of requirements that cover basic DRM functionalities. In this thesis, we will investigate general DRM requirements that mainly satisfy persistent protection and improve DRM scalability.

Chapter 3

Multicast Content Distribution

In the generic DRM architecture, the NSP takes no active role. It only provides a means of communication among the CP, LP and EU. As long as the network capacity is sufficient and exceeds the expectation of the End Users, and the capacity of the Content Server is sufficient, the existing DRM model is feasible. However, it is based on a one-to-one relationship. As the demands of the End Users grow, the linearly-increasing load will reach a point where the expectations of the End Users will grow beyond the capacity of the network or the Content Server to meet it. This is the scalability problem. If we could schedule the delivery of the highly-demanded services, then a promising solution is to shift from unicast to a multicast delivery mechanism.

In this chapter, we will outline the properties, advantages and disadvantages of multicast data delivery, and then summarize two existing approaches to providing security within multicast systems.

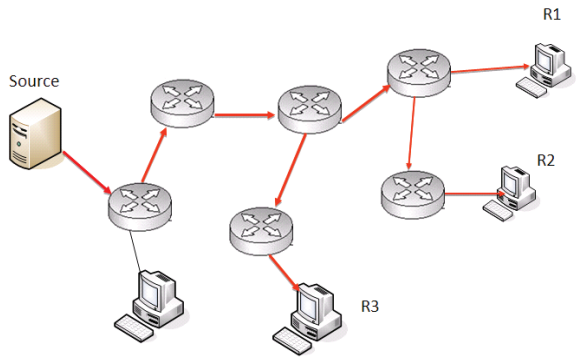

Figure 12: Multicast architecture

3.1 Multicast Data Distribution

It is known that multicast as a distribution mechanism has the advantages of lowering the price of distribution, speeding up the delivery process, lowering the resource usage of both the source and network and improving the network bandwidth [J. 07]. This introduces the possibility of enhancing the scalability of a system by shifting the traditional DRM technologies from a one-to-one relationship, between the sender and the receiver, into a one-to-many relationship. Figure 12 shows the distribution flow of the multicast architecture; the arrows, in the figure, reflect that there is one copy of data flow sourced from one sender to many receivers. Scheduled service works along with multicast, and by decreasing the delivery cost, will bring a motivating factor for the content media distributor to gain more from this distribution model.

The generic multicast model is good for free distribution when there is no need for control over the distributed content media. Multicast, as a technology, is widely used for applications that require fast response time and high quality. It requires fewer resources from the sender and the network and saves network bandwidth, in addition to reducing the provider's and network's CPU processing cycles, which leads to less delivery delay and cost [J. 07, IA06]. Most importantly, it scales up group communication, especially when the number of potential group members is large enough, and if they are connected through few Network Service Providers (NSP(s)).

Thus, it reduces the content server's load to the minimum. For group communication, the sender should synchronize with all group members to make multicast worthwhile. Existing IP Multicast follows IGMP/MLD protocols as follows [Cai02, Kos98]:

1- A group manager creates a multicast group using a specific IP address (Class D).

2- A customer who wishes to join a group sends a join message (IGMP/MLD) to the nearest multicast-enabled router. IGMP/MLD is a protocol used to manage multicast group membership. The data sent to group members is called group-data.

3- A customer who is no longer interested in the group-data sends a leave request.

4- Intermediate routers, that happen to be in the path between the sender and receivers, collaboratively build a multicast distribution tree based on join and leave messages. Routers can use any multicast routing protocol.

5- Intermediate routers know how to forward group-data to all receivers.

6- If any edge router receives a new join message, it extends the multicast tree if needed.

7- Once the multicast routing tree is created, group-data, sent by any sender, will be received by all group members.

8- A router responsible for forwarding group-data to that receiver checks whether there are any other receivers connected to that branch; if not, it truncates itself from the multicast tree; after that, no group-data will be received by that edge.

Multicast is a technology for delivering the same data packets to millions of people who are synchronized to receive them. In doing so, it hides the identities of all users from the sender. Replacing the on-demand service of normal DRM systems with

scheduled service results in the identities of all users being hidden from the original sender. This saves a lot of packets on the network, as well as requiring the content provider to produce and deliver only one copy of the content media.

Multicast allows the achievement of orders of magnitude better performance on the scalability requirement than the unicast model. The drawback is that once the media are delivered, there will be no control on the receiver side. IP multicast does not provide any security measures for the delivered media. Also, it does not preserve the end user's privacy and does not include any accounting or access control mechanisms over digital assets.

The advantages of using IP multicast did not motivate content providers to use it as a distribution mechanism because it only offers free join and does not monitor the sending process or restrict the receiving process. In other words, the content providers cannot control their contents' distribution[IA06]. Let us study the distinctive properties (factors) that contribute to the scalability of the IP multicast scheme:

F1 Separation of concern.

When a problem becomes more complex, the most effective way to deal with it is to divide it into sub-problems, solve these sub-problems and then gather the sub-solutions. When a sub-problem is assigned to an individual role, this procedure is called "separation of concern" in the software engineering process. DRM developers use this concept in their design; they physically separate the content media from the authorization mechanism in a distributed manner [VMF99]. The IP multicast generic architecture consists of three roles: the sender, interconnecting networks (routers) and receivers. IP multicast separates the sending process from the distribution process, the distribution process from delivering process and the delivering process from the registration process; the sending process is done by the sender, the distribution tree is generated by cooperative intermediate routers, the registration process as well as

the delivering process is managed by multicast-enabled routers at the edge of the network. The sender of group data is concerned about creating the group and sending the data that belong to those group members (group data). The interconnecting networks, routers, are concerned with building the multicast distribution tree, copying, forwarding and delivering the group data at the network level. Routers in the multicast case keep more information than in the unicast case, e.g., sender address, group address and output port [VAD99]. Receivers show their interest to receive or stop-receiving the transmitted group data by sending (IGMP/MLD) join/leave requests. Thus those parties have to collaborate with each other to perform the multicast distribution. In general, for a complicated interactions, it is advised to divide it into a set of cooperating interactions and apply the separation of concern concept, which introduces the scalability, flexibility and simplicity to the solution.

F2 Resource reduction.

The number of packets traveling inside the network using the IP multicast model is smaller than in the unicast case. Suppose we want to send a data file to a group of users; having created a shared distribution tree for each sender in a multicast case will definitely improve network bandwidth, since sending N copies of packets to N customers in the unicast case is replaced by one copy of packets using the shared distribution tree. In the multicast model, the multicast enabled routers replicate packets belonging to a group of users and forward them through their appropriate ports. In the unicast case, if a router happens to be in the path between the sender and N receivers, it will forward the same packet N times. The multicast mechanism provides a good solution for saving network bandwidth and decreasing the data traffic, assuming we have a large enough set of receivers and the control traffic is small relative to the data traffic.

F3 Better response time.

Interaction between roles affects the response time and that is what the separation of concern concept asserts. The sender should not directly manage the customers; doing that will affect the scalability of multicast sessions. S/He may not directly be responsible for authenticating, authorizing or accounting for them. This will lessen the interactions between a sender and all receivers. Intermediate routers manage the join and leave process for each user, so the number of control messages between the receivers and a sender is minimized. This improves the response time for the receiving service and improves the network bandwidth as well.

Content providers and owners insist on securing the multicast distribution model to control the use of their intellectual properties. This kind of protection needs to be valid for a certain period of time. If we could not develop a protection mechanism for multicast, then no company will develop intellectual properties.

In the following sections we will explore some of the related works that demonstrate general requirements for adding data protection and access control functions to IP multicast in the network layer.

3.2 IETF Multicast Group Security Architecture

The Internet Engineering Task Force (IETF) Multicast Security working group (MSEC) proposed the Multicast Group Security Architecture (MGSA) [VC04] as a reference framework that provides clear-data concealment to the traditional IP multicast. The MGSA architecture contains four actors: policy server (PS), group control and key

server (GCKS), the sender, and the receiver.

The PS is in charge of creating, managing and maintaining security policies within each group. There is an interface between the PS and the GCKS, which is used to pass security policies. PS peers can talk to each other in a distributed manner in order to scale up the MGSA framework. Peer PS servers need to authenticate each other before securely distributing security policies among each other. The GCKS is responsible for managing multicast groups and maintaining data encryption keys (DEK) for each group; the DEK is used to secure individual group-data. Group-data are the data that need to be securely exchanged within a specific group. The GCKS is also responsible for authenticating and authorizing end users and collaborating with another peer GCKS server in order to scale up and distribute GCKS related responsibilities. The GCKS contacts the senders and the receivers every once in a while for the purpose of KEK updates according to a specific policy issued by PS. Peer GCKS servers must authenticate each other before securely distributing information among each other. The sender is the entity who uses a specific KEK received from GCKS; s/he encrypts the group-data and then sends it to N receivers. In GCKS models any user can be the sender. The receiver is the entity that needs to be authenticated and authorized by the nearest GCKS before receiving the KEK and related policies [VC04]. See Figure 13.

The MGSA model can be implemented to a local area network (LAN) or a wide area network (WAN). The MGSA model has a simple structure in LAN; it is also called the centralized reference framework. It consists of a PS, a GCKS, the sender of a group, and at least one of the receivers. This structure suffices for small groups. When the group is larger (perhaps spanning more than one administrative region), the PS and the GCKS may be replicated, as is shown in the right column in Figure 13, and an individual receiver will connect to its local GCKS. Further details are available in [VC04].

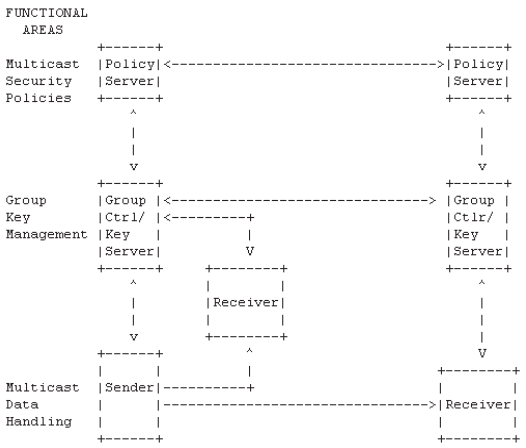

Figure 13: Multicast group security architecture [VC04]

This PS/GCKS model provides an example of the separation of concern concept in addition to the flexibility and simplicity of the IP multicast design model. The GCKS is the core entity in the MGSA architecture; along with the policy server, it is the manager of the group and is concerned with maintaining the confidentiality of the data being sent to all group members. Data confidentiality is achieved by protecting the data before making it available [VC04]. Many proposals have been presented for improving the scalability of the manager task [Mit97, WMSL00, MA07, ZZM$^+$06, CC03, CS05].

There are many advantages to the MGSA architecture:

- It adds a network protection layer to the delivered data flow packets.
- It works with any multicast routing protocol.
- It can be used as a reference framework for securing large multicast groups.
- It provides three functions: multicast security management, multicast group key management and multicast data flow security.

MGSA introduces the need for efficient key management and distribution as well as

the requirements for manipulating and carrying out access control of multicast content media for both the senders and the receivers; however, the specification of MGSA does not mention the requirements for accounting for the content media usage. Since MGSA's introduction, scalable key management protocols and schemes have been proposed by many authors. The IETF Multicast Deployment (MBONED) working group has introduced the requirements for accounting and controlling access to the IP multicast [THVV10]; they call this "well-managed" IP multicast. None of these previous studies provide the requirements for protecting content media from a hostile person who may receive the clear content media legitimately.

In summary, the most obvious advantage of the MGSA model is that the Internet Engineering Task Force (IETF) Multicast Security (MSEC) working group added two new players, the Group Controller / Key Server (GCKS) and the policy server (PS), to the conventional IP multicast in order to provide confidentiality of transmitted data in the network layer [VC04].

3.3 Atwood Model for Multicast Distribution

The increasing development of network technologies removes any need to consider the location or distance of target consumers. If we consider video on demand (VOD) as a small transaction between a content provider and a customer, the content provider or owner will take the responsibility of managing and maintaining the detailed corresponding records. Multicast content media distribution (ex. scheduled program), which consists of multiple instances of unicast connections (distribution tree), is a group of such transactions, and the supervision that exists between CP and EU in unicast case is no longer there in multicast case, because that will hurt the scalability of the multicast model.

As the delivery of high quality multimedia content media becomes faster, because

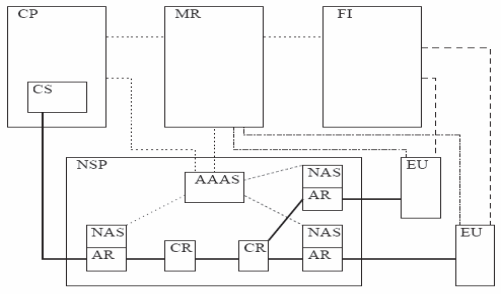

Figure 14: Multicast security architecture [J. 07]

of the increasing development of networking technologies, the location or distance of target consumers ceases to be of concern. This increases the potential size of the target audience for a specific content media stream. If "video on demand" (VOD) is required, then each request is likely to occur at a different time, and therefore must be managed separately. The content provider/owner will take the responsibility for managing a session, and maintaining the detailed corresponding records (session keys, account information, etc.). As the total number of participants in a session increases, the load on the content server increases in proportion. Eventually, as noted above, a point may be reached where the content server is unable to sustain the necessary flows, and it becomes useful to consider relaxing the "on-demand" requirement, in favor of the requirement for "efficient delivery", by using scheduled delivery and multicast data transmission. However, the supervisory relationship that exists between content provider and end user in the first case is no longer present in the second case, because to maintain it would hurt the scalability of the multicast model (F1, F2, F3). In this case, an efficient solution is to give this supervisory responsibility to intermediate proxies, and offer the content providers a limited summary. This was proposed by Islam and Atwood [IA06], see Figure 14.

Islam and Atwood [IA06, IA09] realized that the network service providers (NSP) should contribute to the solution; they state that without the help of the NSP, there will not be a feasible solution for scalable multicast content media distribution. They

propose adding distributed authentication, authorization and accounting (AAA) functionalities to the IETF model as a way to motivate both NSP and content provider(s) to deploy this service. Atwood [J. 07] proposed an architecture for secure multicast content media distribution. He calls his model "Multicast Security Architecture" (MSA). The solution touches billing and security parts for delivered media. Figure 14 shows the components of the MSA architecture. Sultana and Atwood, as well as Islam and Atwood, discuss the end user authentication and authorization services within MSA framework in order to control the access for group members [SA05, IA06, IA07b]. Islam and Atwood propose policy enforcement mechanisms at network level [IA09, IA07a].

MSA [J. 07] has seven main parties: the content provider (CP), the end user (EU), the network service provider (NSP), the merchant (MR), the financial institution (FI), the content server (CS), and the AAA server (AAAS) [J. 07].

The CS is responsible for producing the digital content media for a selected group session; the CP receives the encryption key from the group owner and encrypts the digital content media before the distribution starts. The CS is responsible for delivering the encrypted media. The MR is responsible for advertising and managing content media as well as distributing the electronic ticket for the legal EU according to a defined policy stated by CP. The FI is responsible for providing a proof that the selected EU is able to pay. The NSP provides suitable media to use multicast as a distribution mechanism for content media, account for the use of content media and enforce the usage policy as defined by the CP. It checks the EU's credentials and provides authorization mechanisms for a legal EU via the AAAS, as well as providing the accounting mechanism on behalf the MR and the CP. The EU is the entity that consumes the content media and provides funds for that consumption. It gets an electronic ticket from the MR in order to show it to the NSP who is responsible for attaching the EU to a multicast session. The NSP connects the CS and the EU via access routers (AR), and connects the ARs via intermediate routers called core

routers (CR).

3.4 Summary

Even though IP-Multicast distribution is not secure, it lowers the resources usage at the sender's site, and at the closest NSP resources near to the sender. The IETF multicast group security architecture provides scalable protection at the network level for the delivered content media, in other words, it provides scalable protection before delivery. It also provides simple policy management for each multicast group. The IETF multicast group security architecture does not provide any protection after the delivery of the content media. Atwood's multicast security architecture (MSA) has broader control and management for the delivered content media. The MSA model includes receiver and sender access control, and policy management. In this chapter, we briefly detailed the multicast security architecture, which is based on multicast distribution, and what protections it provides. The main problem with the multicast security model is that it does not prevent a dishonest customer from forwarding any delivered data. What we miss at this stage is to identify a comprehensive set of requirements to enforce DRM functionalities into multicast content distribution model. In this thesis, we will investigate general DRM requirements and show how to push them into multicast distribution model.

We are interested in reaching a solution that provides persistent protection and scalable content media distribution model in such a way that the attacker cannot defeat the system for long period of time. In this thesis, we will suggest some ways to enhance the generic DRM model by integrating some of the scalability properties of IP multicast, which will allow us to gain both DRM functionalities and scalable behavior.

Chapter 4

Problem Statement

In this chapter, we will show a solid understanding of problem statement which the dissertation addresses, which is: design a flexible mechanism, architecture and protocols, for scalable, scheduled and persistently protected content media delivery.

The DRM model is mainly designed to provide persistent control of the delivered content media. The technologies used with most existing DRM systems depend on long-period protection software or hardware. Hardware-based protection solutions are feasible where the protection providers have control over the hardware; therefore, OMA adopts a hardware solution, because cell phone providers have control over the hardware-based protection embedded in the cellphones. These models are not open platform models. There are a lot of pieces inside their community that are private. It is not impossible to reverse engineer these pieces, but like everything else, persistent protection is only valid for a certain time, and that time depends on how hard it is to reverse engineer the protection mechanism. The hardware-based protection solutions are valid for a longer time, but not forever. As a content provider, to increase persistent protection on delivered content media, it is required that s/he has some control over the hardware-based protection, and the cellular community has that control on the hardware-based protection embedded in cellular set. That is out

of scope for our problem at hand, because we want to look at a more general market, which includes personal computers and laptops. This implies that the protection needs to be quickly deployed and easily to be replaced. Software-based protection provides these attributes.

Various approaches exist for doing DRM based on the unicast distribution mechanism, and that creates a concern that unicast-based DRM system is not scalable when the number of users, who are using this service, increases. This increases the resources used at the sender's site, and at the closest NSP's resources near to the sender [PA11]. Scalable content delivery in this thesis context means that the resulting solution can afford any increasing number of end users. A promising and cost effective way to improve the scalability of the current DRM systems is to change their underlying distribution mechanism from unicast into multicast.

Many delivery applications need to support scheduled delivery because some applications, such as online world cup, interest million of users around the world, and the content media owner will be interested if the content media is persistently protected. Persistent protection in this thesis context covers content media protection after delivery.

At the beginning of our work, we did not have a comprehensive list of requirements for unicast DRM or multicast DRM. We start our work by:

Goal1: Exploring and collecting basic list of requirements for both unicast DRM and multicast DRM.

Later on as the work progressed and we become more mature in the area of DRM world, we found the collected list of DRM requirements is applicable to both unicast and multicast DRM. As the work progressed, this realization became clear that only the implementation and architecture are different.

Once the list is established, we will explore what parts of the requirements are not met or cannot be met with existing architecture when multicast is used. We found

that a small part of DRM requirements is met with IETF MGSA architecture, and a larger part of DRM requirements is met with MSA architecture. We will identify missing parts of DRM requirements that MSA architecture misses.

After determining the requirements that are lost, we will continue our work:

Goal2: Develop and validate methods that exhibit superior persistence and more scalable approach compared with previous approaches.

To achieve goal2, we will explore and validate the proposal of Grimen et al., which provides a light on persistent protection, using a formal tool to ensure that it works correctly for the unicast case. In other words, we will use Grimen et al.'s ideas to harden and increase the security of the protection applications that are used to control the content media access after it has been delivered. Then, we will propose and validate an extension to this approach for the multicast case and suggest an appropriate architecture for implementing the new approach. Thus, we improve the performance and scalability of current DRM systems by changing the underlying distribution mechanism from unicast into multicast.

Chapter 5

Requirements for Persistent Protected and Scalable Distribution Model

In this chapter, we are introducing the basic DRM requirements as a starting point for the solution to proposed problems with current unicast and multicast distribution systems. The main problem with the multicast security model is that it does not prevent a dishonest customer from forwarding any delivered data, and that problem is not perfectly solved in the current software-based DRM systems. Also, as we noticed, a typical unicast-based DRM system is not scalable when the number of customers exceeds the capacity of the NSP and/or the CP.

This chapter presents the fulfillment of our first goal, to explore and collect a basic list of requirements for both unicast DRM and multicast DRM.

5.1 Basic DRM Requirements

Nowadays, the digital world makes the copying of digital content media easy and perfect, which makes protecting these media more difficult. Digital rights management is a scheme designed to protect digital assets. There are four basic processes used in the DRM system: protection, distribution, management and control [Ian01, Ltd02]. Many researchers proposed a list of requirements in order to give the content providers the ability to acquire control for their digital intellectual property, and preserve the producers and consumers rights [NN07, JM07, AH05, LSNS03, AH07, LELD05, OLR+07]. We organize these requirements into five categories: access control, security, privacy, robustness and marketing:

R1: Prevent illegal access and allow legal access to valuable media.

- `R1.A:` Prevent the action of capturing clear content media in the distribution path.
- `R1.B:` Prevent the action of stealing clear content media when it is hosted at the end user's machine.

R2: Ensure the authenticity of interacting objects.

- `R2.A:` Digital assets.
- `R2.B:` Sender entity.
- `R2.C:` Receiver entity.

R3: Regulate the legal operation of content media; in other words, permit different authorization activities for different types of transactions.

- `R3.A:` Content owners need to specify their content media usage policies.
- `R3.B:` Content media usage specifications need to be protected and distributed to their appropriate destination.
- `R3.C:` Specified usage activities need to be enforced.

R4: Ensure the integrity of digital assets.

Table 3: DRM Requirements

Req	AC	Security	Privacy	Robustness	Marketing
R1	+				
R2		+			
R3	+				+
R4		+			
R5		+			+
R6			+		
R7				+	
R8				+	
R9					+
R10					+
R11					+

R5: Ensure the non-repudiation for the service.

R6: Ensure the privacy of end users.

R7: Ensure the availability of the service.

R8: Reduce the damage caused by the attacker.

- `R8.A:` Prevent "break-once, break everywhere" (BOBE).
- `R8.B:` Detect and fence the cause of illegal content media distribution.
- `R8.C:` Repair the protection engine once it has been compromised.

R9: Support service on demand.

R10: Ensure efficient use of content provider's resources.

R11: Allow domain access.

In Table 3, we map these requirements into the five categories: access control (AC), security, privacy, robustness and marketing. In the next section, we will give a justification of the elements in this table and discuss the goals behind these requirements in detail.

5.1.1 Goals behind Basic DRM Requirements

The content owners do their best and spend time and effort to produce remarkable intellectual properties. They need to protect their works in order to control the use of them for a certain period of time. DRM strives to achieve "persistent access control" for the content provider/owner and provides him/her the ability to regulate the content media operations [JM07, AH07]. This persistency is achieved by hiding the content media as a first stage, then filtering the access to it as a second stage. Requirement one is about hiding the content media and both requirements two and three are about filtering and organizing the content media access. Hiding the content media is accomplished by making the following sub-requirements valid:

R1.A: Prevent the action of capturing clear content media in the distribution path [Access control requirement].

R1.B: Prevent the action of stealing clear content media when it is hosted at end users' machine [Access control requirement].

To allow only legitimate customers to utilize the service, a demand for identifying and authenticating the customers is needed. Requirement two is to ensure the authenticity of the following:

R2.A: Digital assets.

R2.B: Sender entity.

R2.C: Receiver entity.

This involves clarifying, ensuring and assessing the truth of any declaration sourced by a valid entity (content sender, content consumer and the digital assets), we consider it as security requirement. To start with, consumers want to ensure the identity and authenticity of the sender(s) before receiving any data, and this is useful for protecting customers from sender spoofing. As well, a sender requires that each valid

customer be identified and authenticated before s/he is authorized to use the product, and this will help for billing issues. Finally, the content consumer hopes to authenticate the product itself before s/he starts using it. This process helps him/her to avoid any harm that could be sourced from unknown content providers or products.

Requirement three introduces the need for managing different types of content media usage, that is why we classify this requirement as both access control and marketing. It is further subdivided into the following sub-goals:

R3.A Content owners need to specify their content media usage policies (rights/licenses) [Marketing requirement] [JM07, Irw04].

R3.B Content media usage specifications need to be protected and distributed to their appropriate destination [Access control requirement] [JM07].

R3.C Specified usage activities need to be enforced [Access control requirement] [NN07, Kam02].

Requirement four ensures that any digital assets used in the context of content media distribution have not been changed in the path from content providers to content consumer, and it takes two flavors: checking the digital assets' integrity in the distribution path and giving a promise to the Stakeholders that the digital asset will not be changed at the point when the end user can access it; we considered it a security requirement, because if the integrity of the assets is violated, then it could introduce a hole in the content media distribution model.

Requirement five is to ensure the non-repudiation action for the requesting process, which is an important security service that avoids any tensions that could happen between content seller chain and the end users. There should be evidence of selling/buying the product for both sellers and buyers. The seller needs this service to

prevent the buyer from denying using the service and not paying the fees. At the same time, if user protection is broken because of a delibrate security hole embeded inside a product sold by the seller, then this offensive action should not be denied by the seller, this is a kind of security requirement. To conclude, both the sellers and the buyers should be responsible for their activities. In another view, if the access to the service is granted, then this action should not be repudiated. We consider this requirement as both security and marketing requirement.

The sixth requirement affirms anonymity to end users by preventing unauthorized entities from accessing users' private information such as name, address, date of birth, credit card number, and so on. This information could help the attacker to gain access to transactions that belong to someone else. It is a privacy requirement.

Requirement seven is important for all parties. It concerns keeping the product service, the users can receive the digital product and its license if they are eligible without any blocking, which means that denial of service (DOS) attacks must necessarly be eliminated. We considered it as a robustness requirement.

Requirement eight tries to reduce or mitigate the unsatisfactory effects of service attacks. It tries to find a means of defense against intentional irresponsible actions and return the system to the previous stable state. It is a robustness requirement. We further subdivide it into three lines of defense.

R8.A Prevent "break-once, break everywhere" (BOBE) [PBW02, JM07].

R8.B Detect and fence the cause of illegal content media distribution.

R8.C Revise the protection engine once it has been compromised [JM07, BE06, GMM06b].

Requirement nine endows the DRM system with flexibility; once the users choose the time frame for enabling legal operations on the content media, they should be able

to do that [JM07]. Somehow, it is related to the availability requirement, R7. This requirement is important for marketing issues, because if the service is motivating users, then the service provider is successfully marketing her/his product.

Requirement ten seeks the efficient usage of the content providers' resources. It contradicts requirement nine, which demands reserving fixed resources for each individual user. Because of the fact that senders have limited resources, they can serve only a limited number of users. Therefore the flexibility desired by requirement nine is influenced by the efficiency desired by requirement ten. Requirement 10 is again a marketing requirement.

Requirement eleven gives the ability to each customer to use the same content media on a limited number of devices s/he owns. It is preferable that deploying any technique to achieve this requirement not hurt any of the previous requirements. This requirement attracts a customer to use this service, therefore it is a marketing requirement.

5.2 Addressing Basic DRM Requirements within DRM Examples

Section 2.3.2 shows current DRM solutions and discusses four examples of DRM models and their limitations. In this section, we will address the basic DRM requirements of the DRM model examples.

Figure 15 addresses the availability of the basic DRM requirements in each mentioned DRM technology in Section 2.3.2.

Req	WMRM	RMS	OMA	ISMA
R1.A	Secure Channel	Secure Channel	No secure channel	AES-CTR [ISM06]
R1.B	Securing rendering path	securing rendering path	HW & SW tamper resistance	Client viewer
R2.A	Certificate, watermark & fingerprint	No Information	ROAP authenticates content	message authentication code (MAC) [ISM06]
R2.B	Not mentioned	Certificate	ROAP & LI signs RO	MAC
R2.C	No user authentication	MS active directory	ROAP,PKI infrastructure, SIM & USIM cards	MAC
R3	Licenses	Licenses	RO & DA	Licenses
R4	Secure hashing	SHA-1	ROAP checks FDC/PDFC integrity[Irw04]	SRTP message authentication[ISM06]
R5	Not supported	Not supoorted	Not supported	Not supported
R6	Not supported	Not supported	Not supported	Not supported
R7	Service replication	Service replication	Super-distribution	Super-distribution
R8.A	Individualization	Individualization	Encrypt KEK by DA's public key	Not supported
R8.B	Offline search, watermark	Offline search	Watermark	No Information
R9	VOD	VOD	Service on demand	No Information
R10	P2P distribution	P2P distribution	Super-distribution	Symmetric cryptography & Super-distribution
R11	Support multiple media format	Support multiple media format	Domain registration	No Information

Figure 15: DRM system comparison

5.3 Persistent Protection Requirements

One of goals that has been requested by content providers is to close the security hole that exists in the current content media distribution systems, and to motivate the network service provider to participate in the solution of content media distribution model. The road to having a scalable content media distribution is dependent on having the help of the network service provider [J. 07]. In our opinion, adding persistent protection properties to content media distribution systems is done by closing the security holes generated by dishonest customers.

Persistent protection is English terminology is composed of two words, the word protection and the word persistent. Technically, protection means, to protect an object whether it is tangible or intangible, and persistent in this context means, to make the protection valid for a certain length of time. In the context of persistent protection for delivered digital media, and since we are using the Internet to distribute the content media to a large number of users, the most feasible way to achieve this protection is to use cryptographic technologies. Encrypting the content media using efficient and secure encryption algorithm with secure encryption key converts readable, watchable and/or audible content media into unreadable, unwatchable and/or inaudible media, which is called protected content media. This process achieves the protection phase for the content media.

The next phase is to control the access to the protected content media by authenticating and authorizing the end user(s) who are eligible to read, watch and/or listen to the content media. By authorizing the end user we mean, that the end user is allowed to use the content media according to the rights and privileges the end user has been granted from the content owner or his representative. In order to give the end user the right to use the content media without removing the protection on the content media, we need to use a trusted tamper resistant object, which is responsible for allowing end user to legally use the content media and emphasize permission rights

on the content media.

To achieve persistent protection, the tamper resistant object needs to be protected for the period of time that is enough for the content provider to gain enough control for his/her work. In our opinion, the road to the protection persistency can be achieved by making at least the following three basic requirements valid [BA]:

- R1 Prevent illegal access and allow legal access to valuable media.
- R3 Regulate the legal operation of content media; in other words, permit different authorization activities for different types of transactions.
- R8 Reduce the damage caused by the attacker.

The first one is to hide the digital assets so no one can access them; employing the cryptographic tchnologies is the easiest way to accomplish this hiding in public networks. The second one is to arrange and manage the access to the digital assets by providing legitimate end users with the capabilities to access the protected digital assets; legitimate clients can use the content media they are provided with the decryption key, but that should be under careful manipulation in such a way that they do not really understand the right decryption flow. The third one is to provide the means for a reaction when detecting an attacker has been successful. This requirement is more detailed via the following sub-requirements [BA]:

- R8.A Prevent break-once, break everywhere (BOBE).
- R8.B Detect the cause of illegal content media distribution.
- R8.C React to the action of illegal distribution.

Nothing will remain secret forever, so protecting these protection mechanisms is not something easy, at some time these protection mechanisms will be known and the hidden content is not really hidden, "DRM systems can't protect themselves" [Doc05]. If time is not considered, a knowledgable hacker may have sufficient funds and appropriate tools to extract the sensitive information (key) from the machine's memory to achieve a successful attack. The ability to achieve persistent protection

at the user level depends on the ability to protect the DRM subsystem at the user side.

In order to reduce the damage caused by the attacker, replacing the protection mechanism used to control the access to the content media with a new one is one promising kind of reaction. In this case the attacker spends time and effort in order to attack and reveal the protection mechanism, which will be not valid at the time of the eventual attack.

Arbitrary long persistence is not achievable. If the achievable persistence with a particular mechanism lasts for an insufficient time, then renewability can be used in order to extend the persistency of the protection. "Renewability" is a term used to indicate that one of the DRM component(s) can be refreshed or replaced. The basic mechanism is good enough to last for a specific time, and a sequence of basic mechanisms provides persistent protection with a sufficiently long effectiveness. One entity will interact with the user to renew the protection mechanism periodically or when the current protection mechanism is compromised or has expired [GMM06a, GMM06b]. This mechanism has the disadvantage that it reduces the network performance due to the high network bandwidth that is used for this renewability mechanism with the current DRM system. The renewability mechanism appears not to be used, which gives us an indication that there is no real persistent protection in current DRM systems.

Another way to gain long persistence is to use tamper resistant hardware that is strong enough for the spectrum of expected attacks over the required time, and then replace these hardware devices, after the period where we think they are not useful any more or they have expired [SCA]. A hardware solution is suitable for a high-availability system with few participants. A software solution is suitable and cost effective when we have very many customers. In order to gain a range of persistence, either we need a single technique that is strong enough for a desired time period,

or we need a technique or techniques that are useful only for a shorter time period, coupled with renewability.

We may need renewability as one element of a DRM toolkit that offers the choice to use hardware that cannot be broken for long period of time, or to use a combination of hardware and software that makes the attacker no longer interested in attacking this distribution model.

5.4 Multicast Consequences

Multicast, as a distrbution mechanism, is an environment used to deliver a copy of digital data to selected receivers. The most obvious issues when using multicast as a distribution mechanism are:

- Scale up multicast service to the maximum.
- Remove the exclusive connections between the sender(s) and the receivers.
- Distribute the management of delivering the data stream.
- Reduce the resources that need to be used in multicast service to the minimum.
- Increase the availability of the resources that are used in the multicast service.

We saw in the previous chapter that IP multicast delivers one copy of the data stream packet to multiple recipients by copying and forwarding the received packet during the routing process. Multicast participants connect to the multicast group via the nearest multicast-enabled router using IGMP/MLD protocol. The network routers collaborate with each other to build the distribution tree. There is no direct connection between the sender(s) and the receivers in the IP multicast; the sender(s) is not aware of who the receivers are. The data stream's delivery is managed by the

routers that connect the sender(s) to the receivers. When the sender(s) sends one copy of the data stream, it reduces the resources consumed by the sender(s) to the minimum. In addition, when the routers copy and forward the data stream, it reduces the number of packets sent to the minimum. All previous achievements contribute to increasing the number of packets that can be sent by the sender(s), and thus, increase the availability of the resources used by the multicast service.

5.5 Multicast Content Media Distribution with DRM Enabled Requirements

Current DRM requirements are feasible for DRM systems up to the point where the server capacity cannot satisfy users' demands. By the time we reach the situation where the size of their demands goes beyond the server's capacity, we propably have enough customers to drop the "service on demand" requirement and replace it with an offering of scheduled services. This enables the content owner to upgrade the DRM performance, and therefore, serve more customers. However, it impacts the solutions to the DRM requirements, because multicast as a distribution mechanism truncates some DRM requirements and improves some others as well as produces new challenges in order to mitigate the others. We summarize the following requirements that need to be adjusted in order to enable DRM in the multicast model (DRMM), then we will discuss these requirements in detail.

R1 Prevent illegal access and allow legal access to valuable media

This requirement is to attain "remote control" along with the "Persistent Access Control" [AH07] on multicast content media distribution, we need the following sub-requirements to be valid:

R1.A Prevent the action of capturing the content media in the network level.

This requirement is being achieved by hiding the content media using cryptography techniques and key management protocols, and then allowing the legal customers to use the content media by giving them the means to unhide the hidden content media.

R1.B Prevent the action of illegal copying of the clear content media from customers' machine.

We need to build and deploy a tamper resistent object that controls the use of content media and prevents end users from attaining sensitive data. To achieve this requirement, we further divide it into the following sub-requirements:

R1.B.I The need for building DRM mediator by which only it has the capability to render protected content media for a distinct user.

R1.B.II DRM mediator needs to be trusted.

Trust in this context means that the mediator is not subject to change by any illegal entities.

R2 Ensure the authenticity of digital assets, senders and receivers [AH07].

Authenticating the sender as well as the customers or their devices is a prerequisite requirement for authorizing them. Autentication of the object is required for subsequent use with non-repudiation requirement.

R3 Regulate the legal operation of content media.

To achieve this requirement, we will use the idea of distributing a license to authorize the use of a content media. We further divide this requirement into the following sub-requirements:

R3.A Only legitimate users can gain access to the valid licenses.

R3.B It is recommended that license issuers can account for customers' usages without hurting the scalability of the multicast model.

R4 Ensure the integrity of digital assets [AH07].

Since the sender of the digital assets is not connected to the end user, then, this requirement becomes more urgent than in the unicast case. The reason behind that in the multicast case is the receiver needs to assure that the content media comes from a certain source, not from an attacker, and that helps him to assure for to a certain limit that the recived product will not affect the end user integrity.

R5 Ensure the non-repudiation for the service [AH07, OLR+07].

This service is urgent in the multicast case because the sender cannot track the service usage by the end user, and thus cannot directly have a proof for the content media consumption. This requirement is divided into the following sub-requirements:

R5.A Each end user needs to be uniquely identified.

This unique identification can be used to bind the tamper resistant object to the end user's machine, and thus the next requirement is needed,

R5.B Tamper resistant object should not be predictable or duplicated.

This requirements is needed to prevent the end user from cloning tamper resistant object and forwarding it to his friends;

R6 Protect the privacy of end users.

R7 Ensure the availability of the service [AH07].

Obviously, this requirement is urgent in the multicast case because if the service is not available for a short time, it will discount many users. This requirement is

achieved by the satisfying the following sub-requirements:

R7.A Protect the end user environment from hardware and software attacker.

R7.b Protect the network resources from network attackers.

R7.A Protect the sender's resources from hardware and software attacker.

If requirement R7.A is not achieved for some users, then the service is stopped for those users, which we do not want. However, if requirements R7.B and R7.C are not achieved, then the service is stopped for many if not all users.

R8 Reduce the effect of service compromise.

As we said before, requirement eight comprises three levels of defence, which are summarized in the following sub-requirements:

R8.A Prevent "break-once, break everywhere".

R8.B Detect and limit the effect of that illegal content media distribution.

R8.B is further subdivided into:

R8.B.I End users' Content needs to be distinguished and individualized.

R8.B.II This distinction needs to be robust.

R8.C Revise the protection engine once it is being compromized [JM07, BE06, GMM06b].

R9 Support service on demand.

For requirement nine, multicast does not support this requirement any more.

R10 Ensure efficient use of content provider's resources.

For requirement ten, multicast as a technology improves this requirement.

R11 Allow domain access.

This requirement needs to be considered after satisfying the previous requirements (R1...R8).

The next step is a comprehensive study of these requirements.

5.6 DRM for Multicast Requirements Comprehensive Study

From the scalability and content owners point view, multicast distribution gives up the opportunity to improve the performance for some requirements that are met in DRM unicast case, e.g., efficiency requirement (R10). But, it kills some other requirements, e.g., VOD requirement (R9). In different viewpoint, it cannot easily satisfy the BOBE requirement (R8.A), which is solved by individualization in the unicast solution. In the previous section we mapped the requirements for multicast and those requirements that need to be achieved in order to reach the optimal case for multicast DRM enabled solution. We will discuss each requirement from the multicast point of view.

R1.B Prevent the action of illegal copying of the clear content media from customers' machines.

This requirement is to assure full remote control and "Persistent Access Control" [AH07] on the digital assets for a limited period of time and is considered the most vital requirement for securing current multicast technologies. Without it, the content owner will not be cheering, if he doubts that his extensive work to produce remarkable product is under control once it comes to the customers' hands. To attain remote control on multicast assets, they must be protected at the network and the application levels.

The most mature scheme to protect multicast content media is Secure Multicast [J. 07], but it protects the content media at the network level. A sender encrypts the clear content media before flowing it into the distribution tree and individually sending these keys to each end user. Because some multicast applications require a dynamic membership, keys may need to be refreshed for every membership change. R1.A is well established by many researchers working in multicast key management and distribution, and we will not discuss it further.

A legal customer should behave in the way that they should: not copy or redistribute granted content media; unfortunately, bad users do not behave in the way that they should. To keep clear content media away from the users' hands and give the rights to legal customers to use them are the means for controlling the use of content media. Enforcing these rights would be the responsibility of the network service providers (NSP) and clients' platform. Satisfying both R1.B.I and R1.B.II gives the system the chance to achieve remote control at the application level.

R1.B.I suggests that to remove users' ability to directly access clear content media, it is sufficient to build a DRM mediator to mandate an individual user and give him the capability to legally use the content media with considering only legal usages to the service. In this way, the mediator holds up the customer's ability for extracting keys or any sensitive information used to protect the content media, e.g., knowing which algorithm is used for content media protection, and thus the end user can obtain the clear content media with great difficulty only.

OMA DRM 2 specification achieves the content media protection by keeping the private key inside tamper-resistant hardware, and by hiding the technology secrets used to build the DRM agent. Because the major audience for our proposed multicast content media is generic PCs, applying tamper-resistant hardware for multicast content media distribution is not feasible because the cost is unaffordable. The major problem with those PCs is that they do not have any special hardware that provides

tamper resistance and their operating systems are generic and do not provide any real protection [HB05b].

The feasibility could be achieved when applying tamper-resistant software. R1.B.I proposes to build a DRM mediator to control legal activities. However, a problem arises when a legal customer redirects that mediator to illegal customers, or extracts the secrets embedded inside the DRM mediator, or tries to modify the logic of the DRM mediator, e.g., modify the part of the code that is responsible for enforcement activities. R1.B.II requires a means to establish a trusted mediator in order to resist the following attacks:

1. Forward the working version of DRM mediator.
2. Reverse engineer the mediator.
3. Modify the logic of the mediator.

Requirement R3 helps the content owner to control and manage the access to the content media; it is to manage the relationship among all parties in the system. We can call it a marketing requirement. In this requirement's view point, the content seller needs to be able to specify the terms and conditions for using the content media. The policy server can specify what accounting information needs to be recorded for an accepted End User. This makes it possible, for example, to collect accounting information.

R3 is subdivided into two sub-requirements:

R3.A suggests to use licenses to authorize a legitimate customer to consume the content media. The license should describe the legal rights, constraints and include the encryption key. The license should resist being modified or forwarded to unauthorized users [Coy03]. Knowing that licenses are susceptible to an analyzing attack, which may allow the attacker to discover and extract encryption keys hidden in the

license, we need a mechanism that prevents the customer from tampering with these licenses.

R3.B takes care of another issue, if we adopt the LP role to manage the distribution of the license, we need to deploy it in a way such that the license sender does not need to be aware of the existence of the end user (F1) see Chapter 3, which contradicts the accounting issue. We need a cost effective mechanism to use licenses as an authorization and accounting mechanism.

R4 is to verify the integrity of the digital assets, security services and the customers' device. Digital assets comprises the content media, policy, licence and DRM mediator.

R5 ensures the non-repudiation of requesting a service. A content owner needs its customers to commit to non-repudiation of the request for a service. In the same sense, customers need content owners to commit to non-repudiation of the sending service or any damages could harm customers by using any DRM mediator. This requires that content owners or distributors should not alter the customer's security services.

R6 Protect the privacy of end users. We believe that this requirement has not fully been achieved in the existing DRM models. The content owner needs to trust customers before authorizing them (giving them the license) to use his/her products. In the DRM system, the trust model is based on direct security association between CP, LP and end user. Therefore, there is one advisor who mandates customers to follow her/his protocol. It is the responsibility of the end user to know who he is dealing with. Multicast deals with more complicated requirements due to the simultaneous multiple users' connection and dynamic membership support. Users may not be aware of the real senders' authority and then they should not be responsible for checking the senders' honesty. Entrusted role could break the whole system. Therefore, trust model has to be changed in a way that follows collaborative protocols

between roles and does not reduce the system scalability.

End users need to show their private information to LP as an evidence of their ability to pay for a specific service. Users need be sure that their private information will not be used in a wrong way; this problem is not solved in most DRM systems. Worse than that, users' privacy is under attack, but there is no legal recourse against the attackers [FH06, HB05b]. The largest challenge here is that using the DRM mediator makes saving user's privacy harder, because it may hide a rootkit [FH06, HB05a].

R7 Ensure the availability of the service. Here we are talking about preventing DOS attack and maintaining the consistency of CP, LP, digital assets, policies, and licenses. In the DRM system, the services become more available by introducing more servers or caches near to users, as well as by using peer-to-peer distribution, and this will increase the network load, especially when the number of users is inflating. The nature of multicast distibution reduces the extensive interactions between the senders' servers and receivers' machines (F3) see Chapter 3, which improves the scalability and reduces the number of servers needed to provide such a facility. Multicast introduces two challenges on this issue: a) the need for maintaining and accounting for end users' behavior requires the interaction between servers and clients; b) the increasing number of users will increase the probability of DOS attack. This issue was discussed by Islam and Atwood [IA06].

R8 introduces three lines of defense to mitigate the fact of compromising the service. R8.A is the second line of defense. If the attacker unveils the secrets used to hide digital assets, s/he can affect all the roles (content owners, content distributors and end users), s/he can play the sender role or harm end users and send viruses as well as redistribute content media and throw away content owners' money. This tragedy was limited in DRM solution because of the individualization technique. We need to deploy this technique for each individual DRM mediator and license on

condition that this individualization should not hurt the scalability requirement.

R8.B is the third line of defense. Content owners spent a long time generating content media and if anybody can download them from the Internet for free, then no artist can make any benefit. Monitoring the Darknet [PBW02] for illegal content media distribution and tracing the source of that distribution is the third line of defense and a way to prevent such bad actions. In DRM systems, this could be done by fingerprinting each individual copy. Multicast makes it harder to insert a different mark for each copy, because all customers should receive the same copy. This is one of the big challenging issues.

R8.C is the fourth line of defense, which requires that the system roll back to the previous secure state once it has been compromised. Multicast as a distribution mechanism may provide a promising solution to achieve this requirement.

The previous requirements are needed in the new system, some of them are easy, and others are not.

5.7 Summary

In this chapter, we have built the structure that illuminates the starting point to get to the fundamental solution for scalable and persistent protection for the delivered content media. We presented eleven basic requirements to achieve basic DRM functionalities. These requirements are organized into five categories: access control, security, privacy, robustness and marketing. Some of the eleven requirements are contradicting others, which leads to finding a trade-off when building a DRM system. We found that there are three basic requirements out of eleven that represent the basic rule to achieve persistent protection. These three have a total of eight sub-requirements, of which renewability and individualization are the most important

pieces. We have in mind to use multicast distribution mechanism to achieve scalability for delivered content media. In the next chapter, we will be looking for proposals that claim to achieve persistent protection, and use protection renewability as one piece.

Chapter 6

Grimen Model

Grimen et al. [GMM06b] proposed a software solution based on periodic renewal of the protection scheme that is used to control the access to content media. The software-based solution they proposed appeared to be a promising solution, but we have found two attacks against the protocol that they use to control the access to protected media. We will discuss their solution in Section 6.1, then we will go through the Grimen distribution protocol and introduce a formal version of the protocol in Section 6.2, after that we will show attacks on Grimen model and give a formal model of the attack in Section 6.4. In Section 6.5, we give a solution to the attack, and demonstrate its security.

This chapter presents an approach to DRM that exhibits superior persistence, which is a necessary prerequisite to meeting our second goal.

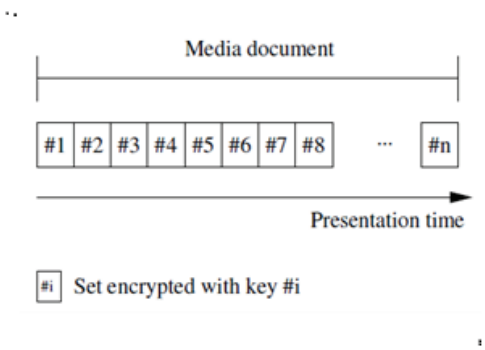

Figure 16: Dividing the media content media [GMM06b]

6.1 Grimen Distribution Architecture

Grimen et al. [GMM06b] proposed an architecture for software-based secure content media distribution that consists of four players: the content provider (CP), the stream server, which we are going to call the content server (CS), the viewer software (VS) and the security server (SS). The CP encodes the content media, and divides it into many pieces. It then produces as many symmetric keys as there are media pieces, in order to protect each piece with one unique symmetric key. This process results in what they called an encrypted encoded media document (EEMD) and makes it ready for distribution. See Figure 16. The authors called the time period for each piece a trust interval.

The CS is in charge of distributing the EEMD piece by piece. The VS player is running within the user environment. It is responsible for decrypting and decoding the EEMD and then rendering the content media and making it available for viewing. The SS is responsible for generating and delivering a piece of code called a Mobile Guard (MG), which is able to execute in the user environment. The MG needs to be plugged into the VS. The MG needs to be sent to the VS in order to configure the VS to maintain security states of the media document. The MG is responsible for checking the integrity of the VS components, including the MG itself, see Section 6.2, and then after successful integrity checking, the SS is going to deliver a corresponding

media key for a current delivered piece of the EEMD. The authors claim that the MG is hard to compromise; if successful software modifications happens, it will be detected and the system will have break once break everywhere resistance [GMM06b], see Section 6.2.

As the encoded media document is being split into multiple pieces, each piece is encrypted with a different key. The SS is going to provide the decryption key for each piece only at the start of its particular trust interval, upon request and after checking the integrity of VS; the VS needs to prove its good intention before delivery of the new decryption key. This good intention is achieved when the VS receives and executes a new generated MG, which means that for every decryption key there is a new MG. Each MG is distinct from the others, and each one is responsible for reconfiguring and securing the VS. The MG needs to be obfuscated to prevent the software hacker from predicting the internal logic of the MG within short period of time [GMM06b]. The MG is replaced for each trust interval, in order to prevent the software attacker from predicting the VS configuration when he has the time to gain the knowledge. Each instance of the MG is responsible for protecting the access control mechanism used by the VS for the duration of the trust interval. Each MG instance is going to check the integrity of the VS every new trust interval. The VS is going to request the corresponding decryption key at the end of each trust interval. The SS is responsible for receiving the checksum for the VS for each trust interval. Then, upon verifying the correctness of each checksum, it is going to send the next decryption key that corresponds to the new media piece. Figure 17 shows the Grimen et al. proposed architecture.

6.2 Grimen Distribution Protocol

The details of the process of receiving the decryption key for each trust interval (as proposed by Grimen et al. [GMM06b]) are as follows:

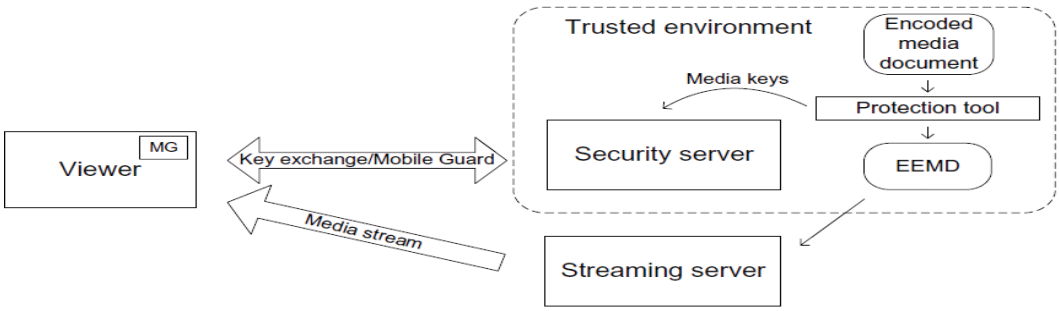

Figure 17: Grimen et al. proposed system architecture [GMM06b]

- The VS generates a random unpredictable transport key.
- The MG executes and determines the integrity checksum of the VS along with the MG itself.
- The integrity checksum is determined by computing a one-way hash function across the VS code, the MG code, MG's public key and the generated transport key. See Figure 18.
- The VS encrypts the transport key with the SS public key.
- The VS sends the encrypted transport key to the SS along with the integrity checksum.
- The SS verifies the checksum. Since the SS can extract the transport key, and since it knows the VS instructions, the MG instructions, MG's public key and the generated transport key, the SS can then generate the checksum. The SS verifies the calculated checksum against the received checksum. If the verification is successful, the SS encrypts the corresponding trust interval's media key with the transport key and sends it to the VS.
- The VS now can decrypt the corresponding piece of the EEMD. See Figure 19.

In the next section, we will introduce a protocol model checker called AVISPA. Then, we will validate the key exchange protocol using the AVISPA tool.

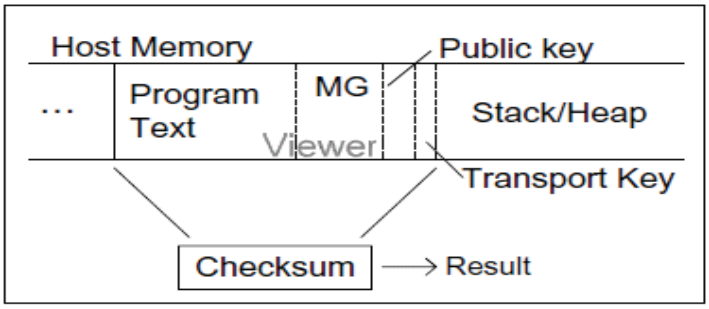

Figure 18: Grimen et al. hash function calculation [GMM06b]

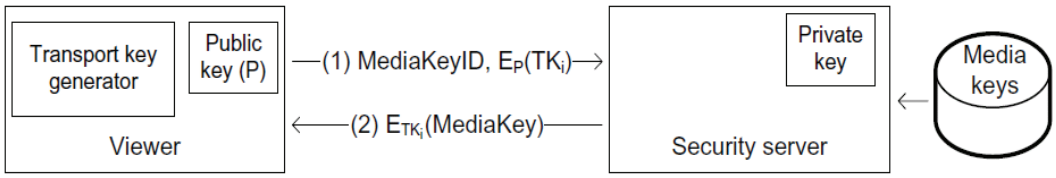

Figure 19: Grimen et al. key exchange protocol [GMM06b]

6.3 Automated Validation of Internet Security Protocols and Applications

The Automated Validation of Internet Security Protocols and Applications (AVISPA) is a model checker verification tool and is also considered a descriptive language that is useful for validating cryptographic protocols. A group of researchers has evaluated 85% of the IETF protocols using the AVISPA tools; this indicates that AVISPA is powerful. The AVISPA architecture is composed of four verification utilities or back-ends [Tea06a, HS06], see Figure 20. Each back-end uses different verification techniques:

- OFMC stands for On-the-Fly Model-Checker.
- Cl-AtSe stands for CL-based Attack Searcher.
- SATMC stands for SAT-based Model Checker.

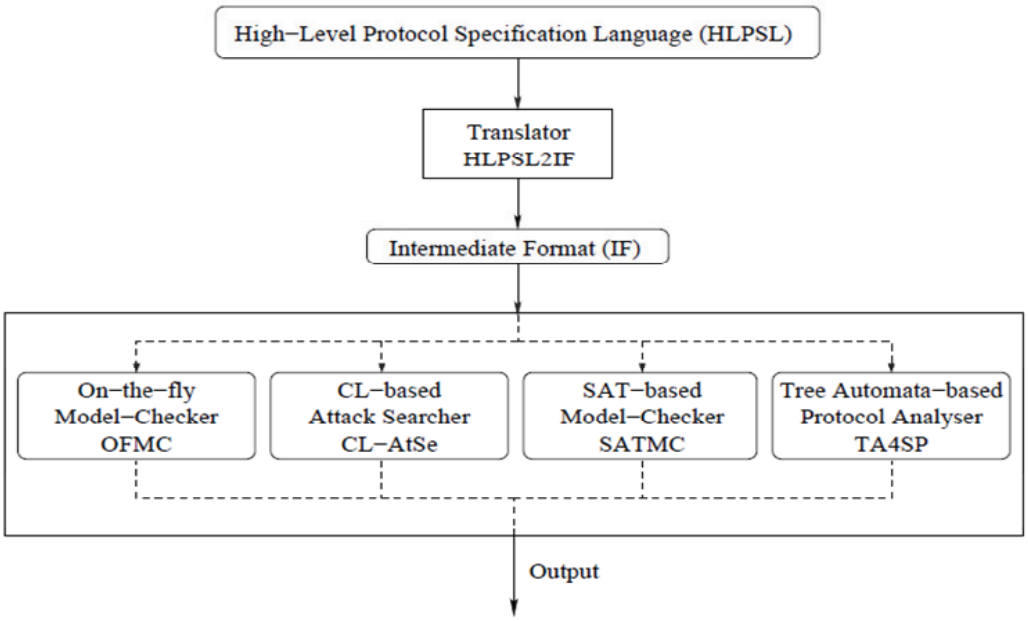

Figure 20: HLPSL architecture [Tea06a]

- TA4SP stands for Tree Automata based Protocol Analyzer.

In real life, interacting users that use the public network need to authenticate each other before securely exchanging data. Using a public network, this authentication is achieved by using a cryptographic protocol. It is important to verify cryptographic protocols to validate their security properties.

In AVISPA, security protocols are modeled with the high-level protocol specification language (HLPSL), which is considered a Meta Language that can be translated into intermediate format (IF). The IF can be used by the four utilities, which are able to understand it.

HLPSL is a role-based formal language; the state of the interacting roles in HLPSL is constructed using the idea of finite state machine (FSM); thus this language is considered an expressive language. In AVISPA model, we can represent interacting entities as roles, each has a state and possible transition(s). The transition is

represented by exchanged messages between participants. HLPSL language has the capability to model security messages and also to show if the security properties are maintained. AVISPA helps to catch vulnerabilities in the security protocols. There are hierarchical roles in HLPSL [Tea06b]:

- Basic roles: they represent interacting parties.
- Session role: it contains composition roles, which sketch how basic roles communicate with each other.
- Environment role: it works as a main coordinator, which declares constant variables, intruder knowledge and sessions call.

Goal section is one component of the HLPSL program. In that section, you can assert one of three security checks:

- Message secrecy: e.g., secret(T,id,A,B), which asserts that the value of token T, that has protocol_id labeled "id" used to identify the goal, is only shared between role A and role B.
- Weak authenticity: e.g., the end user sends a token and claims that he is a specific identifier; the receiver needs to look for the validity of the token not for the validity of the sender's identity.
- Strong authenticity: the end user sends a token and claims that he is a specific identifier; the receiver needs to look for the validity of both the token as well as the sender's identity.

As seen in Figure 20, AVISPA has a tool called hlpsl2if, which converts the HLPSL specification model into intermediate format (IF). Afterward, the user has the option to use any of the back-ends supported by AVISPA. The selected back-end tool takes the IF language as input and analyses the protocol modeled by HLPSL according

to the specified security goal and properties, and then catches vulnerabilities in the security protocols if any [Tea06a].

AVISPA helps to catch vulnerabilities in the security protocols. The security goals that are supported by the current version of HLPSL are authentication and secrecy goals [Tea].

Let us say that Alice and Bob want to exchange a session key before starting a new session [Tea06a]. Alice starts by asking a trusted third party to help her with the exchange, assuming that the trusted third party server shares a secret key (Kas) between him and Alice, and another secret key (Kbs) between him and Bob:

- A ->S: {Kab}_Kas
  Alice generates the session key Kab, encrypts it with the Kas and then sends the encrypted message to the server who is the only one who can decrypt the message.

- S ->B: {Kab}_Kbs
  The server in his turn encrypts the Kab with the Kbs and then sends it to Bob who can decrypt the message. Then Alice and Bob can securely exchange information using the Kab.

The previous protocol seems to be secure; the properties that both Alice and Bob are looking for is the secrecy of the Kab. To model the previous protocol using AVISPA, we need to initialize three roles' structures: role Alice, role Bob and role server. Every role structure is composed of parameters and a state data structure that represents the role state. The role state can be changed only according to the rules that control the state transition. These rules must be specified in the role structure. A role's state transitions are specified in the transition section within each role.

For example, the following role structure represents the trusted third party server. there are three agents in the parameters attached to role server: A, B, and S. S represents the trusted server. A and B represent Alice and Bob respectively, and that can be extracted from the context of the whole program. Kas and Kbs are some of the parameters and they represent the shared key between the server and Alice or Bob respectively. SND and RCV are the channels that are under dy's control. The channel "dy" stands for Dolev-Yao intruder model [Tea06b, DY81]; it is the only intruder model that is supported by AVISPA. The Dolev-Yao model assumes that the intruder can impersonate another user, intercept transmitted messages over dy channels and may be able to alter messages depending on the knowledge s/he has or acquires [Tea06b, DY81].

```
role server(A,B,S : agent,
Kas,Kbs : symmetric_key,
SND, RCV : channel (dy))
played_by S def=
local State: nat, Kab: symmetric_key
init State := 1
transition
step1. State = 1 /\ RCV({Kab}_Kas) =|>
 State:= 3 /\ SND({Kab}_Kbs)
end role
```

The initial state is one as indicated by init State := 1. In the transition section, the state may change from State = 1 into State = 3 only if the server receives the key Kab encrypted by Kas. In this case, the state is changed and the server sends the key Kab encrypted by Kbs. Roles Alice and Bob can be modeled in AVISPA as the following:

```
role alice(A,B,S : agent,
Kas  : symmetric_key,
SND, RCV : channel (dy))
played_by A def=
local State: nat, Kab: symmetric_key
init State := 0
transition
step1. State = 0 /\ RCV(start) =|>
 State:= 2 /\ SND({Kab}_Kas)
end role

role bob(A,B,S : agent,
Kbs  : symmetric_key,
SND, RCV : channel (dy))
played_by B def=
local State: nat, Kab: symmetric_key
init State := 4
transition
step1. State = 4 /\ RCV(start) =|>
 State:= 6 /\ SND({Kab}_Kas)
end role
```

The signal "start" in the RCV(start) in role Alice is a special signal used to start the protocol, and then the selected backend tries all possible intruder interactions to gain more knowledge about secrets that are used between legitimate interacting parties. Figure 21 shows the finite state machine for Alice, Bob and the trusted server. You can get more information about AVISPA tool from [Tea06a, Tea, HS06].

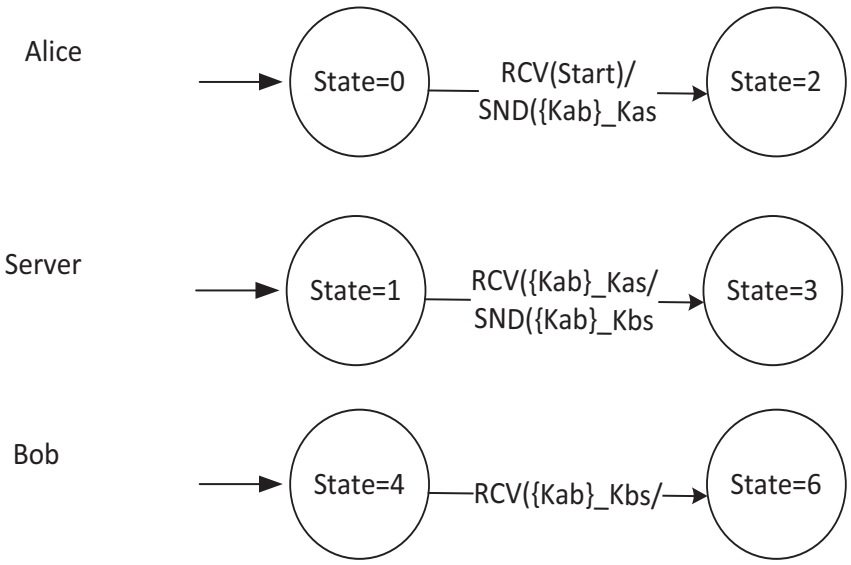

Figure 21: Simple session key exchange protocol represented by FSM.

6.4 Attack on the Grimen Model

We analyzed the presented protocol and found two security attacks. The first attack happens because there is nothing that can prevent the software attacker from forwarding the MG to another VS, and then the forwarded MG produces a new transport key and sends that key to the SS. There is nothing to tell the SS that the MG is a copy, not the original. In this case the server will respond with the media key encrypted with the new generated transport key provided by the pirated MG. Another attack appears because the generation of the transport key is done by the VS, because the VS executable file is stored in the client environment. Thus the client has the chance to statically or dynamically analyze the code and learn the memory address of the transport key.

The main problem with this key exchange protocol comes from the fact that there is nothing to distinguish any instances of MG. All of them have the same features and no individualization technique has been attached to them. The secondary problem is due to the fact that the client has enough time to statically or dynamically analyze

the VS. One way to prevent the previous attacks is to individualize the MG for each client and to store the transport key in an unpredictable place. In the next Section we will discuss how to achieve both goals.

6.4.1 Attack Analysis

We translated the Grimen et al. key exchange proposal into the following messages:

1- MG $\rightarrow$ S: $[Nm.Tki.MG]_{Ks}$. hash(MG.VS.Nm.Tki)
Where MG: The identity of the mobile guard
S: The identity of the security server
Nm: nonce generated by MG for authentication purposes
Tki: transport key for session i, a generated symmetric key randomly generated by MG, used to transport the media key for the ith trust interval session.
Hash: one way hash function.
Ks: public key for S.
VS: is the instructions of the viewer software along with MGs instructions, see Figure 18.
".": concatenation.

2- S $\rightarrow$ MG: $[Nm|Ns|S]_{Km}$
Where Km: public key for MG.

3- MG $\rightarrow$ S: $[Ns|MG]_{Ks}$

4- S $\rightarrow$ MG: $[Mki|S]_{Tki}$
Where Mki: media key for the trust interval ith session. Figure 22 depicts the protocol exchanged between the two roles.

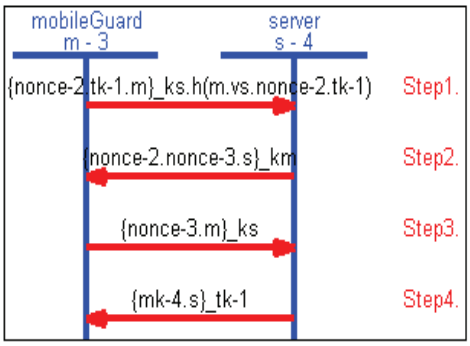

Figure 22: Grimen et al. Key exchange protocol simulation

We translated the previous message exchanges into HLPSL, see Appendix A, and simulated the attack. Figure 23 shows the attack simulation. i represents the intruder entity who is trying to gain access to the media key for all sessions. An intruder is a forwarded MG, i.e., an illegal copy of MG, who knows the public key of MG. The first message shows that the intruder generates a transport key and a new nonce, and encrypts them with SS's public key. Then, the intruder generates a checksum that is a result of applying code instructions into a hash function, these instructions are: fixed parts of the binary code of the VS and MG, transport key and nonce. Then the intruder sends the result to SS. SS extracts the transport key and nonce, and then calculates the checksum code, since the VS has all inputs for the hash function, and then compares the result with received information. Upon successful verification, the SS creates a nonce and then sends the SS's nonce along with MG's nonce all encrypted with MG's public key. The intruder who has the MG's public key can extract the SS's nonce and then encrypt it with SS's public key and then send it to SS. The SS believes that he is talking to a legal MG, and then encrypts the media key for ith session with the transport key. This leads to an attack since the intruder can extract that media key. We assume that forwarding the Mobile Guard to an illegal user is a DRM attack, which means that the illegal user uses indirectly the MG public key or at least the pirated MG can decrypt any messages that have been encrypted with MG's public key.

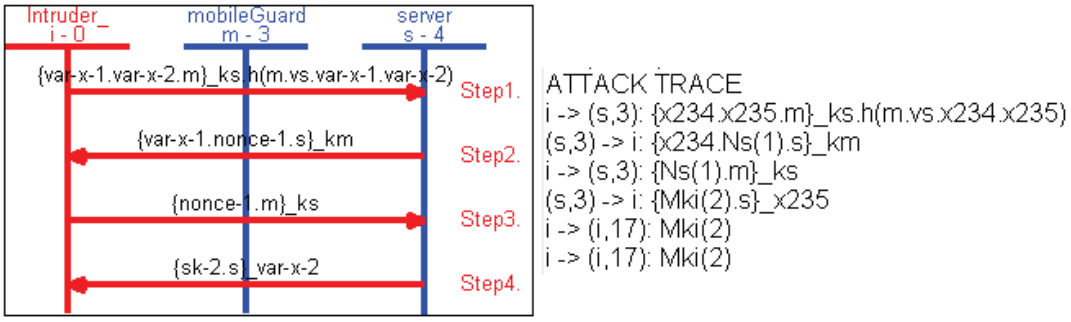

Figure 23: Attack trace simulation for Grimen, et al.'s key exchange protocol

The problem with the Grimen et al. solution is that generating a new transport key from any MG instance does not correspond to the validity of any MG instance, thus any message from a pirated MG is accepted. A solution to prevent the previous attack is to give the ability for the SS to distinguish each MG instance. We can achieve this distinction by individualizing all legal MG copies.

6.5 The Solution

Apple's DRM solution used three keys to protect its products: master key, repository key and client key. The product media is encrypted with the master key, the master key is encrypted with the client key, and the encrypted master key is attached to the protected content media. When the user asks for a license to use the product media, the server that locally stores the client key sends the requesting user a license that includes the client key encrypted with the repository key. The application that presents the product media at the client side has the repository key and is attached with a secret algorithm embedded in the presentation software. The protection mechanism is dependent on keeping the secret algorithm hidden from the user. Each user has a distinct client key and repository, which simplifies the license individualization process. The client key is randomly created by the server and saved in the server's database for further use. The repository key is calculated from serial number of the

first hard disk, BIOS version, CPU name and the ID of windows operating system, which is assumed to be unique [GMM06a].

In the Microsoft DRM solution the product media are encrypted with a content key and the content key is encrypted with client keys. When the user asks for a license to use the product media, the license server sends the license that contains the encrypted content key to the user. In the client side, there is a blackbox file that contains the client key. Each client has an individualized blackbox, which means each instance of a blackbox can only work for a specific machine. The protection mechanism relies on keeping the blackbox hidden from the user access, which simplifies the license individualization process [GMM06a].

OMA DRM standard encrypts the product media with a content encryption key (CEK), the CEK is encrypted with a DRM agent's public key and then inserted into a right object (RO). This process cryptographically binds an RO into DRM agent. Each DRM agent has a unique public and private user key; when a DRM agent asks for a right object to use a specific product, the server, who is responsible for creating right objects, can obtain the DRM agent's public key, and create a unique right object for each DA that includes the CEK key, which is encrypted with the public key of a DA. Only the DA who has the corresponding private key can use that RO and get the CEK [OMA08a].

From the previous solutions we have two software solutions and one hardware solution for the end-user individualization process. The case that we are working with needs to prevent an end user from forwarding the MG to another user. To do this, we need to individualize each MG and make MG code not to work on any other user. We suggest that the SS, which is responsible for generating an MG, needs to authenticate each user and then embed a unique identity (Ticket) in the MG. The security server is going to accept only one request for media decryption key from each legal MG instance per trust interval. Due to the fact that the generation

of transport key is unpredictable to both SS and user, the ticket value along with generated transport key will be the unique identifier for each MG per trust interval. When the SS receives two requests for the same MG that have the same ticket and a different transport key, the second key request will not receive any response. This will prevent a pirated MG instance from successfully attacking the model previously discussed. Here is the modified protocol:

1- MG $\rightarrow$ S: $[msg]_{Km}$. hash(MG.VS.Nm.Tki.Ticket)
   Where msg = $[Nm|Tki|MG|Ticket]_{Ks}$

2- S $\rightarrow$ MG: $[Nm.Ns.S]_{Km}$

3- MG $\rightarrow$ S: $[Ns|MG]_{Ks}$

4- S $\rightarrow$MG: $[Mki|S]_{Tki}$ Where Km here is a symmetric shared key between the MG and the SS, and the Ticket is the unique value for each generated MG.

Before SS delivers an MG to a valid VS, it embeds a symmetric key Km into the generated MG. If a legitimate user receives a valid MG, and then forwards that received MG to his friend, both MGs will have the same identity (ticket). Each MG is going to run and generate a unique transport key at the client host; this makes a checksum calculation unique since the transport key is one of the hash function inputs. We will call the result of the hash function a checksum. When both MGs send the first message, which contains: nonce, random generated transport key, mobile guard identity and ticket all encrypted with the SS's public key and then again encrypted with shared key (Ks), the second part of the massage is the checksum. The SS can extract the ticket value and the transport key and use them to calculate the checksum. The SS compares the received checksum with the one it calculates, if both match then the MG has sent correct information. The SS checks the ticket value, if it receives it for the first time, then the MG is legal, so it will respond with the second message. If it receives the ticket for the second time, and if the checksum is the same, then it can

safely reply to the second message, assuming that its previous response was lost in the network. If the SS receives the same ticket with a different checksum, this means that an illegal MG is sending the second request, in this case the SS will not reply to the request. The second message contains a generated nonce from the SS side, MG's nonce and the SS identity all encrypted with the shared key Km. The third message is from the MG side and contains: SS's nonce and MG's identity all encrypted with SS's public key. Now the SS and MG are mutually authenticated, the forth message is from SS side and contains the media key for the ith session and the SS's identity all encrypted with the transport key for ith session.

To prevent the software hacker from discovering the transport key in the VS space by using static or dynamic analysis, the MG needs to create a random Transport Key and store it in a random place in the VS, or keep it in MG space, note that the contents of MG is not predicted for a short time. The VS should implement a way to call the transport key generation from the plug-in MG instance. This will prevent the end user from knowing the location of the transport key for a short time.

In the solution we provide, the SS only accepts the first media key request for each unique ticket, and rejects any subsequent request for the same ticket with different checksum. We model the proposed solution for Grimen et al. protocol and validate it using AVISPA tool. We show the HLSPL code in Appendix B. We ran the new protocol on AVISPA and did not catch any attack. We therefore believe that our protocol is correct and helps the SS to authenticate valid instances of MG.

6.6 Summary

In this chapter, we discussed Grimen, et al. model that is assumed to achieve persistent protection for delivered content media. The protection used in the model is based on software-based protection, which makes it fast to be deployed and flexible to be

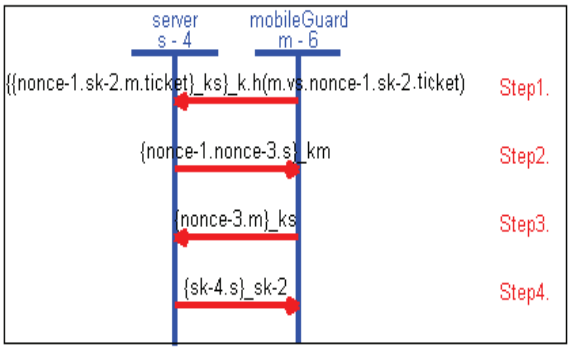

Figure 24: Revised Grimen, et al.'s key exchange protocol simulation

distributed among an acceptable number of users. Grimen, et al. make the software based protection valid for a short time period called a trust interval, and this software protection is periodically changed. They assume that the protection software cannot be defeated within the trust interval because Grimen, et al. assumed obfuscated protection software, and they have added a validation mechanism to the protection software. Grimen, et al. assume that they can provide an extended range of persistent protection to the delivered media content. The assumption the author made was valid because the protection method is changed once the trust interval expires. We have verified the assumption claimed; we used AVISPA model checker to verify the protocol used in the Grimen, et al. model and found that there were defects in the protocol that claimed to achieve persistent protection for delivered content media. We have fixed the protocol and then we have verified the modified model and found that it is working with no bug. Thus, the software hacker has little opportunity to hack the improved protection solution even if s/he is given enough time and suitable tools. The only problem with the model is that it is based on unicast distribution, and thus it is not scalable.

To conclude, persistent protection that is a result of using software based protection is flexible and a promising solution for CP to deploy. We found our target, flexible persistently protected method, especially when the clients are general purpose computers or laptops. The scalability of the model can be improved if we change the

underlying distribution mechanisms from unicast distribution into multicast distribution, and that is our next step.

Chapter 7

Scalable Persistent Protection in Multicast Content Media Delivery

Now, we can observe that many DRM systems suffer from using the same protection scheme for a long time, which gives the software hacker the opportunity to hack the protection solution given enough time and suitable tools. Grimen et al. [GMM06a, GMM06b] analyzed DRM weaknesses, and then proposed a secure software-based DRM system that prevents the DRM attacker from penetrating the DRM system for a short period of time. However, their solution is not scalable because changing the key as well as the mobile guard every short period will exhaust the SS. Although care has been taken, by Grimen et al.'s model, to replace the protection mechanism every trust interval, the Grimen model does not prevent a user from forwarding the mobile guard to his friends, which is a flow that allows illegal end users to illegally access protected media. We found that error in the Grimen model and presented a corrected version, see Chapter 6.

It is known that multicast as a distribution mechanism has the advantages of lowering the price of distribution, speeding up the delivery process, lowering the

resource usage of both the source and network and improving the network bandwidth [J. 07]. We presented Multicast Security Architecture (MSA) in Section 3.3, which works side by side with scheduled content media distribution. The main problem with the multicast security model is that it does not prevent a dishonest customer from forwarding any delivered data.

The goal in this chapter is to find an implementation and an architecture that permits changing the underlying Grimen et al. model from unicast into multicast transmission mechanism. In our opinion, if we could marry the multicast security model with the Grimen et al. model, in this case we could acquire the advantages of both approaches: scalability and a long period of persistent protection.

In this chapter, we propose a scalable model that enables a periodic replacement of the protection mechanisms, thus extending the lifetime of the protection. We then explore how to apply the ideas in the case of multicast data distribution. After detailing the protocols between the major parts of this model, we show the results of a formal validation of the security of these exchanges.

This material provides the scalability that is a necessary prerequisite to meeting our second goal.

7.1 Improved MSA Model

To increase the scalability of current DRM systems, we need the network service provider to contribute to the solution. The NSP is able to authenticate the individual customers, deliver the service to their customers, and account for their resource usage [IA06], as well as monitor end user's behavior. In our opinion, adding persistent protection properties to the multicast content media distribution system is the way to close the security holes generated by dishonest customers. Satisfying the

scalability requirement, which leads to the use of multicast distribution, is hard to achieve without the help of the network service provider. Buyens et al. [BMJ07] stated that building and implementing DRM must consider the following fuctional requirements in the DRM development phase: interoperability, modifiability, extendability, usability, testability, availability, security, scalability and performance. They recognized key DRM blocks for licensing, access, content media handling, abuser identification, content media importing, consumer tracking, and payment. Michiels, et al. [MVJDD05] determined DRM services from different view points: consumer, producer and publisher. They proposed a layered architecture that is composed of multiple DRM services. As hinted by Buyens et al. and Michiels, et al. we strongly believe that we need the following services:

1. Content media protection (Encryption service): due to using the public network for delivering content media, the most feasible way to control the content media access is to encrypt the content media.

2. Authorization service: A service to authorize customers who are entitled to see the content media by delivering them the capabilities to access protected content media such as the encryption key.

3. Trust service: A service that provides the trust relations that need to be established between the sender and the distributor, the distributor and the receiver, as well as the receiver and the received objects. Usually, the trust service is established based on content media, sender and receiver authentication.

4. Usage accounting: A service to account for the content media usage.

5. Policy statement: A service that allows the content owner to specify terms and conditions, usage rights and the payment fees, as well as any related issue for delivery agreements. There is a need for a rights expression language (REL) to express rights and policies.

6. Assets unforgeable proof: A service that provides a proof that the assets production is generated by an authorized entity. It is achieved by signing integrity hash-value. The signature is done by CP, so it gives a proof for originality.

7. Secure storage: A service used to hide sensitive information. Hidden information should not be revealed unless an authorized entity allowed it.

8. Rights enforcing: an authorized protection agent that is authenticated by content owner and that has the responsibility of decrypting the content media. That agent has the decryption key and is responsible for hiding this key away from the customer as well as enforcing terms and conditions.

9. Integrity insertion: A Service that enables the content media integrity verification.

10. Integrity checker: A service that works as a detective and that has the responsibility for checking the validity of the received product.

11. User privacy assurance: a service that guarantees that the user's private information is not in the hands of a privacy attacker.

12. Individualization service: this service enables the distributor of the hardware-based or software-based protection to distinguish between diffrent instances of them, and prevent the attacker from extending his penetration into wider environments.

13. Watermark and fingerprint injection service: this service injects a unique ID that represents either the owner of content media or the paying customer, so that it enables the traceback to the source of illegal distribution.

14. Web spider: search for compromised content media across the Internet and discover the illegality of distributing stolen content media.

Table 4: DRM Basic Requirements VS DRM Services

DRM requirement	DRM services
R1.A	Content media protection
R1.B	Trust service Secure storage Right enforcing
R2	Authorization service Usage accounting
R3.A	Policy statement
R3.B	Content meida protection
R3.C	Right enforcing
R4	Integrity insertion Integrity checker
R5	Assets unforgeable proof
R6	User privacy assurance
R7	Multicast distribution
R8.A	Individualization service
R8.B	Watermark/fingerprint injection Web spider
R8.C	Monitor Protection scheme renewal
R9	Secure storage
R10	Multicast distribution
R11	Not yet determined

15. Protection scheme renewal: this service determines the exact period needed to replace the protection scheme, revokes the old protection scheme, and performs the replacing action.

16. Monitor: this service tracks the behavior of the end user, it periodically sends feedback to the server monitor, if certain condition(s) are met.

Table 4 maps previous services to the eleven requirements we collected in Chapter 5.

In the generic DRM model the network service provider has no role in the DRM plane. It works as a transporter, and thus the relationship between the CP and DA is one-to-one. In the world of multicast content media distribution the content provider cannot directly identify, authenticate, authorize, account and track customers, as a

result of moving the provider-receiver relationship from one-to-one into one-to-many or many-to-many. The provider is only responsible for the production related issues. The network service provider is the best place for doing the customer identification, authorization, accounting and tracking [J. 07, IA06]. Based on this observation, we suggest allocating the previous 16 services into the following roles:

1. The content provider (CP): is the entity that is responsible for providing the content media to the end users, it could be responsible for:
   (a) Content media protection.
   (b) Trust service.
   (c) Integrity insertion.
   (d) Policy statement.
2. The Merchant (MR): a role that is responsible for providing a valid token to a valid requester, it is responsible for enabling the following services:
   (a) Assets unforgeable proof.
   (b) Rights enforcing.
   (c) Protection scheme renewal.
3. The content media distributor: in the context of multicast service, the major distributor is the network service provider, this role could be responsible for:
   (a) Usage accounting.
   (b) Individualization service.
   (c) Watermark and fingerprint injection service.
   (d) Web spider.
   (e) Monitor.
4. The mobile protection (MP) agent: a role that is responsible for:

(a) Integrity insertion.

(b) Content media protection.

(c) Trust service.

(d) Rights enforcing.

5. The mobile guard (MG) agent: a role that is responsible for:

(a) Integrity checking.

(b) Authorization service

(c) Providing a secure storage.

(d) Sender authentication.

(e) Rights enforcing.

We have not allocated a place for user privacy assurance. However, we believe that this service will not affect the DRM workflow, so we will leave it for future study.

7.2 Persistent Protection in Multicast Content Media Delivery Model

In this section we suggest a persistent protection model for content media using a multicast distribution model. We strongly believe that merging ideas that exist within the Atwood and Grimen models will result in a persistent protection and scalable content media distribution model, so we build our solution based on MSA.

If the majority of the end users are using general PCs, it is not feasible to use tamper proof hardware to hide the protection secrets. A better solution is to use tamper resistant software that provides a flexible and pluggable solution. MSA protects the

content media in the network layer, but does not protect it in the application layer. We suggest adding new functionalities to MR and NSP, which inject a DRM agent to an individual EU. We will use the same name as in the Grimen model for this agent, a mobile guard plug-in (MG). We will ignore the FI role at this stage. The MG (tamper resistant object) helps to achieve application protection [GMM06a, GMM06b], it embeds the decryption algorithm and decryption keys that are needed to decrypt the stream media as well as other mechanisms to enable monitoring the EU. We will add a mobile protection (MP) role, which embeds mechanisms for protecting content media. The CP and the end user have to have no access to these secrets in order to reach a global solution.

The MR is responsible for generating the MG, MG authorizes the EU to legally use the content media in a controlled manner. The MG has embeded in it the terms and conditions, enforcing and decryption mechanisms, and other mechanisms for checking and validating the EU environment.

The NSP is the entity that can easily track and account for the EU's activity, so the generated MG should be delivered to the NSP who in turn is going to securely deliver it to each individual EU. Because the NSP does not trust the EU, employing the mobile guard is a way to load a method on the customer's device. A Java applet is a self-contained program that loads mobile code in a remote machine, and executes that code under a standard virtual machine. Java has come to be trusted by the end user community, so this represents a low-risk approach for implementing the ideas. We assume that a contract is established between the following entities:

- The NSP and the MR.
- The MR and the CP.

We assume that the following assumptions are valid:

1. The CP trusts the MR.

2. The NSP trusts the MR.

3. The CP is specialized for generating and encoding the content media.

4. The MR is specialized in selling content media and generating the encryption and decryption mobile code for content media: mobile protection (MP) and mobile guard (MG) [GMM06b].

5. The MP and MG are executable plug-in files.

6. The attacker does not have advance knowledge about the forming, content and production of the MG and MP.

7. The network service provider and the end user share a preshared-key, and will call this key an access-key.

8. The merchant and the end user share another access-key.

9. The attacker does not have advance knowledge about any access-keys shared between the NSP and the EU, and the MR and EU.

There will be a mechanism that establishes the access-key between the NSP and EU, or the EU may receive this access key by telephone, email, or by buying an long-distance calling card, and that card contains the access-key.

Similarly, there exists another access-key between the MR and the end user, and this could be established by creating an account at the merchant side, and then the merchant uses the account name and the user password to generate the access-key. The end user is able to generate the same access-key if s/he follows the same procedure.

We relaxed the persistent protection definition, and in our context, it is defined as a protection with specified permission and constraint; that protection is applied to a

specific media and it is valid for a specified time and place. In order to implement this definition, we need to concentrate on protecting the protection software from reverse engineering and related attacks, in other words, disallow illegal use of protection software. The following list summarizes the threat model:

- Code modification attack.
- Forwarding and illegal use of protection software.
- Reverse engineering attack.
  Previous items are applicable to protection software.
- Message replay attack.
- Man-in-the-middle attack.
- Message integrity attacks.
  The last three points are usually applicable to the security protocols

In the following subsections, we will show the new multicast model for persistent protection distribution. The new model is an improvement to both Atwood (MSA) and Grimen et al.'s model [J. 07, GMM06a, BA10].

7.2.1 The Content Provider (CP)

The CP is considered an artistic director to the art production, s/he is responsible for generating and encoding the content media using the right technology tools, then dividing each content medium into multiple pieces, where each piece has the same duration. The CP uses specific protection executable software on each piece, this protection software is provided by the MR. The resulting protected pieces are called the encrypted and encoded streamed content media document (EEMD) [GMM06b, GMM06c].

The CP also responsible for preparing the meta data that will guide the EU where to get the content media. The CP provides terms and conditions to the merchant that he wants to be applied on the content media. The MR is going to record them in its database, and hard code them in the generated MG.

7.2.2 The Content Server (CS)

The EEMD piece is a secure content to be distributed only at specific time. The CS is responsible for starting the multicasting of each piece of the EEMD stream at the scheduled time.

7.2.3 The Merchant (MR)

The Merchant is responsible for managing, advertising and selling the content media. S/He works as an interface of the CP to the whole world. When the CP generates the content media, s/he contacts the merchant and sends all required information about the content media to be registered. The MR generates the suitable software codes called mobile protections (MP), which are needed to protect and package the content media. The MR generates the enforcing mechanisms for the terms and conditions and embeds them into MG. The MR may embed a mechanism that footprints individual instances of the EEMD. The MR receives requests from end users who are interested in watching one of the goods, and provides them with tickets to use later to show their eligibility for using the content media. The MR provides the NSP with all lists of MGs that are capable to decrypt and enforce access right conditions on the requested EEMD for each individual trust interval.

7.2.4 The Mobile Security Codes (MP and MG)

The MR generates a list of mobile guards (MG), and mobile protection plug-in software (MP), for each content medium. The MP embeds mechanisms to encrypt a sequence of sub-content media with different encryption keys and mechanisms, while MG embeds the capabilities to decrypt the sequence of sub-content media. The MG also has the capabilities to enforce the terms and conditions stated by the CP, as well as, the MG should able to validate and authenticate the end user machine used to access content media.

7.2.5 The Network Service Provider (NSP)

Since we are suggesting to use multicast as a distribution mechanism to convey the content media from content server into many end users, the only feasible way for content media usage accounting, in our opinion, is to asign this task to the NSP. The reason behind our belief is when we use multicast as a distribution mechanism, and for scalability requirement, the CP and the MR need not to be aware of the existence of the end users, only NSP can be aware of end users without degrading the scalability requirement [J. 07].

The NSP receives a ticket from the EU as proof for his/her eligibility to access the content media. The ticket is valid for a certain time period and is unique for each request. The NSP is responsible for validating any request from the end user and contacting the MR to request for getting the suitable MG for the requested media at specific time interval. Once the NSP gets the suitable list of MG, s/he clones and individualizes each of them by inserting the user ticket in a predetermined place in each cloned MG, this place cannot be predicted by the individual EU for short period of time. This individualization helps to trace back the source of illegal distribution in order to mark him/her “Wanted”.

7.2.6 The End User (EU)

The end users are users of machines such as laptops, personal computers, cell phones and so on. The EU searches through the web for some content media to watch, then contacts the MR who is responsible for issuing him a ticket. The EU buys a ticket and then provides that ticket to the NSP, who in turn will connect him to watch the media. The EU needs to install a standard software called viewer software (VS) developed by the standard third party, Section 7.2.7 gives more details of the viewer software.

7.2.7 Viewer Software

The following description of the VS ideas comes from [GMM06b, GMM06c].

Every client machine needs to use standard viewer software (VS), which been developed by the MR. The VS is developed in such a way as to accept an instance of MG. The MG hides a decryption key for a specific piece of the EEMD, and then the VS renders the result on the screen of the client machine. The VS mainly contains two parts: fixed and dynamic parts. The fixed part consists of the text code and some parts of the data section that needs to be unchanged. The dynamic part is the plug-in software, which is dynamically attached to the memory space of the VS, which is MG. Applying the fixed part of the viewer software into a one way hash function leads to a resulting hash-value that is known by all parties.

The plug-in software also has a fixed part and dynamic part. The fixed part consists of the instructions and constant static data. Apply these fixed parts to one way hash function results in a hash-value which should be known to the generator of the plug-in software. We will use these facts to check the validity of VS and plug-in software attached to the VS.

The plug-in software is responsible for carrying out the decryption mechanism and the decryption key in order to decrypt a specific EEMD piece. The plug-in software also provides validity checking of the VS along with the plug-in software itself. We will call the plug-in software mobile guard (MG) as been called by Grimen because it checks the destination viewer and ensures its validity and security. In the next section, we will present the protocol between NSP, MR and EU of the new multicast content media distribution that is scalable and provides persistent protection to the distributed content media.

7.3 The New Model's Workflow

We suggest the following workflow for the new system:

1- The MR generates two classes of executable plug-ins: mobile protection (MP) and mobile guard (MG). The mobile protection is mainly responsible for encrypting and packaging a content media piece; on the other side, the mobile guard is mainly responsible for unpackaging and decrypting a piece of content media. Both classes are also responsible for checking the integrity of the host that one plug-in instance is going to be hosted in.

2- There will be a series of mobile guards MGi and a mobile protection MPi. MGi and MPi symbols represent MG and MP instances. The MGi will be only valid for a certain period of time, we will call this period "trust interval" as has been advised by Grimen [GMM06b, GMM06c]. Every trust interval should be served by only one instance of MGi. Suppose that the trust interval is going to be one hour, then there will be 24 MGi to cover one day.

3- The CP generates and encodes the content media, then divides it into multiple pieces of a certain time slot.

4- The CP contacts the MR to register the media information and specifications such as the time schedule to disseminate the content media, number of pieces of the protected media, the usage policy, terms and conditions for using the content media, stars, genre and the price of the content media.

5- The MR generates a mobile protection that is responsible for generating the EEMD. It then sends the sequence of MPi to the CP. Also, the MR generates a corresponding list of plug-in MGi for unpackaging the content media and reflecting the terms and conditions that the CP wants to enforce at the client side.

6- The CP generates EEMD. The CP may have the option to select different properties for the encryption code, such as, which cryptography and authentication algorithm and key length. The CP sends EEMD to the CS, and gives him the time to start distributing the content media.

7- The MR advertises the content media and provides the time schedule, usage policies and corresponding meta data for the content media. S/He is responsible for selling the tickets for the content media.

8- The EU contacts the MR and gets the needed information about the content media.

9- The end user creates an account at the MR site, and provides some necessary information such as name, address, NSP identity and so on. The created account has account login and password. That password may used as an input to hash function to generate a unique secret key.

10- The EU provides the needed funds to watch a specific content media. The MR creates a ticket and sends it the end user. The client gets a ticket.

11- The ticket is used as a proof of successful payment. The ticket composed of ticket ID, a nonce generated by the MR, a nonce generated by the VS, the MR ID, the NSP ID, the EEMD ID and client ID.

The idea behind using viewer software is to prevent the user from directly accessing the content media, or accessing any capabilities that allow him/her to access the content media. The viewer software works as an interface between the CP and EU. In one side it allows the EU to indirectly use the content media, and on the other side, it enforces the terms and conditions that need to be applied on the content media. Since the VS is software that lives in the user's environment, the EU may have full control on the machine he is using, which may lead him to hack the VS. To prevent EU from hacking the VS, we need a periodic check of the VS. If there is any attack on the VS, then there should be a mechanism to stop sending the media flow to the compromised VS.

Since the NSP is the Internet interface to all EU, we will add VS validity checking function to his/her responsibilities. It is more feasible for him to do that than for other entity. One way to check VS validity, is to send checking software to the EU side, we call it "mobile guard". That mobile guard has the properties that have been mentioned in Section 7.2.7, which allows the periodic configuring of the VS by periodic renewing of the mobile guard itself. If the validity check is not satisfied, the NSP stops sending the media flow and any subsequent MG to the suspicious VS. If the validity checking is passed, a specific signal is sent to the VS to start doing the decryption and encoding process to the media piece that is under flow.

Next we will provide the authentication protocol that applies between the CP, MR and VS before the VS receives the MG plug-in.

7.4 Authentication Protocols among MR, NSP and EU

In this section we will start with the authentication protocols between MR, NSP and VS before the VS receives a new code piece allowing him to decrypt and encode the content media peice. Here is the protocol control flow:

1- The end user creates an account on the MR site, and provides some necessary information such as name, address, NSP identity and so on. The created account has account login and password. The account, password and nonces generated by the two parties (EU & MR) are be used as an input to a hash function to generate a unique secret key.

2- The EU contacts the MR via the viewer software and views posted information about the content media, then sends a request to buy a ticket for the content media s/he is interested in. The EU generates a nonce and provides the needed funds to watch a specific content media.

VS $\rightarrow$ MR: VS.NSP.MR.$Request_{Kvm}$

where:

"A $\rightarrow$ B: Message": means A sends B a message.

. means text concatenation.

Request = $[Nvs.VS.NSP.MR.Minfo.Money]_{KMR}$

VS: End user's viewer software's identity.

MR: The merchant's identity.

Nvs: is a nonce generated by VS.

NSP: Network service provider identity.

Minfo: media information.

KMR: is the public key of the MR.

Kvm: is the symmetric key (access-key) shared between VS and MR.

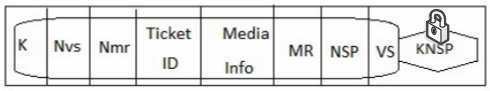

Figure 25: The content of the ticket.

Money: is a token that represents sufficient money for the requested media.

3- The MR generates a nonce and a ticket and sends them to the end user. The client gets a ticket.

MR $\rightarrow$ VS: VS.NSP.MR.$Response_{Kvm}$

where:

Response: $[VS.NSP.MR.K.Nvs.Nmr]_{Kvm}$.Ticket.

Nmr: is a nonce generated by MR.

K: is Symmetric key generated by MR.

Ticket: $[K.Nvs.Nmr.TicketID.Minfo.MR.NSP.VS]_{KNSP}$.

KNSP: is the public key for the NSP.

TicketID: is the identity of the ticket.

MR: is the merchant identity.

4- The ticket is used as a proof of successful payment. The ticket is composed of Ticket ID, nonce generated by VS, nonce generated by MR, MR ID, NSP ID, VS ID and a symmetric key generated by the MR, see Figure 25.
Notice that the client cannot open the ticket, since it is encrypted by NSP's public key. Only the NSP can open the ticket and extract its content media.

5- For validating the ticket, there will be policy statements that have been negotiated between the NSP and the MR for that purpose. For example, one policy statement may say "refer every ticket to the MR for verification purposes", while another may say, "if the ticket starts with a specific properties and has been signed by the MR, accept it and then send the money later". This will give

flexibility and decouple the interaction between the NSP and the MR. Either the NSP is going to get information about the ticket's validity or is going to get a template so that he can match or he is going to get a policy statement on how validation of tickets is taking place. Also the NSP has the policy that defines what has to be done on every step of an interaction between him and any other entity. We will not give any detail on any financial and marketing interactions, which is not the concern of our DRM solution.

6- Here is an example of how the NSP validates the ticket. The VS sends the NSP the ticket:

VS $\rightarrow$ NSP: VS.NSP.MR.$MediaAccessRequest_{Kvn}$

where:

MediaAccessRequest: Ticket.$[Nvs.Nmr]_K$

Kvn: is a shared key (access-key) between NSP and VS. Since the EU is customer of the NSP, there will be a secure tunnel established between them.

7- The NSP send the following request to the MR.

NSP $\rightarrow$ MR: TicketID.$ChkTkReq_K$

where:

ChkTkReq: $[Nvs.Nmr.NSP.MR.VS.TicketID]_{KMR}$

KMR: is the public key of the MR.

8- After the ticket has been verified, the MR sends all MGs that are responsible for decrypting and enforcing conditions on each piece of the EEMD. Sometimes, the MR needs to be updated with the number of customers connected to the NSP for the purpose of accounting.

MR $\rightarrow$ NSP: TicketID.$ChkRes_K$

where:

ChkRes: $[Nvs.Nmr.NSP.MR.VS.TicketID.MG]_{KNSP}$

MG = PCi.MKi

The Mobile Guard includes the decryption keys, public and private keys, enforcement mechanisms and integrity checking mechanisms.

PCi: is the protection code for all media pieces that are distributed at trust interval i.

MKi: is the media key for all media pieces that are distributed by the CS at trust interval i.

9- The NSP individualizes the MG by injecting the VS identity and the ticket, which is unique for any individual client. Then s/he provides the VS with the MG that will help in decrypting the content media flow when CS starts distributing it.

NSP $\rightarrow$ VS: VS.NSP.MR.$[Nmr.Nvs.MGi]_{Kvn}$

where:

MGi: $[PC.Nmr.Nvs.VS.MKi]_K$

MGI is an individualized piece of code for the specific VS. Note that the injected MKi within the MG may not be the true media key, and the correct MKi will be sent by the NSP only after the VS and MG verification is completed.

10- The VS calls the MG and allows him to run, then the MG will compute the hash-value of the MG along with the VS, and then send the computed value to the NSP.

VS $\rightarrow$ NSP: VS.NSP.MR.$[Nmr.Nvs.HashValue]_K$

where:

HashValue: the hash-value of result in applying the VS and MG to a specific one-way secure hash-function.

11- Since the NSP has all information to compute the hash-value, s/he is able to compare the received hash-value and validate the end user's integrity, then s/he send the right media key to the end user.

NSP $\rightarrow$ VS: VS.NSP.MR.$[VS.NSP.Nvs.Nmr.MK]_{Kvn}$

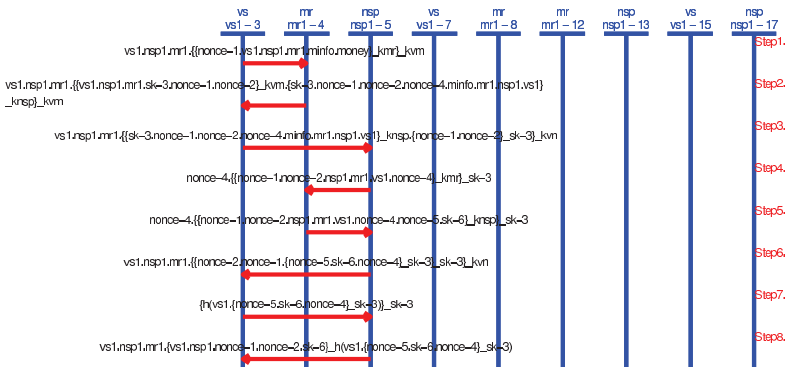

Figure 26: Protocol negotiated between VS, NSP and MR

Figure 26 illustrates the protocol simulation between the VS, the NSP and the MR in order to deliver the MG instance to each client.

7.5 Discussion on the Protocol

The end user interacts with the NSP and MR via the VS. The VS works as an intermediate channel between the EU and the NSP and MR. The end user has the ability to extract the working logic of the VS, so it is susceptible to abuse by end-users due to their static framework.

When the VS receives an individualized instance of the MG, then the resulting VS itself is individualized. If the end user tries to forward the individualized VS or MG to others, this forwarding needs to be invalidated. It is required that the VS execute the received MG, which computes the hash-value of the individualized VS and sends the result to the NSP. The hash-value is a result of secure one-way hash-function; the input of the hash-function is the static part of the VS plus the static part of the received MG, end user ticket, shared-key and MG's public and private keys. The

NSP knows in advance the content and the values of all previous fields, but the end user must not know. Once the NSP receives the correct hash-value, then s/he will respond by sending the sub-media-key to the requesting VS. When an MG computes the hash-value, theoretically, the result is a unique value. When the NSP receives the hash-value, s/he accepts the first valid request and invalidates any other request that carries the same hash-value, assuming that any other request that carries the same hash value is a result of forwarded VS or/and MG. This will prevent forwarding and illegal use of protection software.

The end user cannot modify the VS without being detected by the NSP, because any modification on the VS will lead to a wrong hash-value generated by the MG, Knowing that the content of the MG is not predicted by the end user within the trust interval [GMM06a, BA10], this will prevent code modification attack and reverse engineering attack.

What we discussed so far is a software solution, which is more flexible than the hardware one. Generating multiple instances of obscured PMG and then multicasting them to valid customers via the NSP is an economically feasible solution, especially when the majority of these customers are general PCs [GMM06a]. Knowing that general PCs do not include any tamper resistant hardware, distributing such hardware among users will not be a cost effective solution, because the end user does not appreciate more cost, as well as the NSP or the CP or any other participants.

Since our intent is to maintain a secure environment and prevent message replay attack, man-in-the-middle attack and message integrity attacks, it is necessary to validate the proposed protocol. We used the AVISPA tool to validate the previous protocol, the property we set is to validate the following:

- Authentication on nonces generated by VS and MR
- The secrecy of the tickets generated by MR

- The secrecy of the mobile guard generated by MR
- The secrecy of session key K that needs to be shared between VS and NSP
- The secrecy of the media-key.

The result of running the protocol validation shows that the attacker has no chance to discover the ticket's contents, the MG's contents, the value of the session key K and the value of the media-key. The attacker cannot discover the value of the VS's nonce and the MR's nonce. This proves that both authentication and secrecy goals are satisfied. We use the four back ends that are supported by AVISPA in this protocol validation, and the result out of the four back ends shows that the protocol is safe.

What we model is the protocol that takes place between the major parties: the NSP, the MR and the EU. Other interactions that happen between the MR and the CP, and between the CP and the CS are one-to-one interactions, and that kind of interactions is secure assuming that there a secure channel between the MR and CP and another secure channel between the CP and CS. We believe that Challenge/Response Authentication Protocol, version 2, to establish an authenticated one-to-one connection is enough [TH]. The interactions between the financial institution and other roles within MSA architecture is been validated by Parham and Atwood [PA11].

Within the trust interval, if the VS wants to change to different channels that are supported by the same merchant and allowed to be viewed using the same ticket, then the NSP sends the new media key encrypted with the established secret key K, that is generated by the merchant and embedded in the ticket. Another possible scenario is that the NSP injects an asymmetric key pair in to the MG instance before sending it the specific VS. Then the NSP encrypts any new media key using the public key. The MG instance's public key is not known to anyone but the NSP. This property increases the security relation between the NSP and the VS, and then subsequently,

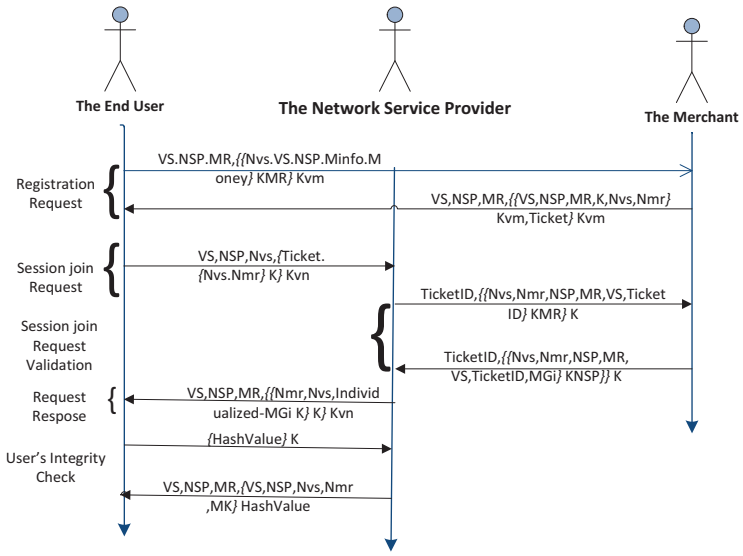

Figure 27: The MG delivery protocol

the MG's private key may used for injecting a unique finger-print into received sub-media document, which can be used for tracking illegal media distribution.

Figure 26 shows the result of validating the proposed protocol solution, we get the interaction result from AVISPA tool, and the tool shows no attack on the protocol model that we proposed.

Figure 27 depicts the interactions between the VS, the MR and the NSP in order to securely deliver the MG to the VS. In Appendix C we provide the HLPSL model the interactions . We may harden the hacker's task by using obfuscation technique to hide the software protection's logic; also we can hide the messages interaction with a protocol and hiding the messages content. Our protection scheme has many building blocks, and they are as follows:

- Cryptography technology.
- Code obfuscation technology.

- Tamper resistant object.
- Software protection renewal
- Hiding the protocol.
- Hiding the interacting tokens' content.
- Hiding the MG's public key.

We can allow every end user to have a domain access for the content media, in such a way that one ticket allows one user to watch the content media via N different machines or VS instances that are owned by the same user. We could allow that by permitting the user to receive N individualized MG on his N machines or VS instances, and then the NSP counts N hash-value that comes from the same user, Afterward, the NSP sends the suitable media keys for each machine or VS instance. This idea gives the end user the ability to watch N different channels that are supported by the same merchant on N different machines or N different VS instance.

We will call our Model, which is a merger of MSA and Grimen model [J. 07, GMM06a, BA10], persistent protection in multicast content media delivery (PPMCD). This PPMCD satisfies the following:

- Content media protection: Achieved by encrypting the content media at the place owned by CP, as well as using Persistent Protection ideas by hiding the protection mechanisms and keys.
- Trust service: Achieved by hash-value validation by MG or NSP and enabling the MG replacement, which helps to prevent hacking the MG.
- Usage accounting: Achieved by the original MSA architecture.
- Policy statement: Achieved by the original MSA architecture.

- Rights enforcing: Achieved by embedding terms and conditions inside the MG, while the enforcement action is done by VS. The VS is checked and approved every trust interval when a new MG is injected into the VS. The periodic replacement of the MG ensures periodic reassessment of the validity of the VS, which makes the hacking of VS infeasible. We assume the hierarchy trust relationships among all interacting parties are valid.
- Sender and Receiver authentication: Achieved by the original MSA architecture.
- Integrity checker: Achieved by the original MSA architecture.
- Individualization service: Achieved by individualizing/injecting each MG instance with user's ticket.
- Watermark and fingerprint injection service: this may be achieved by embedding fingerprint injection mechanism in the MG and embedding watermark injection mechanism in the MP.
- Protection scheme renewal: Achieved by enabling the MG replacement every trust interval expiration, which prevents hacking the MG and VS.
- Monitor: Achieved by enabling the NSP to monitor the VS at the EU every trust interval.
- Authorization service: Achieved by using MG idea.
- Secure storage: Achieved by using MG idea.
- Assets unforgeable proof: Achieved by the original MSA architecture.

Neither the User privacy assurance nor the Web spider is achieved with the PPMD model.

7.6 Summary

In previous chapters, we noted that current DRM solutions do not scale when the number of users who use this model increases above a certain level. We pointed out the basic requirements that make the protection persistent and used these requirements to improve both Grimen et al. model and secure multicast architecture. We proposed a scalable software-based protection that extends the period for securing the delivered content media. The proposed solution relies on utilizing the network service provider, and add more privileges to him/her, and make him able to do extra tasks such as monitoring the integrity of the end user's protection software and individualizing a piece of the protection software; this piece is called the mobile guard. The proposed solution based on utilizing the NSP. We showed how this solution will help to satisfy most of the DRM requirements. We validated the security aspects of our proposal. By merging the Gremin, et al. model and MSA, we acquire the persistent protection property that exists in Grimen et al. model and the scalability property that exists in the MSA model. The weakness of the new model is the extensive load added to the NSP because of cloning MG plug-ins and individualizing each one. The production of each MG may be different among different merchants, and this leads to more time consumption at customer side if s/he wants to switch between channels that happen to be managed by different merchants. This leads to uploading different MGs at the customer side.

To conclude, we proposed a new model that is flexible and persistently protected for delivered content media, and demonstrated that the new model is satisfying the persistent protection requirement to a certain level.

Chapter 8

The Persistent Protected and Scalable Delivery Model

In Chapter 7, we proposed a persistent protection in multicast content delivery model, which is a merger of Grimen, et al. model and multicast security architecture. The new model has five main roles: the content provider, the merchant, the content server, the network service provider and the end user. The idea of the merger is to acquire the persistent protection property that exists in Grimen et al. model and the scalability property that exists in MSA model. The weaknes of the new model is the extensive load added to NSP in term of cloning MG plug-ins, as well as individualizing each one. Add to this, these cloning and individualization processes need to be repeated for each different merchant's customers. In this chapter, we will introduce an improved and more scalable architecture for the persistent protection in multicast content delivery. The new architecture is based on the persistent protection in multicast content media delivery model discussed in Section 7.2, and it will lessen the new NSP extensive load that resulted from cloning and individualizing MG plug-ins for each merchant's customers. The production of each MG may be different for each merchant, which will lead to more time consumption at customer side if s/he wants to switch between

channels that happen to be managed by different merchants. This leads to uploading different MGs at the customer side. We will look for ways to reduce the number of MGs produced.

This material provides additional scalability in our system, which improves the achievement of our second goal.

8.1 Another Improvement for MSA Model

In order to have strong and fast deployment for security, rights enforcement and efficient execution of content media distribution transactions, we introduce the mobile security provider (MSP) role to the MSA architecture, and the main services provided by this new role are:

a Trust service.

b Rights enforcing.

c Protection scheme renewal.

The idea behind introducing mobile security provider (MSP) is to increase the separation of concern, as well as release the MR from some responsibilities, such as rights enforcing and protection scheme renewal, and thus provide the merchant with more flexibilty for utilising his/her resources.

We suggest that we define a distributed MSP role for each country or territory. That role is responsible for building a standard viewer software according to the specifications we mentioned in Section 7.2.7.

In order to deploy our proposal, we assume the following assumptions to be valid:

1. The CP trusts the MR.

2. The NSP trusts the MR.

3. Both the MR and NSP trust the MSP.

4. The CP is specialized for generating and encoding the content media.

5. The MR is specialized in selling content media

6. The MSP is specialized in generating the mobile protection (MP) and mobile guard (MG) that are responsible for encrypting and decrypting the content media.

7. The MP and MG are executable plug-in files.

The MSP is responsible for developing multimedia player software, which is composed of a static part (such as a code section and some portions of a data section) and a dynamic part (such as heap and stack sections). We assume that the static part of the multimedia player is the stamp of the player, and we can use it to represent an application class instance for a specific platform. The multimedia player is designed in a way that accepts a plug-in mobile code called "mobile guard" (MG) [GMM06b], which is a piece of executable code.

The player software works as an executable host that carries out decryption and decoding of the EEMD. The decryption executable code and the decryption key are embedded within the plug-in mobile guard. The main security goal of this scheme is to protect the cryptographic code fragment and key(s) from exposure or modification. The player software can be run on personal computers, PDAs, mobile phones and so on.

The MSP is also responsible for developing multimedia packager software with the same properties as the multimedia player software, namely static and dynamic parts. Again, the static part of the multimedia packager is the stamp, which can be used as a representative instance of the application class used within a specific platform. The

multimedia packager is designed in a way that accepts a plug-in mobile code called "mobile protection" (MP), which is a piece of executable code.

The packager software functions as an executable host that carries out encryption and encoding of the EEMD. The encryption executable code and the encryption key are embedded inside the plug-in mobile protection. The main security purpose of this scheme is to protect cryptographic code fragment and key(s) from being exposed or modified. The packager software should run on high computation servers.

The MSP generates the suitable list of software mobile protection (MP), which are needed to protect and package the content media at the CP side. S/He also generates the enforcing mechanisms for the terms and conditions and embeds them into the MG. The MSP may embed a mechanism that footprints individual instances of the EEMD.

The MSP divides an entire day into epochs, periods of time, and we will use the term "trust interval" for each epoch. The MSP generates one MP and one MG for each epoch; thus we have a limited number of MP and MG for each day.

The MP's and MG's structures contain multiple encryption or decryption methods, a hash function method and multiple secret keys and public keys, some of them are dummy and others are real. The main goal of having these elements is to confuse the hacker, and make his job harder. Figure 28 and Figure 29 depict the MP's and the MG's structures.

The MSP generates a list of MP(s), each of which is embedded with one media key that is used to encrypt one sub-media document. Every mobile protection is associated with one sub-media document. At the same time, the MSP generates a list of MG(s), each of which is embedded with one media decryption key that is used to decrypt one specific sub-media document. In other words, for each mobile protection, there is a corresponding mobile guard.

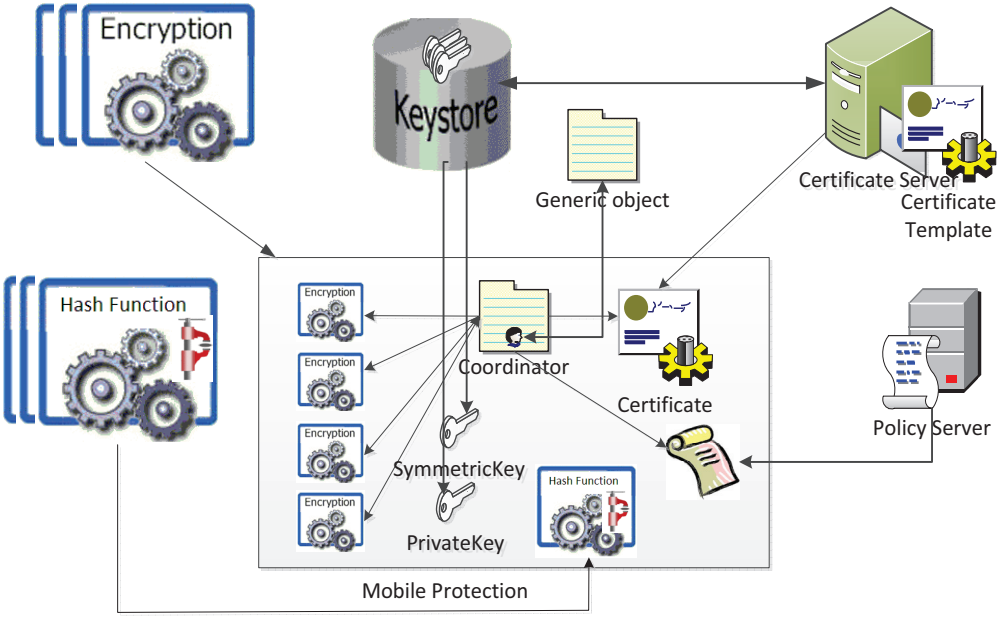

Figure 28: Mobile protection structure

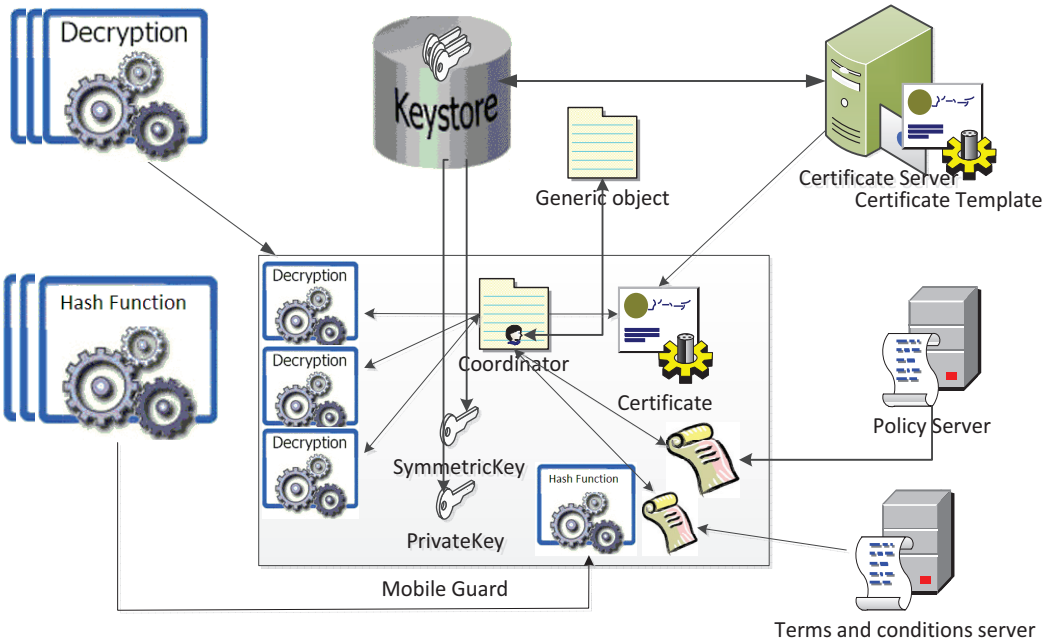

Figure 29: Mobile guard structure

When the CP generates the content media, s/he contacts the merchant and sends all required information about the content media to be registered. The MR contacts the MSP and informs him about the scheduled time for the EEMD. The MSP sends the MR only the related MP(s) that are valid for that scheduled time.

The MR receives requests from end users who are interested in watching one of the goods, and provide them tickets to use them later to show their eligibility for using the content media.

A sufficient amount of time before the new epoch starts, the MSP sends the NSP(s) an un-individualised MG related to that epoch. The epoch-related MG is capable of decrypting and enforcing access right conditions on the requested sub-EEMD for that individual epoch or trust interval. Figure 30 and Figure 31 illustrate the process of validating the CP's and EU's integrity. In this case, we will have one MG for each epoch, and that MG embeds related security associations used to decrypt each sub-EEMD that is supposed to flow at that time. All CP(s) who are supposed to protect a sub-EEMD, which happens to be distributed in the same epoch, need to protect the sub-EEMD using the corresponding MP. This will help to switch quickly between different EEMD that are supposed to be distributed during the same time interval. Figure 32 shows the persistent protected and scalable delivery architecture's work flow.

In the MSP there are many processes working for developing and implementing MP(s) and MG(s). There, you will find a library bank that contains code fragments, sub-routines and functions that are responsible to do MP and MG related tasks such as: encryption and decryption algorithms, mechansism to insert cryptosystem keys, mechanisms for enforcing terms and conditions, mechanisms for evaluating end user's environment-integrity, mechanisms for individualizing MG(s), mechanisms for footprinting content media. Figure 33 shows the mobile security provider's work flow.

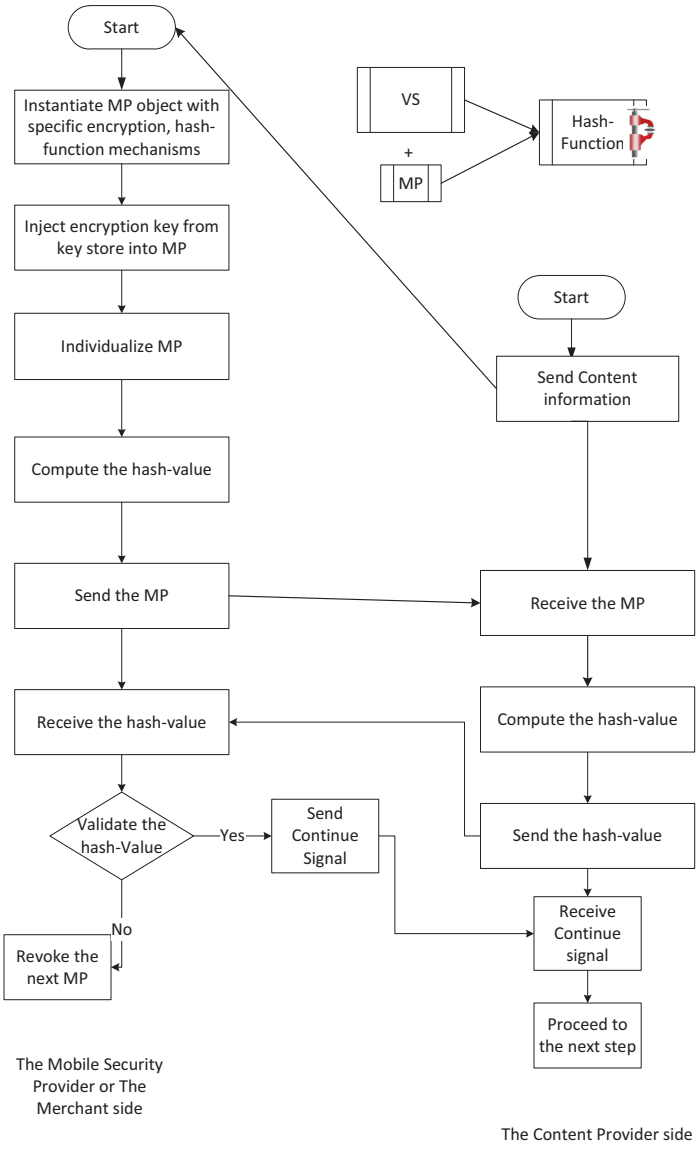

Figure 30: Validating the CP's integrity process

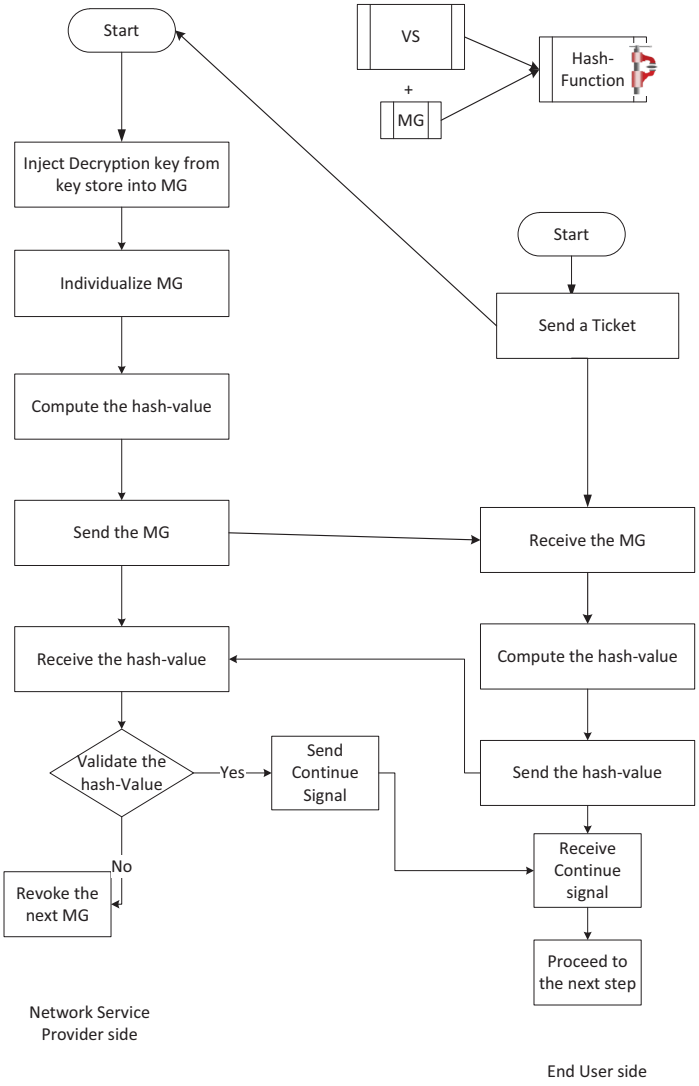

Figure 31: Validating the EU's integrity process

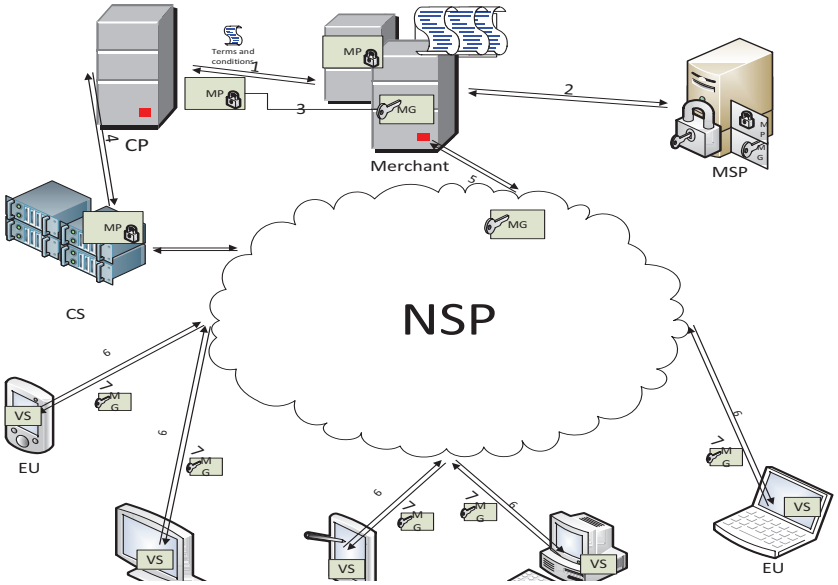

Figure 32: Persistent protected & scalable delivery work flow

As we assumed, there will be a contract signed by both the MR and the CP; that contract is a consent between both parties that their works must be legal. In this case no connction is established by the NSP and the CP or CS unless it has been approved by the MR. That will help if any illegal content media is distributed by an illegal CP, then it is the responsibility of the MR is to judge that bad CP which is easy to catch by the NSP.

Now we come to the most important part: when and where should the MG be individualised? We need to study this issue for two reasons: to trace back the source of illegal distribution and increase the scalability of our proposal to the maximum. To answer this question, we need to know more details about last mile delivery.

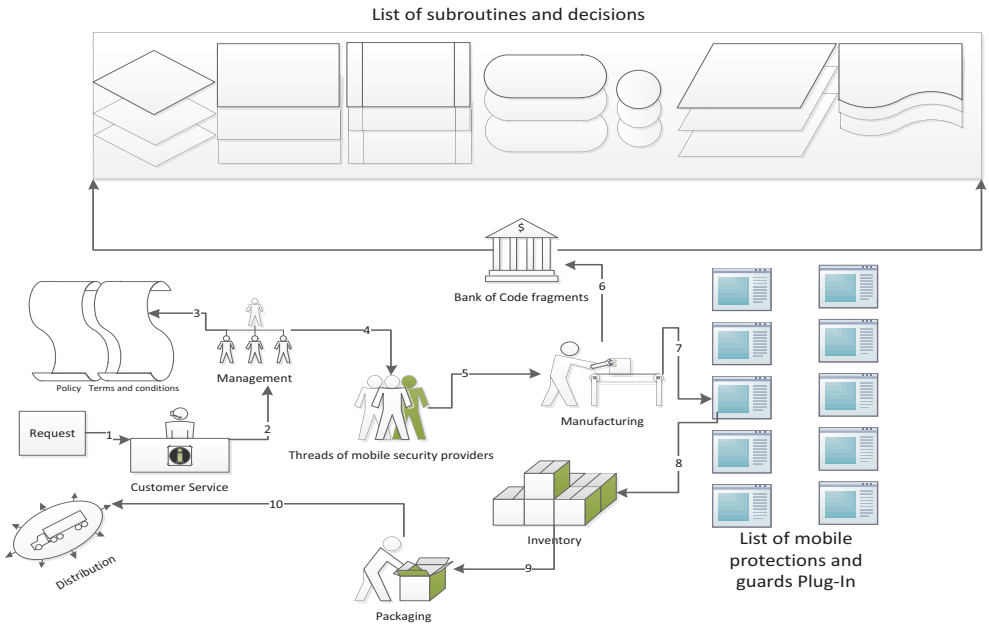

Figure 33: Mobile security provider's work flow

8.2 Last Mile Connection

Last mile connection, also called first mile connection, refers to the one-to-one connection between a network customer and the network service office. Whatever the intermediate connections that builds the network, the last mile connection is the leg that connects the end user to the public or private network, whether that connection is used for data uploading or downloading. As has been suggested in [Hel06], the content media distribution network is divided into three levels:

- National Hubs

  The active components, such as satellite, that are used to connect territories and countries form the national hubs. They are supposed to have a big store for submitted content media to be distributed within the country.

- Regional Hubs

  The active components that form the network backbone or core, where each country has a network core in order to connect the whole country's regions. The regional hub is the intermediate connection between the national hubs and the local hubs.

- Local Hubs

  The components that connect the customer to the network core.

8.2.1 Last Mile Connection Over Copper Based Connection

Asymmetric Digital Subscriber Line (ADSL) is a copper based connection between the end user and a network regional office. For downstream data rate of 8 Mbps and upstream data rate of 768 Kbps, the maximum distance of the copper based connection beween the subscriber's premises and the telephone branch office should not exceed 5.5 Km [Hel06].

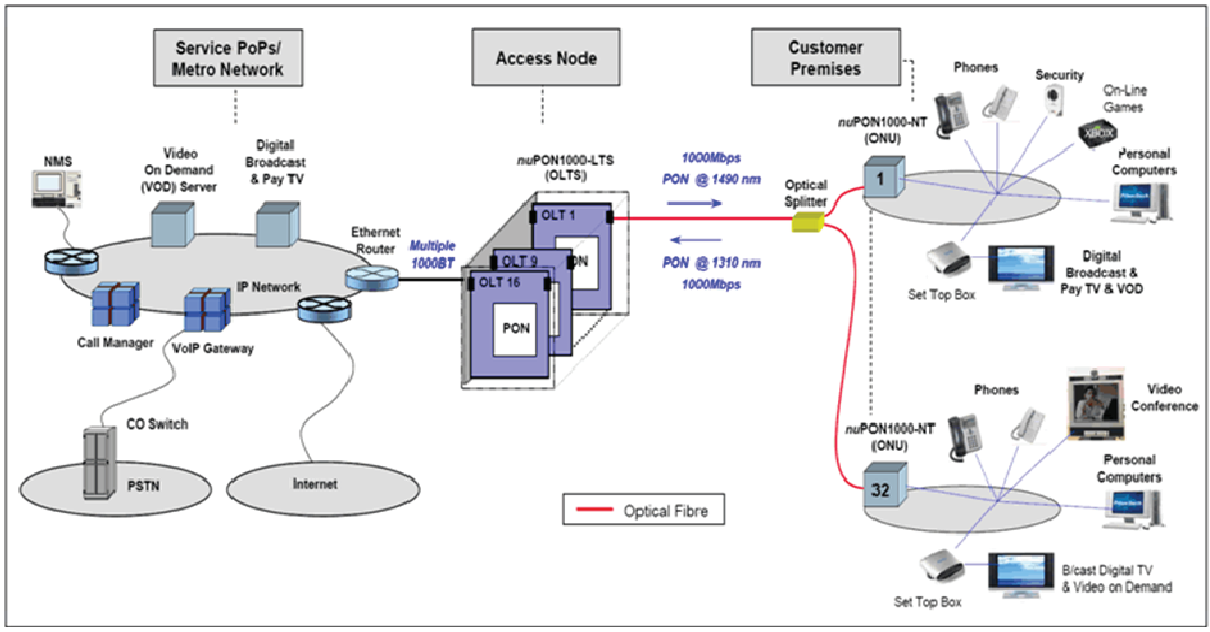

Figure 34: Physical network connection [nuP].

ADSL2+ is a twisted pair copper based wire that provides up 10 Mbps downstream bit rate transmission, and 3 Mbps upstream over maximum distance of 2 km from the DSLAM. Note that ADSL2+ may provide 26 Mbps over 300 meter of twisted pair copper based wire [FB06, Hel06].

The acceptable range of the cable length used in a very-high-bitrate DSL (VDSL) is up to 900 meters. At 300 meters the twisted pair copper based wire can carry up to 52 Mbps acceptable downstream bit rate transmission. The acceptable downstream transmission decreases to 19.2 Mbps when the cable length approaches 900 meters [FB06].

A connection between multiple ADSL, VDSL, or any kind of DSL signals, requires a Digital Subscriber Line Access Multiplexer (DSLAM) device. This device will allow for more functionality and control over the network [wik].

8.2.2 Last Mile Connection Over Optical Connections

Nowadays, due to the increasing demand for more bandwidth, Internet speed is increasing from Mbps to Gbps owing to the advances in fiber optic technology. Compared to copper-based connections, fiber optic lines provide a higher data transmission rate and a longer distance connection between the subscribers and the central office. Fijnvandraat and Bouwman [FB06] categorize last mile optical networks into two types depending on the active electronic devices that participate in the connection between two terminals:

- Optical network or optical Ethernet: In this category, there are routers, gateways or multiplexers between the end user's premise and the network branch office.
- Passive optical network: In this category, there are no routers, gateways or

multiplexers between the end users' premises and the network branch office; instead, it has been replaced by a single fiber optical connection.

Generally, last mile connections that use optical connections are divided into the following categories:

- The Optical Line Termination (OLT): The Optical Line Termination is a network interface that combines electrical data, which comes from multiple base 1000T Ethernet routers, gateways and multiplexers residing in the network core, to form a single optical-fiber. Usually, OLT exists at the companies' central offices. The OLT can support up to 512 end users [Hel06, O'D08, nuP, Opt].
- Fiber-to-the-home (FTTH): It is possible these days to connect the end user directly to the company central office via fiber cable called Fiber-to-the-home (FTTH) connection. This one-to-one connection provides 400 times faster data rate than ADSL connection. This type of network connection is useful for the content server in order to provide the network with high quality content media within fast speed.
- Passive Optical Network (PON): In this category, there are no routers, gateways or multiplexers between the end users' premises and the network branch office; instead, it has been replaced by a single fiber optical connection.

The transmission medium determines the maximum transmission rate, which is measured by bits per second (bps). Transmission media types are as follows [Hal96]:

- Twisted pair lines: it is subdivided into: unshielded twisted pair (UTP) and shielded twisted pair (STP). This type of connection supports data rates of 1Mbps with distances shorter than or equal 100 meters. It supports lower bit rate with longer distance than 100 meters, and support more bit rate with distance shorter than 100 meters.

- Coaxial cables: this type of connections can support 10 Mbps, and with distances lower than or equal to 500 meters. This type of connection suffers high maintenance cost.
- Optical fiber: this type supports 1 G bps with distances less than or equal to 20 Km.
- Satellites: this type composes Terrestrial microwave and radio transmissions. This kind of transmission usually does not support higher transmission rate.

From the previous list, we can notice that fiber optics has larger bandwidth and longer distance for signal to reach. That gives an indication for future bandwidth demand, it is better to install fiber to the home. The end user can be satisfied with ADSL and ADSL+ connections, but the CS, the CP, the MR and the MSP should use fiber optics connections.

8.3 The MP and MG Individualization Point

The CP is a network customer that needs to be connected to the NSP in order to multicast the content media via the CS that works for him. For efficiency purposes, it is recommended that the last mile connection to the CS be fiber optics. A sufficient amount of time before the new epoch starts, the CP needs to receive individualized version of MP that is responsible for fingerpring the sub-content media with a unique identifier of the CP, which is a proof for the legality of the content media distribution. This sufficient time needs to be short enough before the distribution starts for that sub-media, thus the CP or CS does not have the time to attack the MP instance. The individualization process is done by adding unique tokens that represent the CP into specific points of the MP, these points are determined by the MSP. The MSP informs the MR with the predetermined hash value of MP along with the protection software that runs at CP side, which is used to deploy the protection action.

The MP runs at the CP site and computes the integrity of the CP and then sends the result hash value to the MR. The MR checks the validity of the received hash value; in case of valid response, it tells the NSP to open the gate for incoming flow from the CS.

For the end user, s/he can use fiber or coper-based connection for the last mile connection. Every MG instance needs to be individualized by injecting the end user's ticket into the MG; then, the individualized MG must be unicast to the specific end user. In order to prevent the end user from searching for the ticket pattern within the MG, the ticket needs to be split into multiple pieces, which are injected into multiple places within the MG. These places are predicted by the NSP but not by the end user. This process prevents the user from modifying the MG for a certain amount of time, because, the MG need to be excuted on the end user's machine and then compute the hash value of the MG along with VS. That hash value needs to be sent to the NSP. Then, the NSP validates the hash value, which reflects the end user's integrity as has been described in the previous chapter.

To maximize scalability of distributing the MG, the unindividualized MG needs to be multicast until the last mile edge; at that point, the individualisation process should begin. Then, the individualized MG is unicast to the end user. This process assures that the scalability is not affected during the transport from multicast unindividualized MG(s) into unicast individualized MG(s).

For the individualization process we could do the following:

- Place a computer in the branch office, that computer is responsible for doing the individualization process.
- The optical line terminals, which are used for last mile connections, are hardware designed to do a particular job, and it may be easy to add individualization function.

- Redesign the curb side box and then put another hardware card in it. The card works as distributed controller to do the individualization proccess for the NSP.

We do not need to validate the exchanged protocols between the MSP, the NSP and the EU because it is similar to the protocol that happens between the MR, the NSP and the EU that we proposed and validated in the previous chapter.

We can allow every end user to have a domain access for the content media, in such a way that one ticket allows one user to watch the content media via N different machines or VS instances that are owned by the same user. We could allow that by permitting the user to receive N individualized MG on his N machines or VS instances, and then the NSP counts N hash-value that comes from the same user. Afterward, the NSP sends the suitable media keys for each machine or VS instance. This idea gives the end user the ability to watch N different channels that are supported by the NSP on N different machines or N different VS instance.

Within the trust interval, if the VS wants to change to different channels that are supported supported by the NSP and allowed to be viewed using the same ticket, then the NSP sent the new media key encrypted with the established secret key K, that is generated by the merchant and embedded in the ticket. Another possible scenario is that the NSP injects an asymmetric key pair in to the MG instance before sending it the specific VS. Then the NSP encrypts any new media key using the public key. The MG instance's public key is not known to anyone but the NSP. This property increases the security relation between the NSP and the VS, and then subsequently, the MG's private key may used for injecting a unique finger-print into received sub-media document, which can be used for tracking illegal media distribution.

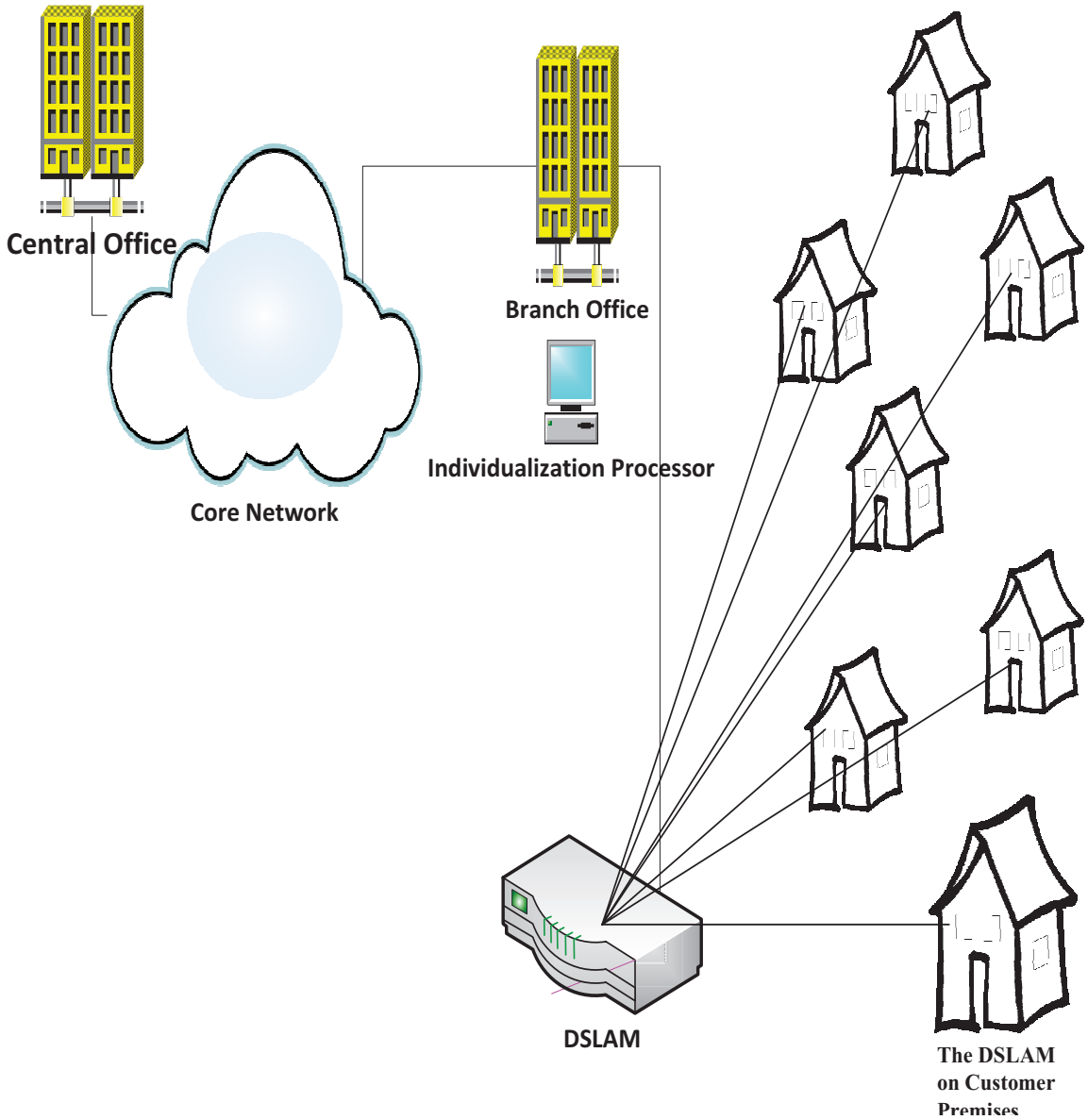

Figure 35: Individualization in branch level

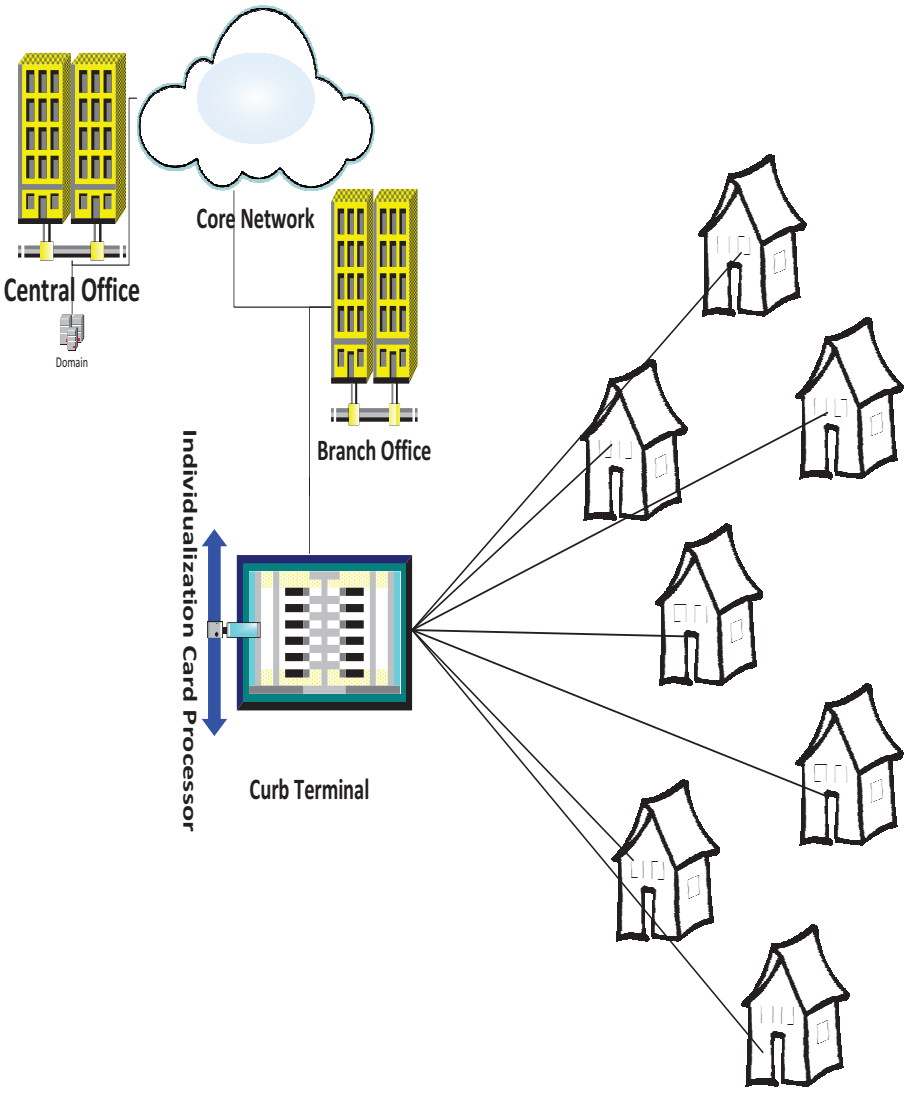

Figure 36: Individualization in curb level

8.4 Summary

In this chapter we minimized the number of MGs produced by adding a new role called mobile security provider (MSP), which is responsible for producing one version of MG and MP per trust interval. In this case the end user receives only one version of MG within an individual trust interval, and if the end user wants to switch between channels, he only needs to receive the decryption key for the new channel. Also we provided distributed cloning and individualization processes, which makes the resulting architecture more scalable.

To conclude, the architecture that we are proposing contains six roles, those roles collaborate together in order to provide a scalable and persistent protection in multicast content delivery (PPMCD). Those roles are: the CP, the CS, the MR, the MSP, the NSP and the EU. In this chapter, we proposed a solution to individualize each MG copy for each individual end user.

Chapter 9

Conclusion and Future Work

Internet distribution provides a flexible way to distribute intellectual property such as software, entertainment, etc. Cryptography technologies can be used to protect the intellectual property in transit in the network; thus it provides protection before delivery. Cryptography technologies alone cannot protect the intellectual property when it is hosted in the target's environment that is outside the control of the content owner; and thus it does not provide protection after delivery. Digital Rights Management is used to protect content media while it is in the end user's machine but it is subject to various attacks such as reverse engineering because it uses long-term protection mechanisms. Thus, the persistent protection that has been promised by DRM systems is not persistent.

In this thesis, we studied some DRM systems and learned through the study their strengths and weaknesses, and also the means used to achieve protection after delivery, which are divided into two directions: protection after delivery via hardware-based protection and protection after delivery via software approach. Because we are looking to a solution that can be quickly deployed into general personal computers and laptops, we preferred to go in the second direction. DRM technologies are primarily based on a unicast one-to-one distribution model, which is not scalable, and that

leads to a growing need for scalable and persistent protection delivery. Secure multicast can satisfy the scalability requirement, but it does not provide protection after content media delivery. Our target in this thesis is to design a flexible mechanism, architecture and protocols, for scalable, scheduled and persistently protected content media delivery.

In order to extend the content owner's control and prevent the customer from accessing the delivered content, we collected and presented eleven basic DRM requirements which are necessary in ideal unicast-based DRM systems, and that was the beginning road to push these requirements into multicast world. We realized that most of the unicast-based DRM are the same for multicast-based DRM, however the implementation and architecture are different. The DRM requirements covers five aspects: access control, security, privacy, robustness and marketing. The comparison we made of different DRM systems with respect to the eleven requirements, see the table in Figure 15, gives us a clear picture that existing DRM systems do not satisfy all these requirements.

Atwood's multicast security architecture has some of the eleven requirements, but it does not deal with DRM or persistent protection.

We analyzed the eleven requirements for basic DRM system, and found that the "on demand" requirement conflicts with the "scalability" requirement. We suggested a trade-off solution would be on dropping the "on demand" requirement and enhancing the "scalability" requirement.

We recognized that Grimen et al.'s model is a potential solution for the persistent protection delivery model. Furthermore, we discovered a flaw in their model, because there is no possibility to distinguish between "mobile guard" copies. We fixed the flaw by individualizing each copy of the mobile guard, and then based on Grimen et al.'s model, we proposed an improved model. Finally, we validated both protocols. However, the original and improved Geimen et al.'s models do not scale for a large

number of users.

The improved Grimen et al.'s model provides persistent protection for the delivered content media, but it is not scalable. Multicast secure architecture model provides a scalable delivery system but the delivered content media is not persistently protected. It is clear both models have advantages and disadvantages. At this stage it is clear that we can reach the final goal if we marry the improved Grimen, et al.'s model with the multicast secure architecture model.

We proposed a protocol and an architecture for the result of merging both multicast secure architecture and improved Grimen et al.'s model, and then validated the protocol. We showed that the new model achieves scalable and persistent protection for delivered content media; and then, we proposed an architecture for a more distributed, scalable and persistent protection of the delivered content media, and there we adapted the architecture so it provides a scalable individualization technology for the mobile guard and mobile protection plug-in code.

In this thesis, we used software-based protection and the idea of renewability of one component of the software-based protection, i.e., mobile guard. We used a multicast distribution mechanism to distribute both mobile guards and media documents. Our novelty is to show how to merge the ideas of using mobile guard, renewability and multicast in such a way that the resulting solution does not need expensive hardware to be deployed, and is scalable, cost effective, and exhibits persistent protection for delivered media.

In this work, we did not address the privacy requirement, so for future work, we will concentrate on the end user privacy protection.

Bibliography

[16w] Windows media digital rights management. http://msdn2.Microsoft.com/en-us/windowsmedia/bb190317.aspx, Accessed March 31, 2008.

[AH04] Alapan Arnab and Andrew Hutchison. Digital rights management - an overview of current challenges and solutions. 7 2004. http://pubs.cs.uct.ac.za/archive/00000139/, Accessed January 29, 2011.

[AH05] Alapan Arnab and Andrew Hutchison. Requirement analysis of enterprise drm systems. In *Proceedings of Information Security South Africa (ISSA) Conference 2005, Johannesburg, South Africa*, 2005.

[AH07] Alapan Arnab and Andrew Hutchison. Persistent access control: A formal model for drm, October 2007. http://pubs.cs.uct.ac.za/archive/00000411/03/acmdrm07-arnab.pdf, Accessed May 08, 2011.

[All] Open Mobile Alliance.

[All06] Open Mobile Alliance. Drm architecture - oma-ad-drm-v2_0-20060303-a -approved version 2.0 03 mar 2006, 3 2006. http://www.omadrm.ru/spec/version2/DRM Architecture.pdf, Accessed June 29, 2011.

[Ans] Answers.com. http://www.answers.com/topic/persistent, Accessed June 15, 2010.

[Arn07] Alapan Arnab. Towards a general framework for digital rights management (drm). *PhD, Department of Computer Science, University of Cape Town, 2007*, 2007. http://pubs.cs.uct.ac.za/archive/00000448/01/alapan.arnab.thesis.final.pdf, Accessed May 08, 2011.

[Ars] Emilija Arsenova. Technical aspects of digital rights management. http://wob.iai.uni-bonn.de/Wob/images/01212504.pdf, Accessed January 29, 2011.

[BA] Malek Barhoush and J. William Atwood. Requirements for enforcing digital rights management inmulticast content distribution. *Telecommunication Systems*.

[BA10] Malek Barhoush and J. William Atwood. Software-based protection for content media distribution. In *Proceedings of the IADIS International Conference on WWW/Internet (ICWI2010)*, October 2010.

[BE06] Hagai Bar-El. Challenges in designing content protection solutions. 2006. www.hbarel.com/publications/Challenges_in_designing_content_protection_solutions.pdf, Accessed May 09, 2011.

[BEM+07] Dagmar Bruss, Gábor Erdélyi, Tim Meyer, Tobias Riege, and Jörg Rothe. Quantum cryptography: A survey. *ACM Comput. Surv.*, 39, July 2007.

[BMJ07] Koen Buyens, Sam Michiels, and Wouter Joosen. A software architecture to facilitate the creation of drm systems. In *Consumer Communications and Networking Conference, 2007. CCNC 2007. 4th IEEE*, pages 955 –959, jan. 2007.

[BP82] H.J. Beker and F.C. Piper. Communications security: a survey of cryptography. *Physical Science, Measurement and Instrumentation, Management and Education - Reviews, IEE Proceedings A*, 129(6):357 –376, 6 1982.

[Cai02] Deering S. Kouvelas I. Fenner B. Thyagarajan A. Cain, B. *Internet Group Management Protocol, Version 3*. RFC3376, IETF, October 2002.

[CC03] Kin-Ching Chan and S.-H.G. Chan. Key management approaches to offer data confidentiality for secure multicast. *Network, IEEE*, 17(5):30 – 39, sept.-oct. 2003.

[Cen] Pocket PC Central. Windows mobile 6 device types - smartphones & pdas. http://pocketpccentral.net/smartphone/help/general/wm6_naming_scheme.htm, Accessed 30 Jan 2011.

[Che08] Chin-Ling Chen. A secure and traceable e-drm system based on mobile device. *Expert Systems with Applications*, 35:878–886, 10 2008.

[Cle] Jan De Clercq. Smart cards. http://technet.microsoft.com/en-us/library/dd277362.aspx, Accessed July 17, 2011.

[Cor04a] Microsoft Corporation. Architecture of windows media rights manager, 5 2004. http://www.microsoft.com/windows/windowsmedia/howto/articles/drmarchitecture.aspx, Accessed 30 Jan 2011.

[Cor04b] Microsoft Corporation. Microsoft announces new version of windows media digital rights management software, 5 2004. http://www.microsoft.com/presspass/press/2004/may04/05-03digitalrightsmanagementtechnologypr.mspx, Accessed 30 Jan 2011.

[Cor08] OMA Corporation. Drm content format. Aug 2008. http://www.openmobilealliance.org/Technical/release_program/drm_archive.aspx, Candidate Version 2.1 05 Aug 2008. Accessed January 29, 2011.

[Coy03] Karen Coyle. The technology of rights: Digital rights management. Nov 2003. http://www.kcoyle.net/drm_basics.pdf, Accessed January 29, 2011.

[CS05] Yacine Challal and Hamida Seba. Group key management protocols: A novel taxonomy. *International Journal of Information Technology*, 2(1):105–118, 2005.

[CT02] Christian S. Collberg and Clark Thomborson. Watermarking, tamper-proofing, and obfuscation - tools for software protection. *Software Engineering, IEEE Transactions on*, 28:735–746, 2002.

[DD00] Stéphane Doyon and Mourad Debbabi. Verifying object initialization in the java bytecode language. In *Proceedings of the 2000 ACM symposium on Applied computing - Volume 2*, SAC '00, pages 821–830, New York, NY, USA, 2000. ACM.

[Doc05] Cory Doctorow. Digital rights management: A failure in the developed world, a danger to the developing world. Technical report, Electronic Frontier Foundation, 2005. http://w2.eff.org/IP/DRM/ITU_DRM_paper.pdf, Accessed March 25, 2010.

[Doe07] Stefan Doehla. Dvb-h handheld video content protection with isma encryption, 5 2007. http://www.design-reuse.com/articles/14971/dvb-h-handheld-video-content-protection-with-isma-encryption.html, Accessed May 28, 2011.

[DY81] D. Dolev and A. C. Yao. On the security of public key protocols. In *Foundations of Computer Science, 1981. SFCS '81. 22nd Annual Symposium on*, pages 350 –357, oct. 1981.

[EDB04] Martin R. Stytz Eric D. Bryant, Mikhail J. Atallah. A survey of anti-tamper technologies. *CrossTalk: The Journal of Defense Software Engineering*, 17(11):12–16, 11 2004.

[FB06] Marieke Fijnvandraat and Harry Bouwman. Flexibility and broadband evolution. *Telecommunications Policy*, 30(8-9):424 – 444, 2006.

[FH06] E.W. Felten and J.A. Halderman. Digital rights management, spyware, and security. *Security Privacy, IEEE*, 4(1):18 – 23, jan.-feb. 2006.

[GA98] Stefanos Gritzalis and George Aggelis. Security issues surrounding programming languages for mobile code: Java vs. safe-tcl. *SIGOPS Oper. Syst. Rev.*, 32:16–32, April 1998.

[GMM06a] Gisle Grimen, Christian Mönch, and Roger Midtstraum. Building secure software-based drm systems, 11 2006. http://www.nik.no/2006/Grimen.pdf, Accessed March 26, 2010.

[GMM06b] Gisle Grimen, Christian Mönch, and Roger Midtstraum. Software-based copy protection for temporal media during dissemination and playback. In *Information Security and Cryptology - ICISC 2005*, volume 3935/2006, 2006.

[GMM06c] Gisle Grimen, Christian Mönch, and Roger Midtstraum. Tamper protection of online clients through random checksum algorithms. In *ISTA*, pages 67–79, 2006.

[Gro07] Trusted Computing Group. Tcg specification architecture overview - specification revision 1.4 2nd august 2007, 8 2007.

http://www.trustedcomputinggroup.org/files/resource_files/AC652DE1-1D09-3519-ADA026A0C05CFAC2/TCG_1_4_Architecture_Overview.pdf, Accessed June 29, 2011.

[Hal96] Fred Halsall. *Data Communications, Computer Networks, and Open Systems*. Addison-Wesley, 4 edition, 1996.

[HB01] Mohamed Hefeeda and Bharat Bhargava. On mobile code security. Technical report, 2001. http://www.cs.sfu.ca/ mhefeeda/Papers/OnMobileCodeSecurity.pdf, Accessed August 8, 2011.

[HB05a] Greg Hoglund and James Butler. *Rootkits: Subverting the Windows Kernel*. Addison-Wesley, 2005.

[HB05b] Grog Hoglund and James Butler. Rootkits: Subverting the windows kernel. page 352, 2005.

[Hel06] Gilbert Held. *Understanding IPTV*. Prentice Hall, 2006.

[Hou04] R. Housley. *Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)*. RFC3686, IETF, January 2004.

[HP07] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. 2007.

[HS06] M. Hussain and D. Seret. A comparative study of security protocols validation tools: Hermes vs. avispa. In *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, volume 1, pages 6–308, feb. 2006.

[Hua07] Tiejun Huang. Evolvement of drm schema: From encryption to interoperability and monitoring. In Nicu Sebe, Yuncai Liu, Yueting Zhuang,

and Thomas Huang, editors, *Multimedia Content Analysis and Mining*, volume 4577 of *Lecture Notes in Computer Science*, pages 65–75. Springer Berlin / Heidelberg, 2007.

[IA06] Salekul Islam and J. William Atwood. A framework to add aaa functionalities in ip multicast. In *Proceedings of Advanced International Conference on Telecommunications (AICT'06)*, Guadeloupe, French Caribbean, February 2006. IEEE Computer Society.

[IA07a] Salekul Islam and J. William Atwood. A policy framework for multicast group control. In *Consumer Communications and Networking Conference, 2007. CCNC 2007. 4th IEEE*, pages 1103 –1107, jan. 2007.

[IA07b] Salekul Islam and J. William Atwood. Sender access control in ip multicast. *Local Computer Networks, Annual IEEE Conference on*, 0:79–86, 2007.

[IA09] Salekul Islam and J. William Atwood. Multicast receiver access control by igmp-ac. *Comput. Netw.*, 53:989–1013, May 2009.

[Ian01] Renato Iannella. Digital rights management (drm) architectures, 2001. http://www.dlib.org/dlib/june01/iannella/06iannella.html, Accessed May 08, 2011.

[Int] InterTrust. The darknet and the future of content distribution. http://www.intertrust.com/main/overview/drm.html, Accessed 12 Oct 2008.

[Irw04] James Irwin. Digital rights management: The Open Mobile Alliance DRM specifications. *Inf. Secur. Tech. Rep.*, 9:22–31, December 2004.

[ISM06] ISMA. Internet streaming media alliance implementation specification - isma encryption and authentication, version 1.1, 2006.

http://www.mpegif.org/m4if/bod/ISMA/ISMA_E&Aspec1.1.pdf, Accessed Aug. 8 2011.

[J. 07] J. William Atwood. An architecture for secure and accountable multicasting. In *Local Computer Networks, 2007. LCN 2007. 32nd IEEE Conference on*, pages 73–78, 10 2007.

[JM07] Hugo Jonker and Sjouke Mauw. Core security requirements of drm systems, 2007. http://alexandria.tue.nl/extra1/wskrap/publichtml/200524.pdf, Accessed May 08, 2011.

[JPKJ06] Kun-Won Jang, Chan-Kil Park, Jung-Jae Kim, and Moon-Seog Jun. Manuscript received august 2005. a study on drm system for on/off line key authentication, 2006.

[Kam02] Mayur Kamat. Security requirement for digital rights management., 2002. http://proc.isecon.org/2002/353b/ISECON.2002.Kamat.pdf, Accessed May 08, 2011.

[KC04] William Ku and Chi-Hung Chi. Survey on the technological aspects of digital rights management. In Kan Zhang and Yuliang Zheng, editors, *Information Security*, volume 3225 of *Lecture Notes in Computer Science*, pages 391–403. Springer Berlin / Heidelberg, 2004.

[Kos98] Dave. Kosiur. *IP Multicasting : The Complete Guide to Interactive Corporate Networks*. New York : Wiley, c1998., 1998.

[LD03] Cullen Linn and Saumya Debray. Obfuscation of executable code to improve resistance to static disassembly. In *Proceedings of the 10th ACM conference on Computer and communications security*, CCS '03, pages 290–299, New York, NY, USA, 2003. ACM.

[LELD05] E.I. Lin, A.M. Eskicioglu, R.L. Lagendijk, and E.J. Delp. Advances in digital video content protection. *Proceedings of the IEEE*, 93(1):171 –183, jan. 2005.

[LG99] Charlie Lai and Li Gong. User authentication and authorization in the java(tm) platform. In *In ACSAC 99: Proceedings of the 15th Annual Computer Security Applications Conference*, page 285. IEEE Computer Society, 1999.

[Lou00] Panagiotis Louridas. Some guidelines for non-repudiation protocols. *SIGCOMM Comput. Commun. Rev.*, 30:29–38, October 2000.

[Low98] Douglas Low. Protecting java code via code obfuscation. *Crossroads*, 4:21–23, April 1998.

[LSNS03] Qiong Liu, Reihaneh Safavi-Naini, and Nicholas Paul Sheppard. Digital rights management for content distribution. In *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21*, ACSW Frontiers '03, pages 49–58, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[Ltd02] Sonera Plaza Ltd. Digital rights managament. white paper. Feb 2002. http://www.medialab.sonera.fi/workspace/DRMWhitePaper.pdf, DRM White Paper. Accessed January 29, 2011.

[MA07] Ritesh Mukherjee and J. William Atwood. Scalable solutions for secure group communications. *Comput. Netw.*, 51:3525–3548, August 2007.

[Mar] Mark Stamp. Stamp: Digital Rights Management: The Technology Behind The Hype . http://www.csulb.edu/web/journals/jecr/issues/20033/paper3.pdf, Accessed January 15, 2011.

[MD03] Thomas S. Messerges and Ezzat A. Dabbish. Digital rights management in a 3g mobile phone and beyond. In *Proceedings of the 3rd ACM workshop on Digital rights management*, DRM '03, pages 27–38, New York, NY, USA, 2003. ACM.

[Med03] Sonera MediaLab. Mobile digital rights managment. 8 2003. http://www.medialab.sonera.fi/workspace/MobileDRMWhitePaper.pdf, Accessed July 19, 2011.

[Mic] Microsoft. Windows media rights manager providers. http://www.Microsoft.com/windows/windowsmedia/forpros/drm/upgrade.aspx, Accessed March 31, 2008.

[Mit97] Suvo Mittra. Iolus: a framework for scalable secure multicasting. *SIGCOMM Comput. Commun. Rev.*, 27:277–288, October 1997.

[MRB01] Shafay Shamail Muhammad Razeen, Javaid Iqbal Zahid and Haroon Atique Babri. Development of digital certification authority in Pakistan. In *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, pages 239 – 245, 2001.

[MST05] Juergen Tacken Frank Bormann Miguel Soriano, Stephan Flake and Joan Tomás. Mobile digital rights management: Security requirements and copy detection mechanisms. In *Database and Expert Systems Applications, 2005. Proceedings. Sixteenth International Workshop on*, pages 251 –256, aug. 2005.

[MT08] Security Newsletter Microsoft TechNet. "deploying active directory rights management services at microsoft, technical white paper, 6 2008. http://technet.microsoft.com/en-us/library/ee156482.aspx, Accessed 25 May 2011.

[MVJDD05] Sam Michiels, Kristof Verslype, Wouter Joosen, and Bart De Decker. Towards a software architecture for drm. In *DRM '05: Proceedings of the 5th ACM workshop on Digital rights management*, pages 65–74, New York, NY, USA, 2005. ACM.

[NN07] Maria Nickolova and Eugene Nickolov. Conceptual model and security requirements for drm techniques used for e-learning objects protection. *International Journal "Information Technologies and Knowledge"*, Vol.1, 2007. http://sci-gems.math.bas.bg:8080/jspui/bitstream/10525/106/1/ijitk01-1-p18.pdf, Accessed Aug 08, 2011.

[nuP] nupon. http://www.ceos.com.au/products/nupon-content.htm, Accessed June 12, 2011.

[O'D08] Grard O'Driscoll. *Next Generation IPTV Services and Technologies*. Prentice Hall, 2008.

[OLR+07] Jose Onieva, Javier Lopez, Rodrigo Roman, Jianying Zhou, and Stefanos Gritzalis. Integration of non-repudiation services in mobile drm scenarios. *Telecommunication Systems*, 35:161–176, 2007. 10.1007/s11235-007-9050-4.

[OMA08a] OMA. DRM architecture. Technical report, 10 2008.

[OMA08b] OMA. DRM architecture oma. Aug 2008. http://www.openmobilealliance.org/Technical/release_program/drm_archive.aspx, Candidate Version 2.1 05 Aug 2008. Accessed January 29, 2011.

[Opt] Optical line termination unit oltu. http://www.selex-comms.com/internet/localization/IPC/media/docs/OLTU_EN_LR.pdf, Accessed June 12, 2011.

[PA11] Mohammad Parham and J. William Atwood. Validation of security for participant control exchanges in multicast content distribution. In *Proceedings of the Ninth Annual Conference on Privacy, Security and Trust (PST 2011)*, pages 73–78, Montreal, Quebec, Canada, 7 2011.

[Pat08] Viral Patel. Java virtual machine, an inside story!!, 12 2008. http://viralpatel.net/blogs/2008/12/java-virtual-machine-an-inside-story.html, Accessed April 27, 2011.

[PBW] Marcus Peinado Peter Biddle, Paul England and Bryan Willman. Digital rights management. http://searchcio.techtarget.com/sDefinition/0,,sid182_gci493373,00.html, Accessed January 29, 2011.

[PBW02] Marcus Peinado Peter Biddle, Paul England and Bryan Willman. The darknet and the future of content distribution. Nov 2002. http://msl1.mit.edu/ESD10/docs/darknet5.pdf, Accessed January 29, 2011.

[PJK06] Sang-Ho Park, Jaewoon Jeong, and Taekyoung Kwon. Contents distribution system based on mpeg-4 ismacryp in ip set-top box environments. *Consumer Electronics, IEEE Transactions on*, 52(2):660 – 668, May 2006.

[RB97] S. Berson S. Herzog S. Jamin R. Braden, L. Zhang. *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification*. RFC2205, IETF, September 1997. https://datatracker.ietf.org/doc/rfc2205/, Accessed January 29, 2011.

[RLMR00] Sergio Loureiro Refik, Sergio Loureiro, Refik Molva, and Yves Roudier. Mobile code security. In *In proceedings of ISYPAR 2000 (4ème Ecole dÍnformatiquedes Systèmes Parallèles et Répartis), Code*, 2000.

[Ros05] Bill Rosenblatt. Enterprise drm – technology comparison: Authentica active rights management and microsoft windows rights management services. Technical report, 2005.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21:120–126, February 1978.

[SA05] N. Sultana and J.W. Atwood. Secure multicast communication: end user identification and accounting. In *Electrical and Computer Engineering, 2005. Canadian Conference on*, pages 1691–1694, May 2005.

[San] Sandbox. http://www.webopedia.com/TERM/S/sandbox.html, Accessed May 3, 2011.

[Sav02] John Savard. A cryptographic compendium, 2002.

[SCA] Smart Card Alliance SCA. http://www.smartcardalliance.org/, Accessed June 21, 2010.

[Scr01] Beale Screamer. Microsoft's digital rights management scheme - technical details, 2001. http://cryptome.org/ms-drm.htm, Accessed May 9, 2011.

[(SD05] Sun Developer Network (SDN). Java security overview, white paper, 4 2005. http://java.sun.com/developer/technicalArticles/Security/whitepaper/JS_White_Pap
Accessed May 3, 2011.

[Smaa] Smart card. http://en.wikipedia.org/wiki/Smart_card, Accessed July 17, 2011.

[Smab] Smart cards. http://ewh.ieee.org/r10/bombay/news5/SmartCards.htm, Accessed July 17, 2011.

[SSK04] G. Selimis, N. Sklavos, and O. Koufopavlou. Crypto processor for contactless smart cards. In *Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean*, volume 2, pages 803 – 806 Vol.2, may 2004.

[Sta03] William Stallings. *Cryptography and Network Security Principles and Practices*. Prentice Hall, 3 edition, 2003.

[SV01] William Shapiro and Radek Vingralek. How to manage persistent state in drm systems. In *Digital Rights Management Workshop*, pages 176–191. Springer-Verlag GmbH, 2001.

[Sys] Azuki Systems. Drm in the new world of mobile media. http://www.azukisystems.com/contentOwners/DRMBrief_AzukiSystems.pdf, Accessed January 22, 2011.

[Tea] The AVISPA Team. Deliverable d2.1: The high level protocol specification language.

[Tea06a] AVISPA Team. Hlpsl tutorial: A beginners guide to modelling and analysing internet security protocols. Technical report, 6 2006.

[Tea06b] The AVISPA Team. Avispa v1.1 user manual, 2006.

[Tec] Techterms. Drm (digital rights management). http://www.techterms.com/definition/drm, Accessed January 29, 2011.

[TH] Haykal Tej and Paul Hankes. Challenge/response authentication protocol, version 2. http://www.avispa-project.org/, Accessed May 9, 2011.

[THVV10] Satou H. Ohta T. Hardjono Verisign, H. He and S Vaidya. *Requirements for Multicast AAA coordinated between Content Provider(s) and Network Service Provider(s).* draft-ietf-mboned-maccnt-req-06, IETF

Internet Draft, 10 2010. http://tools.ietf.org/html/draft-ietf-mboned-maccnt-req-10, Accessed May 9, 2011.

[TS02] Andrew S. Tanenbaum and Maarten Van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.

[VAD99] ATUL VADERA. Scamp : Scalable multicast protocol for communication in large groups. Master thesis, Indian Institute of Technology - Department of Computer Science & Engineering, 1 1999. http://www.cse.iitk.ac.in/users/dheeraj/mtech/atul-vadera.pdf, Accessed May 13, 2011.

[VC04] T. Hardjono Verisign and B. Weis Cisco. *The Multicast Group Security Architecture*. RFC3740, IETF, 3 2004. http://www.faqs.org/rfcs/rfc3740.html, Accessed May 9, 2011.

[VMF99] Kaushik Veeraraghavan, Andrew Myrick, and Jason Flinn. Cobalt: Separating content distribution from authorization in distributed file systems. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies. USENIX Association*, pages 231–244, 1999.

[wik] wikipedia. Digital subscriber line access multiplexer. http://en.wikipedia.org/wiki/Digital_Subscriber_Line_Access_Multiplexer, Accessed June 12, 2011.

[WMSL00] Chung Kei Wong, Wong Mohamed, Gouda Simon, and S. Lam. Secure group communications using key graphs. *Networking, IEEE/ACM Transactions on*, 8(1):16 –30, feb 2000.

[YKM+06] Hiroki Yamauchi, Yuichiro Kanzaki, Akito Monden, Masahide Nakamura, and Ken-ichi Matsumoto. Software obfuscation from crackers' viewpoint. In *Proceedings of the 2nd IASTED international conference on Advances in computer science and technology*, pages 286–291, Anaheim, CA, USA, 2006. ACTA Press.

[ZZM+06] Jun Zhang, Yu Zhou, Fanyuan Ma, Gu Dawu, and Yingcai Bai. An extension of secure group communication using key graph. *Inf. Sci.*, pages 3060 – 3078, 2006.

Appendix A

Original Grimen et al. Protocol Validation

In this section we will show the code we wrote to validate the Grimen proposed protocol. The following lines are the code used to validate the key exchange protocol proposed by Grimen et al. that takes place between the mobile guard and the security server:

```
role mobileGuard (M, S: agent,
            Ks,Km: public_key,
            Tki,Mki: symmetric_key,
            Hash: hash_func,
            SND, RCV: channel (dy))
played_by M

def=
```

```
  local State : nat,
        Nm, Ns: text,
        ChkSum : message

  init State := 0

  transition

    0.  State  = 0 /\ RCV(start) =|>
State':= 2 /\ Nm' := new()/\ Tki':= new()
   /\ ChkSum' := Hash(M.Nm'.Tki')
           /\ SND({Nm'.Tki'.M}_Ks.ChkSum')
 /\ secret(Tki', secTK,{M,S})
               /\ witness(M,S,mobileGuard_server_nm,Nm')
 /\ witness(M,S,mobileGuard_server_tki,Tki')

    2.  State  = 2 /\ RCV({Nm.Ns'.S}_Km) =|>
State':= 4 /\ SND({Ns'.M}_Ks)



    4.  State  = 4 /\ RCV({Mki'.S}_Tki) =|>
State' := 6 /\ secret(Mki', secMK,{M,S})
/\ request(M,S,server_MobileGuard_ns,Ns)
end role
%%
role server(M, S: agent,
         Km, Ks: public_key,
         Tki,Mki: symmetric_key,
         Hash: hash_func,
```

```
         SND, RCV: channel (dy))
played_by S def=

  local State : nat,
Nm, Ns: text

  init State := 1

  transition

    1.  State  = 1 /\ RCV({Nm'.Tki'.M}_Ks.Hash(M.Nm'.Tki')) =|>
State':= 3 /\ Ns' := new() /\ SND({Nm'.Ns'.S}_Km)
 /\ witness(S,M,server_mobileGuard_ns,Ns')

    3.  State  = 3 /\ RCV({Ns.M}_Ks)
        =|>
State':= 5 /\ Mki' := new()
           /\SND({Mki'.S}_Tki)
                 /\ secret(Mki', secMK,{M,S})
   /\ request(S,M,mobileGuard_server_tki,Tki)
   /\ request(S,M,mobileGuard_server_nm,Nm)



end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role session(M, S: agent, Km, Ks: public_key, Tki,Mki: symmetric_key,
             Hash: hash_func)
 def=
```

```
  local SA, RA, SB, RB: channel (dy)

  composition

mobileGuard(M,S,Ks,Km,Tki,Mki,Hash,SA,RA)
     /\ server  (M,S,Km,Ks,Tki,Mki,Hash,SB,RB)

end role
%%
role environment() def=

    const m, s         : agent,
  km, ks, ki   : public_key,
  tk, mk, tki,mki: symmetric_key,
  secMK, secTK, mobileGuard_server_nm,
  mobileGuard_server_tki, server_mobileGuard_ns   : protocol_id,
  h : hash_func
 % mobileGuard_server_ns,



    intruder_knowledge = {m, s,ks, h, ki, inv(ki),inv(km)}%,tki,mki,km }

    composition

session(m,s,km,ks,tk,mk,h)
           /\ session(m,i,km,ki,tk,mki,h)
/\ session(i,s,ki,ks,tki,mk,h)
```

```
end role
%%
goal

  secrecy_of secMK, secTK
 % authentication_on mobileGuard_server_ns
 % authentication_on server_mobileGuard_nm

end goal
%%
environment()
```

Appendix B

Improved Version of Grimen et al. Protocol Validation

In this section we will show the code we wrote to validate the proposed solution for Grimen et al. protocol. The following lines are the code used to validate the key exchange protocol proposed by us to solve the problem found at Grimen et al. proposed key exchange protocol that takes place between the mobile guard and the security server:

```
role mobileGuard (M, S: agent,
            Ks,Km: public_key,
            K: symmetric_key,
            Ticket: message,
            Hash: hash_func,
            SND, RCV: channel (dy))
played_by M
```

```
def=

  local State : nat,
        Nm, Ns: text,
        ChkSum : message,
        Tki,Mki: symmetric_key

  init State := 0

  transition

    0.  State  = 0 /\ RCV(start) =|>
State':= 2 /\ Nm' := new()/\ Tki':= new()
   /\ ChkSum' := Hash(M.Nm'.Tki'.Ticket)
           /\ SND({{Nm'.Tki'.M.Ticket}_Ks}_K.ChkSum')
   /\ secret(Tki', secTK,{M,S})
           /\ request(M,S,mobileGuard_server_nm,Nm')
  % /\ witness(M,S,server_mobileGuard_tki,Tki')

    2.  State  = 2 /\ RCV({Nm.Ns'.S}_Km) =|>
State':= 4 /\ SND({Ns'.M}_Ks)
   %/\ request(M,S,mobileGuard_server_ns,Ns')

    4.  State  = 4 /\ RCV({Mki'.S}_Tki) =|>
State' := 6 /\ secret(Mki', secMK,{M,S})

end role
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role server(M, S: agent,
         Km, Ks: public_key,
         K: symmetric_key,
         Ticket: message,
         Hash: hash_func,
         SND, RCV: channel (dy))
played_by S def=

  local State : nat,
Nm, Ns: text,
Tki,Mki: symmetric_key

  init State := 1

  transition

    1.  State  = 1 /\ RCV({{Nm'.Tki'.M.Ticket}_Ks}_K.Hash(M.Nm'.Tki'.Ticket)) =|>
State':= 3 /\ Ns' := new() /\ SND({Nm'.Ns'.S}_Km)
   /\ secret(Tki', secTK,{M,S})
 /\ witness(S,M,mobileGuard_server_nm,Nm')

    3.  State  = 3 /\ RCV({Ns.M}_Ks)
        =|>
State':= 5 /\ Mki' := new()
           /\SND({Mki'.S}_Tki)
                 /\ secret(Mki', secMK,{M,S})
          % /\ request(S,M,server_mobileGuard_nm,Nm)
```

```
end role

%%

role session(M, S: agent, Km, Ks: public_key, K,Tki,Mki: symmetric_key,Ticket: mes
             Hash: hash_func)
 def=

  local SA, RA, SB, RB: channel (dy)

  composition

mobileGuard(M,S,Ks,Km,K,Ticket,Hash,SA,RA)
     /\ server  (M,S,Km,Ks,k,Ticket,Hash,SB,RB)

end role

%%

role environment() def=

    const m, s         : agent,
  km, ks, ki   : public_key,
  k,tk, mk, tki,mki: symmetric_key,
  secMK, secTK, mobileGuard_server_nm   : protocol_id,
  ticket: message,
```

```
  h : hash_func
 % mobileGuard_server_ns,



    intruder_knowledge = {m, s,km ,ks, h, ki, inv(ki),inv(km)}%,tki,mki}

    composition

session(m,s,km,ks,k,tk,mk,ticket,h)
/\ session(m,s,km,ks,k,tk,mk,ticket,h)
           /\ session(m,i,km,ki,k,tki,mki,ticket,h)
/\ session(i,s,ki,ks,k,tki,mki,ticket,h)

end role
%%
goal
  secrecy_of secMK, secTK
 % authentication_on mobileGuard_server_ns
 % authentication_on server_mobileGuard_nm

end goal
%%
environment()
```

Appendix C

Persistent Protection in Multicast Content Delivery Protocol Validation

In this section, we will check the validity of the proposed protocol using the AVISPA tool. The following is the code used to validate the authentication protocol that takes place between the MR, NSP and VS. Figure 27 illustrates the MG delivery protocol:

PROTOCOL: Mobile Guard Delivery

PURPOSE: This protocol describes the interaction among three major entities: the merchant, the viewer software and the network service provider. The purpose of this protocol is to securely deliver an individualized mobile guard instance to the viewer software, which is located in end user's machine. The individualized mobile guard is a plug-in software that contains the media key. This protocol allows the network server to authenticate and authorise the viewer software to view a requested media document.

ALICE_BOB notation:

- VS ->MR : VS.NSP.MR,{{Nvs.VS.NSP.Minfo.Money}_KMR}_Kvm
- MR ->VS : VS,NSP,MR,{{VS,NSP,MR,K,Nvs,Nmr}_Kvm,Ticket}_Kvm
- VS ->NSP: VS,NSP,Nvs,{Ticket.{Nvs.Nmr}_K}_Kvn
- NSP ->MR: TicketID,{{Nvs,Nmr,NSP,MR,VS,TicketID}_KMR}_K
- MR ->NSP: TicketID,{{Nvs,Nmr,NSP,MR,VS,TicketID,MGi}_KNSP}}_K
- NSP ->VS: VS,NSP,MR,{{Nmr,Nvs,Individualized-MGi_K}_K}_Kvn
- VS ->NSP: {HashValue}_K
- NSP ->VS: VS,NSP,MR,{VS,NSP,Nvs,Nmr,MK}_HashValue

PROBLEMS:

- Secrecy of session key K that is established to be shared between the NSP and VS
- Secrecy of the MG
- Secrecy of the ticket
- Secrecy of the media key mk
- Authentication on the VS nonce
- Authentication on the MR nonce

ATTACKS: No attack has been found

```
role vs (VS, MR, NSP: agent,
KNSP, KMR : public_key,
% KNSP : NSP's public-key and KMR : MR's public-key
Kvm,Kvn : symmetric_key,
% Kvm:Access key between VS and MR  Kvn:Access key between VS and NSP
  Money, Minfo : text,
  % Money : Payment-token and Minfo : Media information
Hash: hash_func,
% Hash : One way hash-function
SND_MRVS, RCV_MRVS, SND_NSPVS, RCV_NSPVS: channel(dy))
% SND_MRVS, RCV_MRVS, SND_NSPVS and RCV_NSPVS are channel controlled by dy
played_by VS
def=
local State : nat,
Nvs,Nmr : text,
% Nvs: a nonce generated by the VS
% Nmr: a nonce generated by the MR
K : symmetric_key,
% K : shared-key issued by the merchant
Ticket : {symmetric_key.text.text.text.text.agent.agent.agent}_public_key,
% The Ticket issued by the MR
MK: symmetric_key,   % media key
MGi: {text.symmetric_key.text}_symmetric_key,
% MGi: An instance of mobile guard for a specific trust-interval
HashValue : message
  % Hash-value: used for integrity checking for the MG+VS+MK+Ticket
const vs_mr_na, vs_nsp_na, vs_nsp_nb  : protocol_id
init State := 0
transition
```

```
% generate nonce Nvs and choose the suaitable media information
% and send them with the Money token to the merchant
1. State = 0 /\ RCV_NSPVS(start)
=|>
   State':= 2 /\ Nvs' := new()
/\ SND_MRVS(VS.NSP.MR.{{Nvs'.VS.NSP.MR.Minfo.Money}_KMR}_Kvm)


2. State = 2
/\ RCV_MRVS(VS.NSP.MR.{{VS.NSP.MR.K'.Nvs.Nmr'}_Kvm.Ticket'}_Kvm)
=|>
    State':= 4
     /\ SND_NSPVS(VS.NSP.MR.{Ticket'.{Nvs.Nmr'}_K'}_Kvn)
/\ secret(Ticket',ticket,{VS,NSP,MR})

3. State = 4
/\ RCV_NSPVS(VS.NSP.MR.{{Nmr.Nvs.MGi'}_K}_Kvn)
=|>
   State':= 6
    /\ HashValue' := Hash(VS.MGi')
/\ SND_NSPVS({HashValue'}_K)

4. State = 6
/\ RCV_NSPVS(VS.NSP.MR.{VS.NSP.Nvs.Nmr.MK'}_HashValue)
=|>
   State':= 8
    /\ request(VS,MR,vs_mr_na,Nvs)
/\ request(VS,NSP,vs_nsp_na,Nvs)
/\ request(VS,NSP,vs_nsp_nb,Nmr)
```

```
end role
%%
role mr (VS, MR, NSP : agent,
KNSP, KMR : public_key,
% KNSP : NSP's public-key and KMR : MR's public-key
Kvm : symmetric_key,
% Kvm:Access key between VS and MR
Money, Minfo : text,
% Money : Payment-token and Minfo : Media information
SND_VSMR, RCV_VSMR,SND_NSPMR, RCV_NSPMR: channel(dy))
% SND_VSMR, RCV_VSMR,SND_NSPMR and RCV_NSPMR are channel controlled by dy
played_by MR
def=
local State : nat,
Nmr,Nvs : text,
% Nvs: a nonce generated by the VS
% Nmr: a nonce generated by the MR
K : symmetric_key,
% K : shared-key issued by the merchant
TicketID : text,
% The Ticket ID generated by the MR
PC: text,   % protection code.
MK: symmetric_key   % Media key
const vs_mr_na : protocol_id

init State := 1
transition
1. State = 1
```

```
/\ RCV_VSMR(VS.NSP.MR.{{Nvs'.VS.NSP.MR.Minfo.Money}_KMR}_Kvm)
=|>
   State':= 3
    /\ Nmr' := new() /\ TicketID':= new()  /\ K' := new()
/\ SND_VSMR(VS.NSP.MR.{{VS.NSP.MR.K'.Nvs'.Nmr'}_Kvm.
{K'.Nvs'.Nmr'.TicketID'.Minfo.MR.NSP.VS}_KNSP}_Kvm)
/\ secret(K',k,{VS,NSP,MR})
/\ witness(MR,VS,vs_mr_na,Nvs')

2. State = 3
/\ RCV_NSPMR(TicketID.{{Nvs.Nmr.NSP.MR.VS.TicketID}_KMR}_K)
=|>
   State':= 5 /\ MK':= new()
          /\ PC':= new()
/\SND_NSPMR(TicketID.{{Nvs.Nmr.NSP.MR.VS.TicketID.PC'.MK'}_KNSP}_K)
/\ secret(PC',mg,{VS,NSP,MR})

end role
%
role nsp (VS, MR, NSP: agent,
KNSP, KMR : public_key,
% KNSP : NSP's public-key and KMR : MR's public-key
Kvn : symmetric_key,   %   Kvn:shared key between VS and NSP
% Kvn:Access key between VS and NSP
Minfo : text,
% Minfo : Media information
Hash: hash_func,
SND_VSNSP, RCV_VSNSP, SND_MRNSP,RCV_MRNSP: channel(dy))
% SND_VSNSP, RCV_VSNSP, SND_MRNSP and RCV_MRNSP are channel controlled by dy
```

```
played_by NSP
def=
local State : nat,
Nmr, Nvs : text,
% Nvs: a nonce generated by the VS
% Nmr: a nonce generated by the MR
K : symmetric_key,
% K : shared-key issued by the merchant
TicketID : text,
% The Ticket ID generated by the MR
PC: text,      % Protection Code
MK: symmetric_key, % Media key,
HashValue : message
% Hash : One way hash-function

const vs_nsp_na,vs_nsp_nb: protocol_id
init State := 5
transition
1. State = 5
/\ RCV_VSNSP(VS.NSP.MR.{{K'.Nvs'.Nmr'.TicketID'.Minfo.MR.NSP.VS}_KNSP.
{Nvs'.Nmr'}_K'}_Kvn)
=|>
   State':= 7
/\ SND_MRNSP(TicketID'.{{Nvs'.Nmr'.NSP.MR.VS.TicketID'}_KMR}_K')
/\ witness(NSP,VS,vs_nsp_na,Nvs')
/\ witness(NSP,VS,vs_nsp_nb,Nmr')


2. State = 7 /\ RCV_MRNSP(TicketID.{{Nvs.Nmr.NSP.MR.VS.TicketID.PC'.MK'}_KNSP}_K)
```

```
=|>
   State':= 9 /\ SND_VSNSP(VS.NSP.MR.{{Nmr.Nvs.{PC'.MK'.TicketID}_K}_K}_Kvn)
/\ secret(MK',mk,{VS,NSP})
3. State = 9  /\ RCV_MRNSP({HashValue'}_K)
=|>
   State':= 11
/\  HashValue' := Hash(VS.{PC.MK.TicketID}_K)
/\  SND_VSNSP(VS.NSP.MR.{VS.NSP.Nvs.Nmr.MK}_HashValue')
end role
%%
role session(VS, MR, NSP  : agent,
KNSP, KMR : public_key,
Kvm, Kvn : symmetric_key,
Money, Minfo : text,Hash: hash_func)
def=
local
SMRVS, RMRVS,
SNSPVS, RNSPVS,
SVSNSP, RVSNSP,
SMRNSP, RMRNSP,
SVSMR, RVSMR,
SNSPMR, RNSPMR : channel (dy)
composition
vs (VS, MR, NSP, KNSP, KMR, Kvm,Kvn, Money, Minfo, Hash, SMRVS, RMRVS,
 SNSPVS, RNSPVS)
/\ mr(VS, MR, NSP, KNSP, KMR, Kvm,  Money, Minfo, SVSMR,
  RVSMR,SNSPMR,RVSMR)
/\ nsp (VS, MR, NSP, KNSP, KMR, Kvn,  Minfo, Hash, SVSNSP, RNSPVS,SMRNSP,RMRNSP)
end role
```

```
%%
role environment()
def=
const vs1, nsp1, mr1 : agent,
knsp, kmr, ki : public_key,
kvm, kvn  : symmetric_key,
money, minfo : text,
h: hash_func,
vs_mr_na, vs_nsp_na, vs_nsp_nb,k, mg, ticket,mk: protocol_id
% Nonce_VS_NSP , Ticket_Key, Mobile_Guard_Mobile_protection, Media_Key
intruder_knowledge = {vs1, nsp1, mr1, ki,inv(ki), knsp, kmr, h }
composition
   session(vs1,mr1,nsp1,knsp,kmr,kvm,kvn,money, minfo,h)
/\ session(vs1,mr1,i,knsp,ki,kvm,kvn,money, minfo,h)
/\ session(i,mr1,nsp1,knsp,kmr,kvm,kvn,money, minfo,h)
/\ session(vs1,i,nsp1,knsp,ki,kvm,kvn,money, minfo,h)
end role
%%
goal
secrecy_of k
secrecy_of mg
secrecy_of ticket
secrecy_of mk
authentication_on  vs_mr_na
authentication_on  vs_nsp_na
authentication_on  vs_nsp_nb
end goal
%%
environment()
```