A PEER-TO-PEER RIGHTS MANAGEMENT SYSTEM

A Report

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Pallavi Priyadarshini

November 2006

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Mark Stamp, SJSU

Dr. Robert Chun, SJSU

Ashish Kumar, Oracle Corp.

APPROVED FOR THE UNIVERSITY

ABSTRACT

A PEER-TO-PEER RIGHTS MANAGEMENT SYSTEM

By Pallavi Priyadarshini

Peer-to-peer (P2P) networks have proliferated and become ubiquitous. A school of thought has emerged with the conviction that harnessing the established user-base and ease of content dissemination presents a lucrative opportunity for businesses to generate revenues through P2P networks. However, content creators have been reluctant in adopting P2P networks as a distribution vehicle to monetize digital content since these networks are plagued with piracy owing to the lack of central policing.

Our project focuses on developing a solution for distributing digital content in P2P networks in a way that both businesses and amateur artists can make a profit without copyright infringement. We propose a P2P content distribution system that integrates Digital Rights Management (DRM) technologies. Our system not only relies on security technologies to deter piracy, but also provides a reselling feature to encourage legitimate buyers to redistribute content for profit.

ACKNOWLEDGEMENTS

I would like to thank Prof. Mark Stamp, my project advisor, who not only inspired me to undertake research involving information security, but also provided his technical and professional guidance throughout the project.

I am also grateful to my family who supported me through the long hours and provided encouragement whenever I needed.

INDEX OF CONTENTS

INDEX OF FIGURES

1. INTRODUCTION

There has been an explosion of digital content distribution through the Internet. Apple iTunes [4] and YouTube [34] exemplify the paradigm shift in music and video distribution. Users are increasingly obtaining digital content through downloads. Given the tremendous popularity of digital content, exploring new channels to enable content distribution and creating new non-traditional marketplaces is a logical next step forward.

P2P networks present the potential of transforming themselves into popular vehicles of digital content distribution. With the continued proliferation of peer-to-peer (P2P) networks such as Kazaa [1] and Gnutella [2], industry and academia are beginning to realize the potential of such networks in this era of digital information. Instead of seeing P2P networks as a threat, a growing number of people believe that P2P networks could be formidable instruments of content dissemination. At present, distributing content through P2P networks is rife with risks of infringement of copyrights. P2P networks lack

many security features inherent in client-server networks to protect the rights of content owners.

Our project focuses on exploring solutions that enable large-scale distribution of digital content in P2P networks such that intellectual property rights are not violated and the content creators are able to collect due profits. We augment the basic distribution through content creators with distribution through authorized resellers. Such extensive distribution can only be achieved by applying appropriate Digital Rights Management (DRM) technologies to P2P networks to ensure that P2P networks benefit creators and legitimate customers, not just pirates.

The rest of the report is organized as follows:

- Chapter 2 gives details about P2P and DRM technologies. It describes the requirements of a DRM system suitable for P2P networks. It delineates the design goals we established for our system. It concludes with a review of related work done in the field of DRM for P2P networks.
- Chapter 3 describes the design of our proposed system. It includes justifications of the decisions we made

during our design. It also provides a description of the system architecture and the functional flow in our proposed system.

- Chapter 4 covers the implementation details of all the components in the system. We do not cover the details of security-related features, which are discussed in the next chapter.
- Chapter 5 covers the security features included in the proposed system. It discusses the implementation aspects and analyzes the strengths of each significant security feature.
- Chapter 6 involves testing of the working prototype of the proposed system. It illustrates the underlying functionality of the prototype by going through the steps of a sample use case.
- Chapter 7 concludes the report by summarizing the achievements of the project and presenting ideas that could be developed as extensions of this project.

2. BACKGROUND

2.1. P2P networks

A peer-to-peer (P2P) network connects the nodes in an ad-hoc manner as opposed to a traditional client-server configuration where all nodes communicate to and from a central server. Figure 1 depicts the difference in the topologies of P2P and client-server networks.

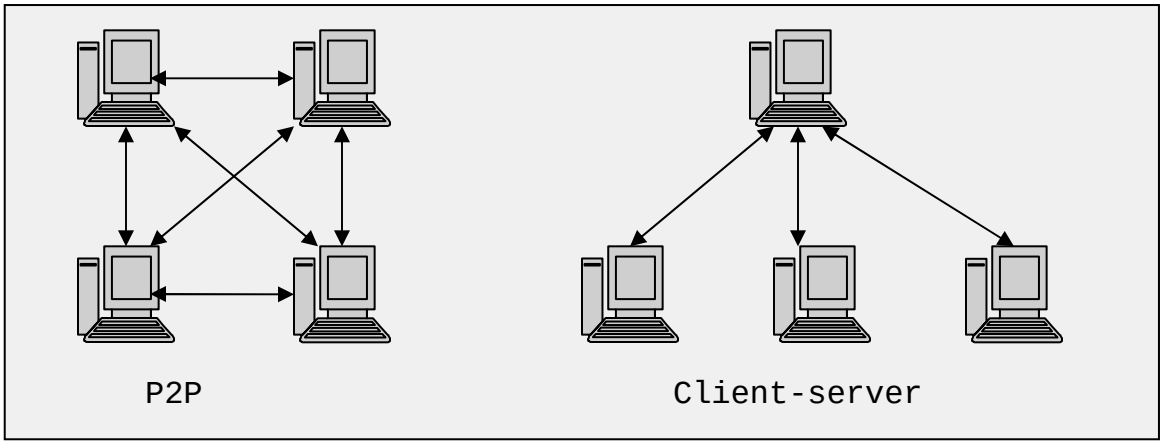


Figure 1. Network topology of P2P client-server networks

Each P2P network has its own policies for connecting the P2P clients. For example, the classic P2P network Napster [3] had a central server used to index all content that peer users had to offer. This central server based approach gradually evolved into more de-centralized networks. Kazaa

and Gnutella are well-known examples of decentralized P2P networks which bypass any central server and directly connect users in the network [1, 2]. Although Kazaa does not have any central indexing servers, it relies on a subset of network users called supernodes. Supernodes are users with more advanced machines and faster connections who host a list of files their neighborhood kazaa users are sharing. Ordinary Kazaa users connect to the supernodes to search content. Gnutella uses a more distributed protocol (also called “pure P2P”) whereby any search requests are progressively routed to directly connected users, effectively creating a “query flood”.

P2P networks have an ever-increasing user base. They provide several advantages over traditional client-server based network model, namely:

- Scalability – P2P networks avoid server bottleneck as content exchange happens directly between peers.
- Faster response - In traditional systems, users in close geographical proximity can end up downloading content from the same central server located far away. In P2P networks, more peers can offer contents that are in high demand, thus improving chances that

content is downloaded from a source that is geographically closer.

- Empowerment of peers – In P2P networks, ordinary peers can exchange files directly with others. Any peer can offer content rather than only central servers.
- More resources in the network – Compared with traditional systems, P2P networks represent an improvement in resources such as computing power, storage and network bandwidth since the member peers can contribute these resources to the network.

However, there is another side to P2P networks. These networks have become havens for pirates of digital content owing to the lack of central policing. Widespread piracy is the primary reason for the reluctance on part of major studios and record labels to embrace P2P networks. This reluctance, though not unfounded, probably cost these businesses dearly. Due to the growing popularity of P2P networks, the industry might be wise to adopt appropriate business models for legitimizing and monetizing content distribution on P2P networks [4]. Apple’s iTunes online music store [5] provides an exemplary business model, where the number of downloads after payment has exceeded one

billion [6]. To harness the advantages P2P networks have to offer with their extensive user base, industry should be the early adopters of technologies enabling legal content distribution through these networks.

2.2. Digital Rights Management

Digital Rights Management (DRM) refers to enforcement of rules and policies to control access to digital content like software, music, videos and documents [7]. It controls not only the way a piece of digital content is delivered to legitimate end-users, but also enforces rules after the content has been delivered. For example, a particular DRM system might allow a legitimate user to access content only three times.

Like P2P networks, DRM also has its own share of detractors. Many people believe that information, especially digital, is free and they find the enforcement of rules on digital goods too restrictive. The authors in [36] argue that DRM has inherent technical limitations with respect to enforcement and efficient “darknets” for illegal

distribution of digital goods would continue to exist. Additionally, most of the DRM products and technologies are proprietary. Given the volatile nature of business, users of any proprietary technologies are legitimately concerned about the support to the DRM products if the company providing the DRM products goes out of business. However, it is necessary to recognize that businesses need to be adequately profitable to fuel further innovations. While too many limitations on the purchase of content can understandably irritate consumers, a reasonable level of enforcement might lead to equitable profits for content creators and consequently foster creativity.

2.3. DRM for P2P networks

Harnessing the wide user-base of P2P networks presents a huge opportunity for both established content providers and amateurs. Utilizing P2P networks for distributing content is challenging due to the lack of central control. Owing to the popularity and benefits of P2P networks, it is worthwhile to explore solutions that would provide an appropriate balance between distributing content on a large-scale and preserving the right of intellectual

property owners. DRM can provide a means to achieving that end [8, 9].

DRM solutions make it possible to honor the rights of copyright holders. However, conventional DRM techniques would not be effective in a true P2P network owing to the decentralized nature of distribution and complete control of peers over the network. While a typical DRM system is tailored for a client-server model of distribution, content distribution in a P2P network can lead to new requirements due to the following reasons:

- Any ordinary peer in a P2P network can be a content creator, distributor and seller.
- Content is not downloaded from a central repository in a P2P network. Peers can exchange content with each other.

Any DRM system designed for P2P network has to take into account the above-mentioned requirements of P2P model of distribution to be effective.

2.4. Our motivation and goals

The goal of our project is to demonstrate a successful convergence of DRM and P2P. The root of this project is attributed to our search for a DRM solution tailored for P2P networks that enables rapid content distribution. While there are well-established DRM techniques, applying DRM to P2P networks presents complex functional and security requirements. The complexity can be exemplified by the following comparison: Apple iTunes music store has a central distribution site, and the songs are protected with proprietary DRM technology called FairPlay [28], which can be downloaded onto Apple iPods for playing. Contrast this to a P2P network, where any peer can be both creator and distributor, and the content is not accessed through any particular product. It was challenging to devise a reasonably secure architecture tailored for P2P networks with important functions decentralized.

We aimed to look for a solution that would accomplish multiple goals we established for our system. These goals are enumerated below (each of these design goals has been

tagged with an identifier, eg. DG1 for Design Goal 1, for convenient cross-referencing):

DG1: Well-defined roles of the peers in the system.

DG2: Any peer can be content creator, copyright owner and seller.

DG3: Redistribution with profit sharing by peers

DG4: Essential functions like packaging content are decentralized.

DG5: Independence from underlying peer-to-peer network

DG6: Minimal overheads for end users

- No need for pre-installed client software
- Convenient payments

DG7: Code portability on platforms

We wanted to complement the functional features of the solution with some distinctive security features (within the inherent limitations of DRM), namely:

DG8: Secure transactions

DG9: Secure authentication and authorization without excessive overhead.

DG10: Secure license distribution.

DG11: Break once, break everywhere (BOBE) resistance, which implies that an approach used to compromise one

piece of content cannot be successfully applied to compromise another piece of content.

2.5. Existing research

In our search for existing industry and academic research, we discovered several promising solutions that addressed some aspects of our functional and security goals.

The Music2Share (M2S) system [10] proposes a P2P protocol based on audio fingerprinting and watermarking. The proposed protocol in [10] makes several questionable assumptions, such as the assumption that general users are honest. In addition, many of the crucial technologies on which the proposal rests are categorized as research problems. Consequently, many of the most critical security issues are essentially ignored. M2S would also not operate on existing P2P networks and it does not have features for redistribution of contents. The authors of Music2Share have also stated “to date there exists no operating M2S network”.

Berket, et al [11] have proposed a solution using Secure Group Layer (SGL) to achieve group communication. Their system makes heavy use of public-key infrastructure (PKI), and the heavy reliance on PKI technologies comes with considerable overhead. The security ideas presented in [11] have not been implemented. The intent of their publication is to serve as a baseline for other types of P2P applications.

Iwata, et al [12] discuss the requirements of a DRM system applicable to P2P content sharing. The authors present a very high-level functional description without giving any insight into the underlying technologies. The functions identified in [12] do not meet many of our stated design goals.

CITADEL [13] uses the concept of “content containers” to pack role-based access control lists (ACL) with the content being distributed. It is not clear how users obtain different roles in the system. CITADEL also makes use of digital signatures and X.509 public-key certificates for all users, which are expensive operations for practical use.

The DiMaS system [14] also proposes a solution based on distribution packages. While DiMaS is successful in eliminating the need for client software on the machines of end users, it does not incorporate reselling of content or BOBE resistance. In fact, none of the systems described above, or any of the other systems we evaluated ([15] [16] [17]), support explicit features for BOBE resistance or redistribution and profit sharing between content owners and resellers through reselling. Thus in our search for a system that would provide a complete solution to our design goals stated above, we did not encounter any reasonably complete solution. Our objective is to design and develop a DRM system for P2P networks with both improved and new features as compared to existing designs.

3. DESIGN

We have designed a DRM system tailored to the needs of P2P networks. We call our system P2PRM, which is an acronym for Peer-to-Peer Rights Management. In this section, we discuss the design decisions we made and describe the architecture of the proposed system.

3.1. Design Decisions

Throughout the design of P2PRM, we attempted to keep in focus the design goals we established in Section 2.4. In each sub-heading that follows, we identify the design goals that the given design decision fulfills.

3.1.1. A hybrid approach

The goal of P2PRM is to enable an ordinary peer to benefit from content creation. A content creator can make his work available for download by other users on a peer-to-peer network, which provides ready access to a huge audience-base. Creators who wish to distribute their creations free of cost do not need to take any extra step for

distribution. However, if a creator wishes to charge a fee for use of his content, additional precautions need to be taken to ensure that he has a reasonable chance of collecting the fee.

Our goal was to make P2PRM as decentralized as possible. We considered possible models for P2PRM without any central transaction server to take care of transactions and billing. In such a system, all billing related functions would be done at individual peers. Although this model is completely decentralized, it is more insecure and unreliable as compared to a model with central transactions server. It is virtually impossible to ensure fair financial transactions when peers are tracking them, since there is no validation of the transactions by a third party. Accounting can become very difficult since, for example, a malicious peer can tamper with transaction logs stored at his terminal.

To improve security, we decided to use a central trusted peer, which we call the Secure Super-Peer (SSP), to track all transactions, maintain all billing records, validate any financial information, and provide other security

related functions. Only a trusted peer can act as the SSP to enable large-scale secure distribution of content. The introduction of an SSP makes our model a hybrid model, since we have some central point of control in addition to the distributed peers. Adopting a hybrid approach fulfils the design goal identified by ***DG8*** (secure transactions).

3.1.2. Entities

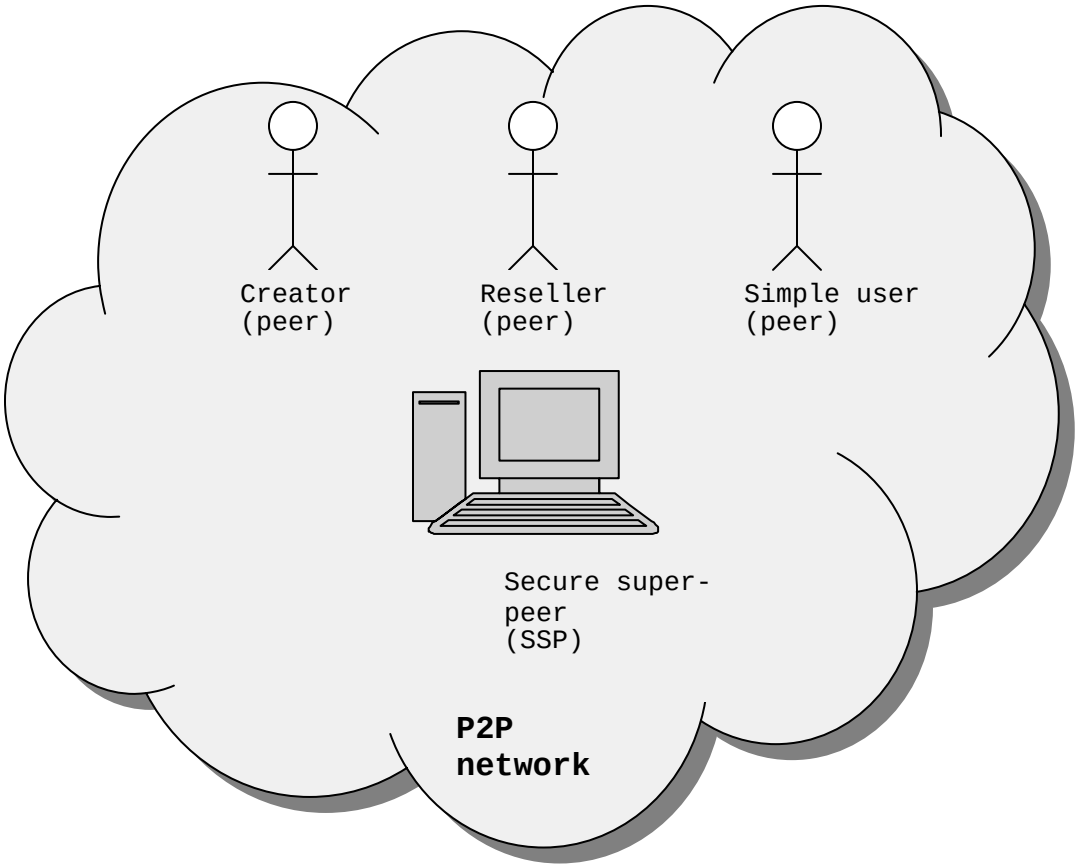


Figure 2. Entities in P2PRM

As shown in Figure 2, our proposed model has four main entities, namely:

- Creator - any peer who creates content to be distributed.
- Simple user - any buyer in the P2P network who purchases content for his own use.
- Reseller - any buyer in the network who purchases content not only for his own use, but also to redistribute to make a profit.
- Secure super peer (SSP) – server that handles all transactions between other entities in P2PRM, and performs some security related tasks in P2PRM. We assume that the SSP is trusted by all other entities in the network.

Classifying users in the system in this way fulfils the design goal identified by ***DG1*** (well-defined roles).

3.1.3. Content distribution

Any peer who creates content can make that content available for use by another peer. If the creator wishes to charge a fee for usage of his content, the system should

incorporate additional functionality to govern the entire process of creation and consumption of content. Financial transactions lead to the following new requirements as opposed to distributing content free:

- Establishing identities of entities involved in a transaction (seller, buyer and any intermediary).
- Protecting content from unintended recipients.
- Keeping track of all transaction details and preventing any illegitimate profits and transactions.

These functions are performed at different entities in P2PRM, namely, at the creator, buyer, reseller and SSP. In addition, we introduce another entity, called the content wrappers, which work together with other entities and make secure transactions and rights enforcement possible. In P2PRM, the content is not distributed as-is. Rather, it is packaged with some additional information and logic that plays a vital role in rights management and enforcement. We refer to the packaged content as the wrapper. Instead of distributing content, the creators distribute wrappers, as shown in Figure 3. The buyer needs to purchase access permission in order to access the content embedded in the wrapper.

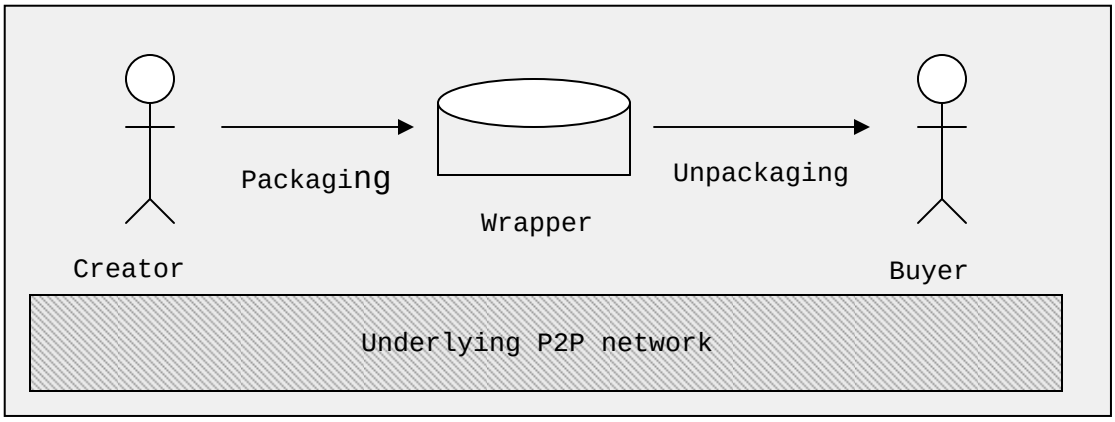


Figure 3. Wrapper distribution environment

The wrappers can be downloaded free of charge. Only when a buyer wishes to access the content does he need to make payment to the SSP to obtain access rights. Packaging and distribution of contents using wrappers fulfils our design goals identified by ***DG2*** (any peer can be content creator and seller) and ***DG6*** (minimal overheads for end users).

3.1.4. Model for reselling

We have not seen any DRM enabled content distribution system for P2P networks that facilitates reselling. One of the strengths in the design of P2PRM is that it enables content reselling, which can lead to larger scale distribution of content and deter piracy through profit sharing. Complexities in the reselling model arise because:

- Resellers cannot be trusted with unencrypted content.
- A flexible model should allow any reseller to repackage content with his own profit margin (the new price should reflect both the creator's and reseller's markup).

In this section, we describe some concepts applicable in P2PRM that focus on the area of reselling content.

A creator can decide upon a markup that he wishes to charge in case resellers sell his content. Every time a reseller sells a wrapper, the creator becomes entitled to the markup he had specified during the original wrapper creation. To facilitate selling to both simple users and resellers, a creator can package his content in two forms: a user wrapper and a reseller wrapper. A user wrapper is intended for buyers who would use the content for their use only. A reseller wrapper is meant for buyers who wish to repackage the content and charge a markup for redistribution. Note that P2PRM allows a reseller to repackage a purchased content for distribution only as a user wrapper, not another reseller wrapper.

When a creator packages content, P2PRM makes a distinction between user price (price of a user wrapper) and reseller price (price of a reseller wrapper). It is generally expected that reseller price would be higher than user price since a reseller can potentially benefit from redistributing content.

Allowing resellers to redistribute content works to the advantage of several entities in the system, namely, content creator, reseller and simple user. A content creator can discover a larger audience for his creation. Every time his content is sold by a reseller, the creator receives his markup. In this way, he may obtain a larger profit as compared to selling his content by himself. Another advantage to the creator is that there would be less chance of another user compromising his content since reseller profits give an incentive to other users in the system for legal distribution.

For a reseller, P2PRM provides opportunities to make profits. A reseller can repackage content with a new price, which would include creator markup and reseller markup. If a reseller is on a connection that can provide faster

download times for the wrapper, potentially, he can even charge a higher price for the content.

For any buyer, P2PRM provides more choices. The buyer can choose one seller from the many sellers of content (creator and resellers). His decision to purchase a particular copy of the wrapper can be based on several factors:

- Price of content offered by a seller
- Speed of connection a seller is connected to (dial-up, cable, DSL, T1/T3).
- Geographical proximity to the seller.

Having resellers create and distribute wrappers fulfils design goal ***DG3*** (redistribution with profit sharing).

3.1.5. Content Packaging

One option would be to create the wrapper at the SSP. This would require the creators to upload the content to the SSP. The SSP would then process the content to create the wrapper. Packaging content at the SSP has some drawbacks as discussed below:

- It would make the system more centralized.

- Packaging is an intensive process. It would pose more computing demands on the SSP.
- Content sizes can get very large, especially audio and video content. Uploads to the SSP can be very time-consuming for large content.

Since our goal is to make necessary functionalities decentralized in P2PRM without compromising security, we decided to perform the function of content packaging at the peer. Utilizing computing power of individual peers is a cornerstone of P2P advantages and this would make the system more efficient overall. Creating wrappers at peers instead of the SSP fulfils design goal ***DG4*** (essential functions decentralized).

3.1.6. Use of PKI

One possible approach would be to rely on Public-Key Infrastructure (PKI), namely certificates. The primary objective of using certificates would be authenticating entities in the system. It is well-known that public-key operations are expensive. Such operations would also require all peers to have access to a Certificate Authority

(CA) who can grant certificates to them. This would impose other requirements on the system, which we deemed unnecessary. We decided to eliminate the need for certificates for all the peers except the SSP and develop alternative identification schemes for the rest of the peers. Only the SSP would need a valid certificate since it is an entity trusted by all peers in the system. On the Internet, we have the analogous situation, where, for example, Amazon has a certificate, but ordinary users are not required to have certificates. Typically, any P2P network would work with only one SSP and requiring a certificate for only the SSP would not cause significant overhead. Eliminating the need for certificates for peers in P2PRM fulfils design goal identified by ***DG9*** (secure authentication without excessive overhead).

3.1.7. License distribution

We considered different approaches for license distribution:

- Independent license manager - It would be responsible for distributing decryption keys to the buyer once appropriate payments are made at the SSP. An

independent license manager would add an extra central point of control in the system.

- SSP - Without any compromise in security, we could move the functions of license manager to the SSP. The SSP would be responsible for storing the decryption keys and distributing them. While license functions would have made the SSP more complex, we did not find a significant security advantage as opposed to storing the license with the content itself.
- License with content – We could store the license with the content itself. To achieve security, the decryption key for the content is encrypted with a master key known only to the SSP. We opted for this approach as it eliminates the need for the SSP to maintain a database of all decryption keys in the system and we did not see any threats beyond those involved in storing the key at the SSP. Distributing licenses with the content securely meets the design objective identified by ***DG10*** (secure license distribution).

3.1.8. Independence from underlying P2P network

One of our design objectives was that P2PRM should be able to work with any underlying peer-to-peer network. That is, it should not be tied to the technologies or protocols used in any particular P2P network. Rather, all functions of P2PRM could be incorporated with any network. To achieve this, we made P2PRM self-contained in that it performed all functions using its own modules and standards-based protocols and does not rely on a particular underlying P2P network. Our model utilizes the P2P network only as a vehicle for distribution. Once a wrapper is created using P2PRM modules, it can be made available on any P2P network for distribution. The wrapper should also be accessible using P2PRM modules so that any buyer can open the content upon obtaining the wrapper from any P2P network. By not depending upon any particular P2P technology conforms to design goal ***DG5*** (independence from P2P network).

3.1.9. Need for client software

We wanted to eliminate the need for peers to install any kind of client software on their terminals. However, packaging of content wrappers by the creators on their

terminals necessitates some software to do the packaging. We require all content creators to install P2PRM client software (called wrapper creator software). Buyers in the system, namely the resellers and simple users, do not need any client software to repackage or access the content. All the logic for repackaging and access is included in the wrapper itself. This logic is referred to as the wrapper access software in P2PRM. While the wrapper creator software needs to be installed on a creator's terminal, the wrapper access software is launched automatically by opening the wrapper. By doing away with installing any client software for buyers, we believe P2PRM is more flexible, and has reduced setup needs. This is also in line with our design objective ***DG6*** (minimal overhead for end-users).

3.1.10. Portability

P2PRM modules should be portable across platforms. All terminals belonging to users of a P2P network should not be assumed to be uniform. Accordingly, all modules in P2PRM should be capable of executing on all platforms. We designed P2PRM keeping in mind openness and standards-

compliance for easy portability. This meets our design criteria ***DG7*** (code portability).

3.1.11. BOBE resistance

According to our design objective ***DG11*** (break once, break everywhere resistance), P2PRM should foil BOBE attacks. This means that an attacker, upon compromising one piece of content, should not be able to follow an identical approach to compromise another piece of content. The purpose of BOBE resistance is to keep the amount of compromised contents in the system to a minimum by increasing the work for attackers. This can be achieved by individualizing each wrapper (which encapsulates content) using some proprietary algorithms. We designed each wrapper so that before the underlying content is encrypted with well-known cryptographic algorithms, it is scrambled with one of a variety of proprietary algorithms. Since these scrambling algorithms are not in the public domain, it obfuscates the wrapper code and renders large scale reverse engineering attacks more difficult.

3.2. Architecture

In this section, we discuss in detail the content of the wrapper, the processing at different entities and the interactions amongst different entities, which culminate in the creation of the content wrapper and access of content.

3.2.1. Content Wrapper

Before we delve into the functional architecture of P2PRM, it is important to know what information is included in a content wrapper. We noted in earlier sections that a content wrapper is one of the following two types:

- User wrapper – These wrappers are created for simple users (who purchase the content for their use, not for reselling)
- Reseller wrapper – These wrappers are created for resellers and include the capability to repackage the underlying content according to a new price determined by the reseller.

Figure 4 presents a pictorial representation of a content wrapper.

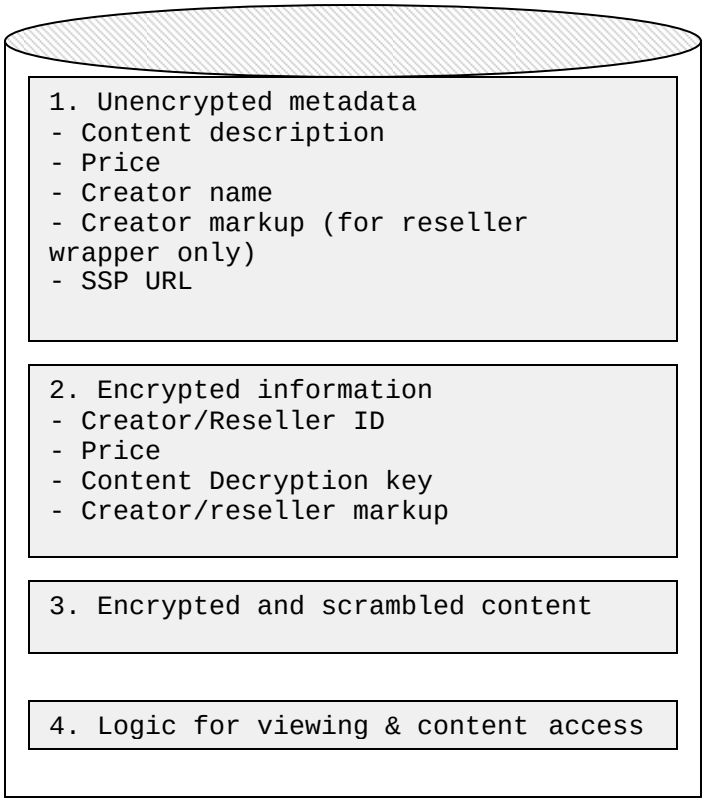


Figure 4. Contents of a content wrapper

As shown in the figure, a content wrapper has the following components:

- Unencrypted metadata – This consists of information that has a potential buyer can view even before purchasing rights to access underlying content of the wrapper. This includes a description of the content, its price and the name of the creator. For a reseller wrapper, the creator markup is also a part of this

information. In addition, the URL of SSP is included so that a connection to the SSP can be established to perform the SSP related steps during content access.

- Encrypted information – This includes sensitive information, which is encrypted to prevent tampering by attackers. The sensitive data includes the price of accessing the content, creator identifier and decryption key with which the content can be decrypted by a buyer. In case of a reseller wrapper, reseller identifier and reseller markup are also encrypted in the wrapper. All this information is encrypted with a secure symmetric master key belonging to the SSP (K_{master}).
- Encrypted and scrambled content – The content is scrambled and then encrypted with a symmetric key (K_c) before being included in the wrapper.
- Logic for viewing and access – Since the wrapper is self-contained, it includes all logic for a buyer to purchase and access content. This eliminates the need for dedicated client software on a buyer's terminal. A reseller wrapper also includes logic for a reseller to repackage content with a different price determined by the creator markup and reseller markup.

3.2.2. Functional architecture

Figure 5 depicts the functional architecture of P2PRM:

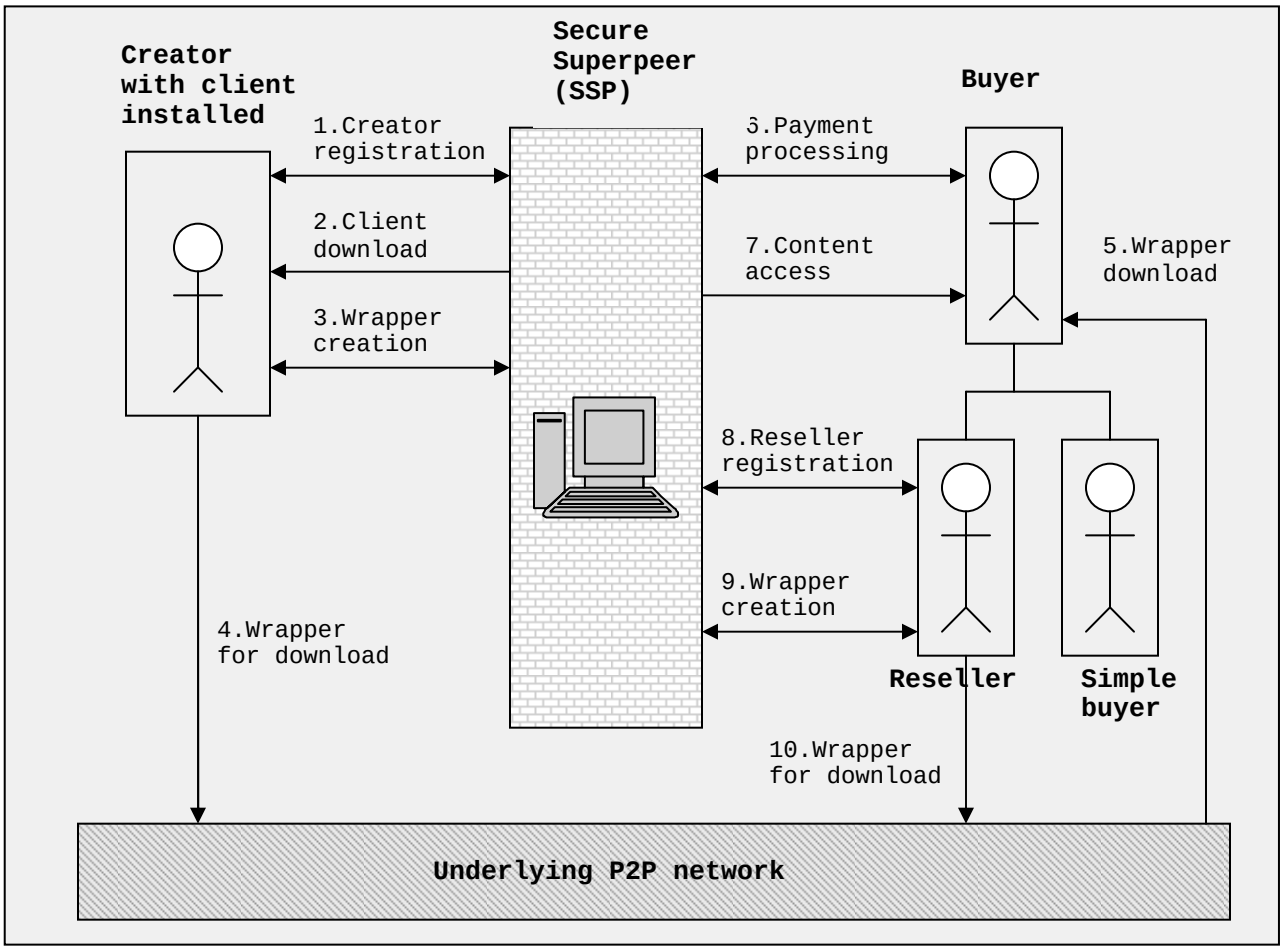


Figure 5. Interactions between system entities

All communication between different entities takes place through secure channels. We discuss the interactions between the components in more detail in the order of step numbers depicted in Figure 5:

Step 1) Creator registration – Any peer who wishes to distribute content created by him for a fee needs to register at the server. This step generates a unique identifier for the creator. Note that the identifiers assigned to the creators and resellers are only for billing purposes. If another dedicated server is introduced in the system to take care of all billing functionality, there is no need for the SSP to remember all the identifiers. The billing server could then track all the identifiers and transactions while the SSP would just need to perform master key encryption and decryption operations. Such an approach would make the SSP even more lightweight.

Step 2) Client download – A creator needs a copy of the client software for packaging content called the wrapper creator software in P2PRM.

Step 3) Wrapper creation – This step can be divided in the following sub steps:

- A creator opens the wrapper creator software, enters details such as name, user price, reseller price, creator markup, content location and description. He

then chooses to create a user wrapper or reseller wrapper.

- The wrapper creator software scrambles the content, generates a symmetric key $K_{c,}$ and encrypts the scrambled content with K_c.
- It then sends sensitive information like creator ID, price, reseller ID (for a reseller wrapper) and decryption key K_c to the SSP.
- The SSP has a symmetric master key K_{master} which it uses to encrypt the sensitive information received from the wrapper creator software. It then sends the encrypted information back to the wrapper creator software.
- The wrapper creator software then creates the content wrapper with the unencrypted metadata, encrypted information, encrypted and scrambled content, and logic for accessing content by a buyer.

Step 4) Wrapper for download – The creator makes the content wrapper available for download by other peers by hosting the wrapper on his machine and making it searchable in the underlying P2P network. Since the system allows for resellers, multiple copies of one wrapper can exist for

download at a given time at different peers in the same network.

Step 5) Wrapper download – A peer can obtain a content wrapper through multiple sources in the P2P network, namely, the creator or any reseller.

Step 6) Payment processing – A peer can obtain access right to the content in the following way:

- He opens up the downloaded wrapper, which launches the wrapper access software to view information about the content including its description, price and creator.
- Upon viewing the information, he can choose to pay the required sum at the SSP by entering payment information.
- The wrapper access software sends the payment information to the SSP along with the master key encrypted information included in the wrapper.
- The SSP decrypts the master key encrypted information to obtain the creator ID, price and the content decryption key K_c. From the provided credit card information of the buyer and the ID of the creator, it performs the necessary steps for a financial

transaction, namely, validating payment information, charging the buyer's account for the required sum, crediting the creator's account for the same sum and logging the details of the transaction.

- The SSP then securely sends the decryption key K_c to the wrapper access software that had initiated the current session.

Step 7) Content access – Upon obtaining K_c, the wrapper access software decrypts the encrypted content with K_c and de-scrambles the result to obtain the content. It then invokes the default content reader/player on the buyer's terminal for the buyer to access content.

Step 8) Reseller registration – In case a buyer wishes to access a reseller wrapper, in addition to payment processing, the SSP registers the user as a reseller. The SSP generates a unique ID for the reseller and passes it back to the wrapper access software for use in wrapper re-packaging.

Step 9) Wrapper creation – A reseller can repackage the encrypted and scrambled content as a user wrapper with a

new price. This new price is calculated by the wrapper access software as a sum of the markup expected by the reseller and the creator markup included in the downloaded wrapper. Since the creator markup in the downloaded wrapper is encrypted with K_{master}, it is decrypted by making a request to the SSP.

Step 10) Wrapper for download – After a reseller creates a repackaged user wrapper, he can make the wrapper available for download by other peers in the network.

Table 1 summarizes the functions of different entities in P2PRM.

Table 1. Functions of entities in the system

Component	Functions
Creator	-Content creation -Content packaging -Upload content wrapper to P2P network
Reseller	-Download reseller wrapper from P2P network -Pay to the SSP -Access content -Repackage content -Upload user wrapper to P2P network

Simple user	- Download user wrapper from P2P network -Pay to the SSP -Access content
SSP	-Creator and reseller registration -Maintain billing information -Master-key encryption/decryption

4. IMPLEMENTATION

In this section, we cover details of the implementation phase. We strived to implemen P2PRM using open source and standards based technologies. We used Java as the development tool, Eclipse as the development environment and Windows XP as the development platform.

The implementation of P2PRM is divided into three major modules: Secure Super-Peer (SSP), wrapper creator software and wrapper access software. In the following sections, we discuss key implementation aspects and present UML (Unified Modeling Language) class diagrams for all three modules. The UML diagrams were created with UML editor Violet software [23]. All user-interfaces associated with the three modules were implemented with Java Swing [22].

Note that in this section, we do not cover implementation of security-related features for the above-mentioned modules. Security characteristics included in P2PRM are critical to the system functionality, which warrants their discussion in a chapter devoted fully to Security Analysis (Chapter 5).

4.1. Secure Super Peer

4.1.1. Multiple connections

SSP has to be able to handle connections emanating from multiple creators and buyers concurrently. It is therefore modeled as a multi-threaded server capable of simultaneously managing multiple connections. Upon accepting a client connection request, the SSP spawns a SSP thread dedicated to service the requests associated with that connection. This allows the SSP to accept new incoming client connections while servicing prior requests from other clients.

4.1.2. Identification of users

The SSP assigns a unique identifier to each creator and reseller so that it can keep track of monetary transactions for billing purposes. We have used the java service java.rmi.server.UID [29] to generate identifiers that are unique to the host they are generated on. Each identifier consists of the following components to make it unique:

- A Virtual Machine (VM) identifier, which uniquely identifies the VM with respect to the host that the identifier is generated on.
- The system time when the VM generated the identifier.
- A count to distinguish between identifiers generated by the same VM at the same time.

For large-scale implementations, this identifier which is unique with respect to the host, can be combined with a unique identifier for the host itself (eg., IP address) to create a globally unique identifier.

4.1.3. User interface

User interface for the SSP consists of a main SSP window and one child window for each SSP thread dedicated to a client connection. The main SSP window allows a creator to register with the SSP. It also displays connection information such as the port number it is listening on and messages about accepting client connections. When the SSP accepts a new connection and spawns a SSP thread to service that connection, a separate SSP thread window becomes active for that particular connection and displays all

information exchanged with the client associated with that connection.

4.1.4. Class Diagram

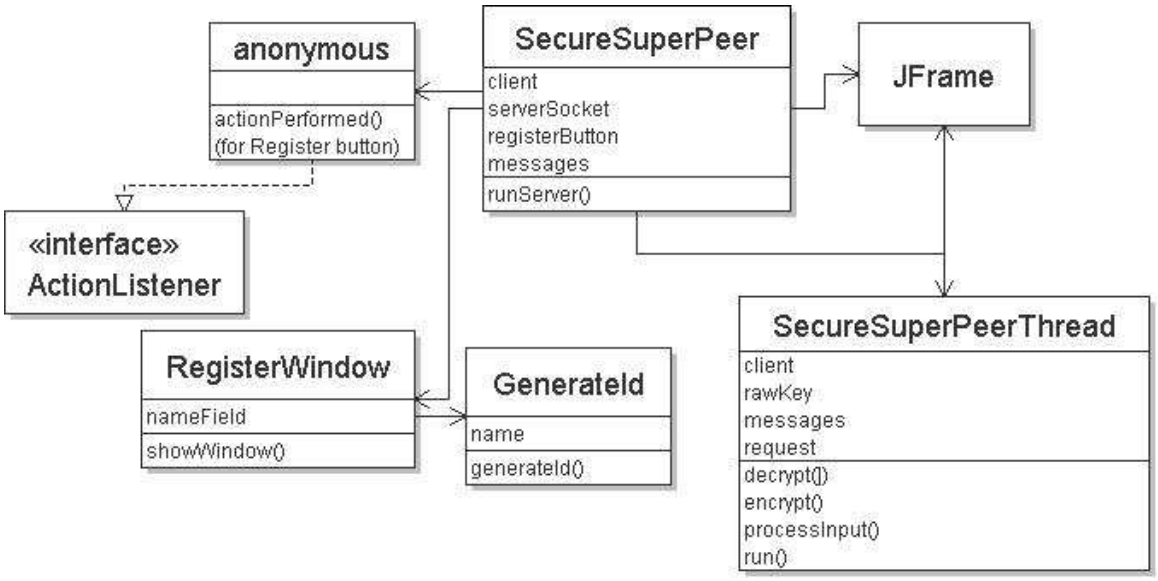


Figure 6. SSP class diagram

4.2. Wrapper creator software

4.2.1. Design pattern

One of the ways to achieve modular and reusable code in software is to apply proven design patterns. We applied Model-View-Controller (MVC) architecture [30] to the

wrapper creator software. Conforming to this architecture, we have:

- a Model class that contains all the data required by the wrapper creator software.
- a View class to display input fields that a creator needs to complete during wrapper creation.
- a Controller class to handle the input entered by the creator, update information contained in the Model class and start the wrapper creation process.

4.2.2. Wrapper as an executable archive

The wrapper creator software executes all the steps needed to create a content wrapper, such as information exchange with the SSP and encapsulating the content with both unencrypted metadata and encrypted data. Both user wrappers and reseller wrappers have been implemented in P2PRM. The wrapper creation implementation for a reseller wrapper is more advanced compared to the user wrapper, since the reseller wrapper has capabilities to perform additional operations needed to repackage content.

Since our design eliminated the installation of any client software for buyers, we had to look for implementation alternatives by which the wrapper access software could be invoked at the buyer’s terminal during access without prior installation. The most buyer-friendly approach would be double clicking a content wrapper on the buyer’s terminal should automatically launch the wrapper access software. To achieve that, we have implemented content wrapper as an executable archive in Java called executable jar [31]. The content wrapper encompasses not only the encapsulated content, but it also includes logic for completing a purchase and accessing content. The wrapper creator gets all information to be included in the wrapper such as the protected content ready and packs them together with the logic files (java classes) needed by the wrapper access software. Double-clicking on the archive created by the wrapper creator will automatically launch the main class of the wrapper access software. The main class specified in the archive manifest [31] serves as the entry point into the wrapper access software.

4.2.3. Format of unencrypted metadata

All unencrypted metadata in the content wrapper like the creator name and description is stored in the form of XML (Extensible Markup Language) [21]. This implementation decision was inline with our goal to implement the system using standards-based technologies. XML has emerged as the standard for representation and exchange of data. The wrapper creator software is capable of writing out data in the XML format to be included in the content wrapper, which can be used by wrapper access software during content access processing.

Note that the wrapper creator software stores some information in the wrapper such as the content price in both encrypted and unencrypted formats. While the encrypted form is tamper-resistant and used by the SSP for billing when the buyer purchases content, the unencrypted format is viewable by the buyer through the wrapper access software in order to make a purchase decision. During the decision making process for purchase, no connection to the SSP is required as the wrapper access software can display the unencrypted metadata to the buyer locally. After viewing

this information, if the buyer chooses not to purchase the content, unnecessary connections to the SSP are averted.

4.2.4. User-Interface

The wrapper creator software has a user-interface component to enable the creator to enter information about the wrapper being created such as creator name, content description, price and creator markup. The creator has a choice to create a user wrapper by clicking on a create user wrapper button or create the reseller wrapper by clicking on a create reseller wrapper button.

4.2.5. Class diagram

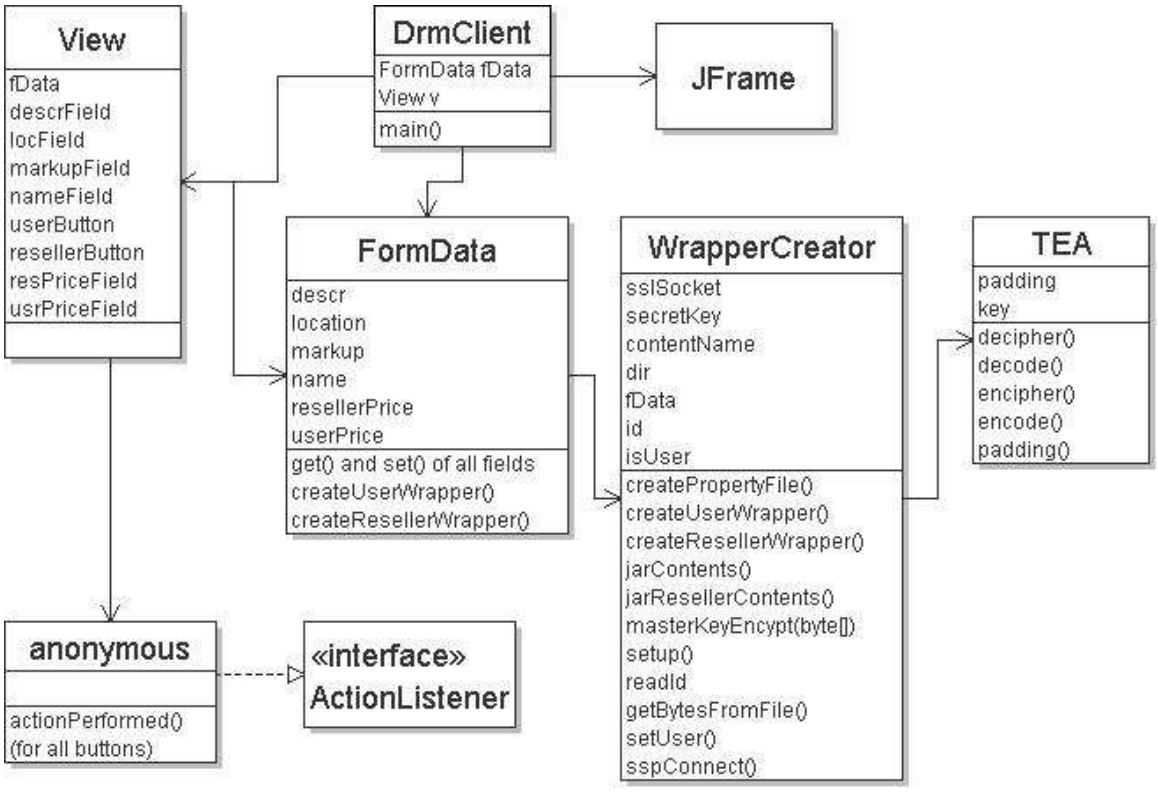


Figure 7. Wrapper creator software class diagram

4.3. Wrapper access software

4.3.1. Archive Extraction

The wrapper access software is launched by double-clicking the downloaded content wrapper by a potential buyer. During launch, all the files packed in the content wrapper archive are unpacked into the current directory. These files include all the data files (like xml) and logic-related

files (java classes). Wrapper access software uses the unpacked files to display relevant information to the potential buyer (like content description, creator name, price) which are useful to make a purchase decision. The wrapper access software gives the choice to purchase access rights to the content, play content or even repackage content (for a reseller wrapper).

4.3.2. User-Interface

The wrapper access software has a user-interface to display all the details of the content when the wrapper is downloaded and double-clicked. This window also includes a button for the user to purchase access rights to content. The user enters all payment information upon clicking the purchase button. The user can then checkout his transaction after which wrapper access software establishes a secure communication link with the SSP to perform the transaction. The play button gets enabled when the transaction completes successfully and the SSP sends the content decryption key to the wrapper access software. The buyer can then click the play button to access the content.

In case the wrapper downloaded was a reseller wrapper, after the payment transaction with SSP is completed, the wrapper access software also enables a repackage button in addition to the play button. Using the repackage button, the buyer can repackage the underlying content. The wrapper access software allows the reseller to enter his markup to be able to calculate the new price of the repackaged content.

4.3.3. Class diagram

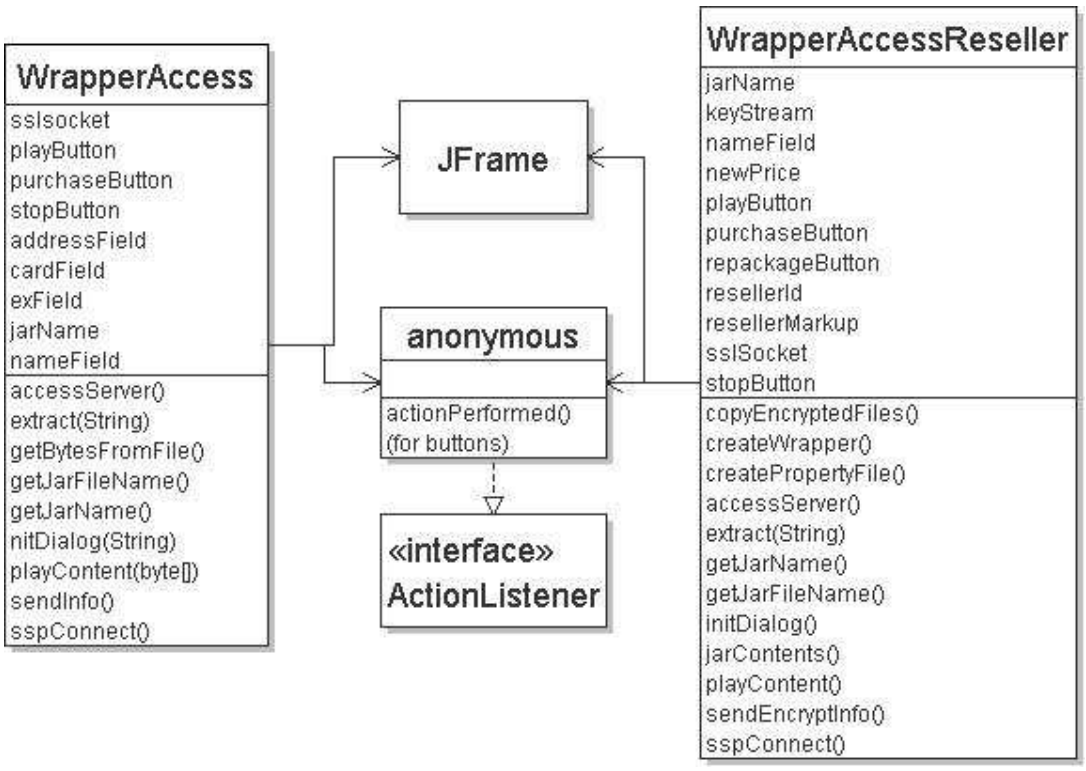


Figure 8. Wrapper access software class diagram

5. SECURITY ANALYSIS

In this section, we discuss the security-critical features implemented in P2PRM. We have divided this section into the three main implementation modules of P2PRM, namely, SSP, wrapper creator software and wrapper access software, and discuss the security features of each module. For each feature presented, we consider possible attacks, and discuss the resilience of the system to such attacks. Note that throughout this section, we use the terms server and SSP interchangeably, and often use the term client for the wrapper access and the wrapper creator software.

5.1. Security features in SSP

5.1.1. Secure Sockets Layer (SSL)

A client transfers several pieces of sensitive data to the SSP. A creator of content has to send the content decryption key, creator ID and content price to the SSP for master key encryption. A buyer of content has to send credit card details in order to purchase access rights to the content. Sensitive information transferred to the SSP from the client can be considered reasonably secure only when the following three conditions are met:

- Identity of the SSP is validated (authentication).
- Communication channel through which information is exchanged is protected so that only the intended recipients view the information (confidentiality).
- Data in transit is not intercepted and modified by an attacker (integrity).

We achieve all the above functions, namely, SSP authentication, confidentiality and integrity, in P2PRM using Secure Sockets Layer (SSL) [32]. While the first two functions of authentication and confidentiality are addressed by SSL through handshaking and encryption, message integrity is achieved by using secure hashing algorithms.

SSL handshaking is the first step in SSL communication. It achieves two purposes: authenticating the server and generating keys to be used for encryption of communication. In the subsequent sub-sections, we cover more of the general mechanics of SSL and P2PRM specific implementation details.

5.1.1.1. SSP authentication through SSL

In P2PRM, only the client authenticates the SSP and not vice-versa. The SSP needs a certificate for its authentication, which involves the use of public-key cryptography. The SSP generates a pair of keys, namely a public and a private key. While it keeps the private key to itself, it distributes the public key through certificates. A trusted third-party certification authority (CA) digitally signs these certificates, which guarantees the identity of the owner of the certificate.

When the server presents its certificate in the handshake phase, the client can trust the public key of the SSP upon verification. SSP authentication protects the SSP URL included in the wrapper, which is unencrypted and thus susceptible to tampering. An attacker could modify the unencrypted SSP URL to his own URL in order to masquerade as the SSP and redirect all communication from the client to himself. If this happens, although a client will open up connection with the forged SSP, communication with the forged SSP will terminate as soon as SSP authentication fails through handshaking. The forged SSP will not be able

to produce a genuine certificate for verification by the client. SSP certificates ensure that that a client exchanges information only with the genuine SSP.

5.1.1.2. Secure data exchange through SSL

In addition to authenticating the server, SSL serves another critical security purpose, namely, confidentiality of data in transit between the client and server through encryption. After server authentication, at the end of the handshaking phase, the SSP and client establish the secret symmetric key used to encrypt all ensuing communication for that particular session. Encryption with a secret key algorithm foils any eavesdropping of information exchanged between the client and server by an attacker.

SSL allows the server and client to agree upon secret keys for secure communication at the time of the transaction itself. No prior distribution of secret keys is necessary eliminating the possibility of secret key compromise during distribution. The algorithm used to encrypt SSL traffic requires the secret symmetric key known to both communicating parties. While asymmetric public key

algorithms such as RSA [33] do not require knowledge of common secret key by communicating parties, they are considered slow. In most practical applications of cryptography, like SSL enabled communication, an asymmetric public key algorithm is used once only to generate a symmetric secret key, which encrypts all traffic with a symmetric algorithm.

5.1.1.3. SSL implementation in P2PRM

For use in P2PRM prototype, we generate the keys and certificates using a certificate management utility from Sun called *keytool* [27]. We use the RSA algorithm [33] to generate 1024-bit key pairs for use in SSP certificates and SSL handshaking. The Advanced Encryption Standard (AES) [19] is used to encrypt all SSL communication. We have used Java Secure Sockets Extension (JSSE) [18] from Sun to implement SSL in Java. JSSE uses the term keystore to signify a repository of key information (including public-private keys and certificates belonging to an entity). In our system, the SSP stores its key information and certificate in its keystore. A truststore is a repository an entity consults when making a decision to trust some

other entity. The client has a truststore with entries for the certification authorities (CAs) that signed the SSP certificate. For a client to successfully authenticate the SSP, the chain of certification authorities belonging to the SSP certificate should be a part of client's truststore. We used self-signed certificates by the SSP in our prototype development. To emulate the chain of CA's in SSP certificate becoming a part of client truststore, we installed the SSP public key in the client truststore.

Below we summarize the setup steps needed in the P2PRM prototype before SSL handshaking can be performed for SSP authentication and secret key agreement:

- Generate a pair of asymmetric keys consisting of SSP private key and SSP public key.
- Generate a SSP certificate with the SSP public key.
- Store the private key and public certificate in a secure keystore.
- Extract the server public key from the keystore.
- Export the server public key to the client's truststore

Note that these steps are only one-time setup task for a SSP. Each instance of communication between the SSP and

client during wrapper creation and access can use the same SSP certificate with the same SSP public key for SSL handshaking. However, SSL handshaking in each communication instance will culminate in a different encryption secret key valid only during the session it is generated in. Session specific secret keys reduce the window of any attack emanating from key compromise during a specific session.

5.1.2. Master key generation and encryption

The SSP generates a symmetric key for use as a master key to encrypt all sensitive information to be included in the content wrapper. The master key is known only to the SSP. Any robust cryptographic algorithm will serve the purpose of encryption with the master key. We chose the Advanced Encryption Standard (AES) [19] as the algorithm for master key encryption and decryption function. AES has evolved into a standard cryptographic algorithm. With appropriate key lengths, it is perceived sufficiently secure, due to which it is used even in national security systems. The SSP generates an AES key and stores it securely for use in all master key encryption and decryption functions. An

important assumption in the security of P2PRM is that the SSP maintains strict confidentiality on its master key to prevent any compromise of content or other sensitive information.

Sensitive information encrypted with the master key of the SSP include the decryption key of underlying content, content price, creator/reseller markup and creator/reseller identifier. This sensitive information could be tampered with by an attacker in the following ways, thereby necessitating their protection by the SSP:

- Decryption key – An attacker could maliciously retrieve the decryption key of the underlying content if the key is not sufficiently secured to gain unauthorized access to content. He could even tamper the decryption key to prevent a buyer from accessing the content.
- Price – An attacker could reduce the price included in the wrapper to decrease the profits of the creator or could increase the price to prevent large scale selling of content.
- Creator/reseller identifier – An attacker can maliciously change the creator or reseller identifier

included in the wrapper to his own identifier to obtain undue profits, which belong to the legitimate creator or reseller.

- Creator/Reseller markup – An attacker could maliciously change the markup of both creator and reseller to deny them of potential profits.

Encryption with the master key of the SSP prevents the above-mentioned attacks. By using master key encryption coupled with SSL, SSP securely tracks all the transactions and sensitive billing information rather than each peer tracking all their transactions. Decentralized tracking is more prone to vulnerabilities since each peer needs to be trusted, which is not a reasonable assumption. Master key security feature is critical to accurate billing and profit allocation since only the SSP can decrypt the information to charge and credit the responsible parties with the correct amounts. In this prototype, we have not implemented a full-blown billing mechanism at the SSP that takes care of accounting (credit card verification, charging accounts, maintaining transaction logs, charging creators and resellers, distributing profits to creators and resellers).

5.2. Security features in wrapper creator software

5.2.1. Content encryption

The wrapper creator software encrypts all content before packaging it into the content wrapper to be distributed. For each piece of content to be packaged, the wrapper creator software generates an AES key. We encrypt the content using AES encryption for the same reasons we chose AES for master key encryption and decryption by the SSP. AES is deemed sufficiently secure against most types of attacks. Its resilience is the reason behind its use to protect classified information by the government. In P2PRM, the AES key is generated and the content is encrypted without storing the key in the creator’s terminal as an extra measure of security against key compromise at the source itself.

5.2.2. License with content

The wrapper creator software stores the AES decryption key required to decrypt the content in the content wrapper itself. To protect the content decryption key against unauthorized use, the content decryption key is encrypted with the master key of the SSP. The SSP decrypts the

content decryption key only after the buyer makes the desired payment. It is difficult for any attacker to retrieve the content decryption key from the content wrapper without payment unless he can break the master key encryption algorithm or obtains the master key. While our choice of AES as master key algorithm renders it sufficiently secure due to its ability to withstand most attacks, we assume that SSP securely stores the master key. It is paramount that the master key of the SSP be strictly confidential to prevent any compromise of sensitive information such as the content decryption key. Any successful attack to retrieve the master key is a single point of failure for the system.

We also wanted to include a mechanism by which if a determined hacker successfully compromises a content wrapper to recover the underlying content, other wrappers could not be attacked using the same attack approach to prevent large-scale tampering. This leads us to the next security feature offered by the wrapper creator software.

5.2.3. Break-once, break everywhere (BOBE) resistance

Our design strives to achieve resistance to BOBE attacks by scrambling the content before encryption. This adds an extra layer of security in retrieving unencrypted content from content wrappers. If an attacker successfully breaks the standards-based encryption protecting the content by reverse engineering, he still has to break the proprietary scrambling algorithm before recovering the actual content. While any cryptographic algorithm is universally available (the key is the only secret), proprietary algorithms are unknown to attackers and will add a degree of difficulty in code reverse engineering to recover content.

Any proprietary scrambling algorithm could be applied in P2PRM. We have created variants of the TEA (Tiny Encryption Algorithm) [20] for use as our scrambling algorithm. Different variations of TEA were created to achieve individualization of content wrappers. We do not rely on the scrambling algorithm for cryptographic strength, but only for its ability to obscure underlying code with effectiveness and simplicity.

5.2.4. Secure connection to SSP

The wrapper creator software needs to communicate securely with the SSP for master key encryption of sensitive information. Rather than sending this information in the clear, the wrapper creator initiates a SSL channel with the SSP, authenticates the SSP and establishes a secret symmetric key with the SSP (as described in the section on SSP security features). All communication between the wrapper creator and SSP is protected with this symmetric key. This prevents any attacker from snooping data or maliciously altering data exchanged in the client-SSP interaction.

Having discussed the general implementation aspects and security specific implementation features of the wrapper creator software, we present the following diagram to summarize the steps executed by the wrapper creator software to create the content wrapper. At each step, we have included the technology used to implement the function.

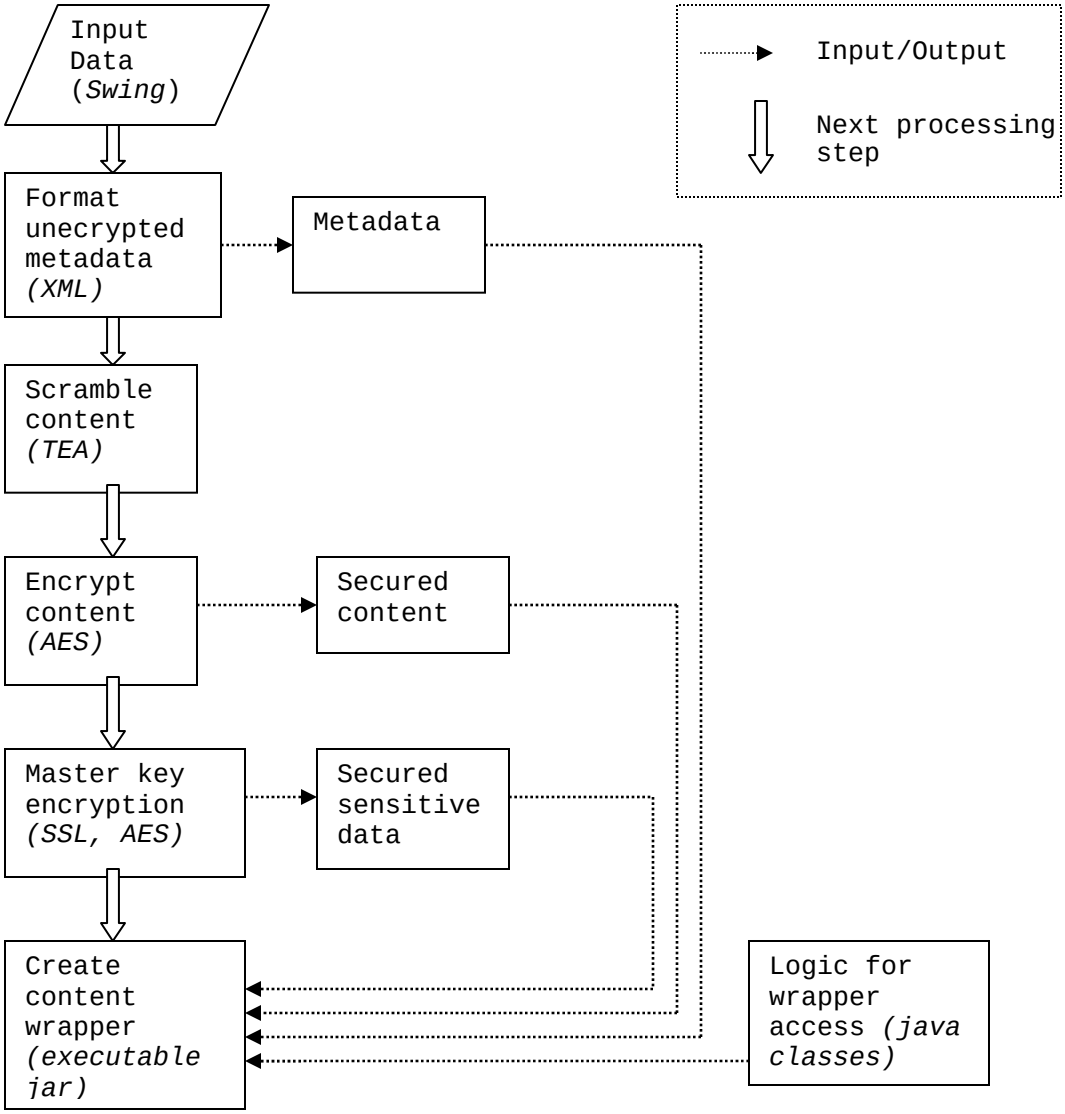


Figure 9. Steps in wrapper creation with implementation technology

Executing the steps discussed above encapsulates the content in a content wrapper. The underlying content is secured by the layers of protection applied by the security

features provided by the SSP and the wrapper creator software. Each layer of protection adds a measure of security, translating into more work required by an attacker to recover content. We present the following diagram illustrating the levels of security features protecting underlying content in P2PRM.

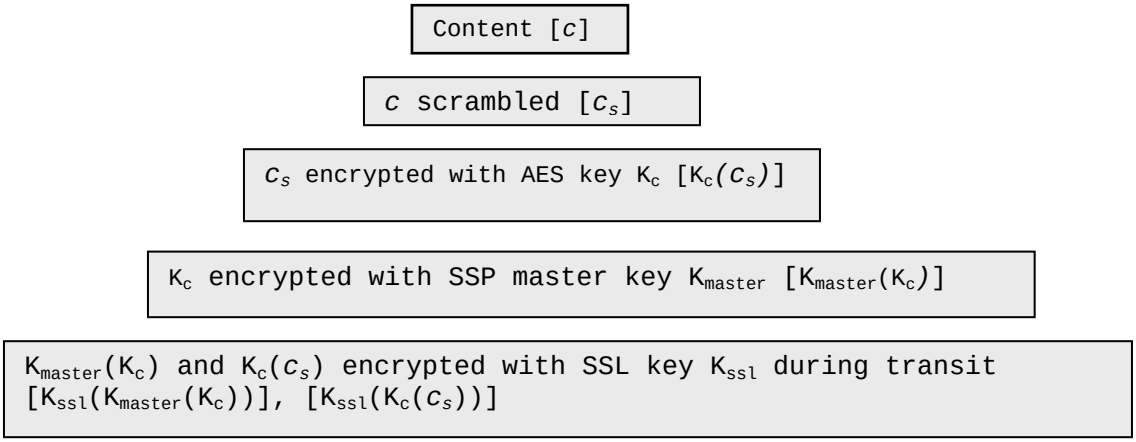


Figure 10. Levels of protection applied to content

5.3. Security features in wrapper access software

5.3.1. Unauthorized access

Even if a peer receives a content wrapper from another peer, the embedded wrapper access software will ask for payment at the SSP before the peer can get the decryption

key and access the content. This ensures that all users of content have made the desired payment before accessing the content.

5.3.2. Hard disk storage

The wrapper access software never stores the decrypted content on the hard drive of any peer terminal. It directly feeds the decrypted content stream to the underlying content reader application. Although this is subject to a fairly trivial attack, there is no way to avoid it in such a “lightweight” implementation (would need NGSCB [26] or a comparable heavy-duty approach for real protection on a client terminal).

5.3.3. Secure connection to SSP

Wrapper access software sends buyer’s payment information, in addition to sending the master key encrypted decryption key to the SSP. Once the SSP processes the payment, it decrypts the content decryption key and sends it back to the wrapper access software. Both payment information and decryption key should not be transmitted in the clear. To

protect this information, the wrapper access software is capable of communicating to the SSP through a secure SSL channel.

6. TESTING

P2PRM introduces a new way by which digital content can be distributed in a P2P network. This project aimed to introduce a new system design through new functionalities. Due to the facts that the focus of implementation was achieving the conceived system functionality, and that we are not aware of any comparable system, a performance analysis or performance comparison in this project is not feasible. The cornerstone of our testing was verifying that the implementation realizes the design objectives and the system works as designed.

We tested the functionality of all the components constituting P2PRM. For our testing, we used audio files to emulate digital content to be distributed in the P2P network. In this section, we discuss a sample test case that we used to verify the implementation and demonstrate the functionality of the developed prototype. We illustrate all the steps involved in creating, selling, accessing and reselling a content wrapper sequentially. For purposes of clarity, we have shown the user-interfaces associated with a given step, if any.

Creator registration

- Start the SSP.

- Click on Register button. Enter all the details.

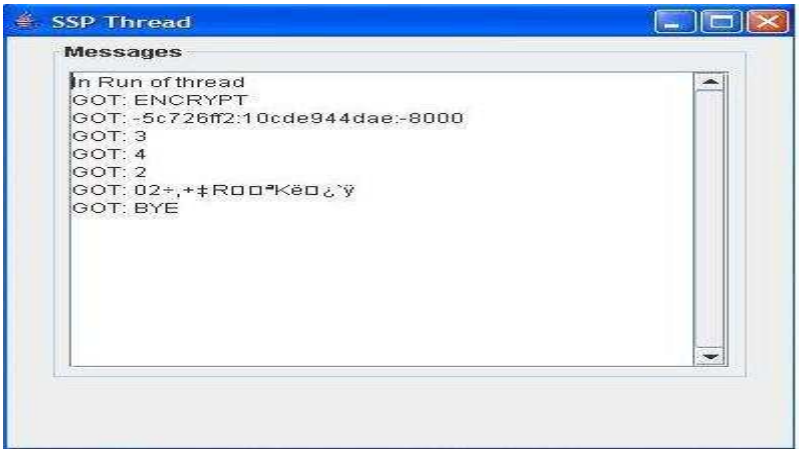
- Click *Save & Generate ID* button. A confirmation window appears.

Verify that an ID file (*pallavi.UID*) is generated in the working directory with the correct content.

User wrapper creation

- Start the wrapper creator software. Enter all the details as follows:

- Click *User Wrapper* button. This creates a secure communication channel with the SSP. SSP spawns a separate thread to service the connection.



- Once the creation of the user wrapper is complete, a confirmation window appears. Verify that a user wrapper is created in the working directory (*music1_wrapper_user.jar*).

User wrapper access

- Copy the generated user wrapper in a test directory and double-click the user wrapper, which launches the user wrapper access software. Double-clicking also extracts all the files needed by the wrapper access software from the content wrapper into the current directory.

  Verify the contents of the XML information file contained in the wrapper.

  Verify that the information displayed by the wrapper access software was correct. Enter all the details required to purchase access rights to the wrapper as follows:

- Click on the *Purchase* button from the wrapper access software. This creates a secure communication channel with the SSP, which spawns a separate thread to service the wrapper access connection.

- Once the SSP and wrapper access software complete their transaction, the *Play* button in the wrapper access software is activated. Click on the *Play* button to play the decrypted music stream.

During the time music is being played, the *Stop* button is activated in the wrapper access software to allow canceling of current music at any time.

Reseller wrapper access

- Once a reseller wrapper (*music1.wrapper_res.jar*) is created, for which the steps are similar to creating a user wrapper, double-click on the reseller wrapper

after copying it to a test directory. This launches the reseller wrapper access software. Verify that the information displayed by the software is correct (for eg., reseller price). For purchasing the wrapper, follow the same steps as that for purchasing a user wrapper.

- Once the transaction between the SSP and wrapper access software is completed, the *Play* button is activated for us to click on it and play the audio file. Not only that, the *Repackage* button is also activated. This means that we can now repackage the underlying content.

- Click on the *Repackage* button. Enter the reseller markup in the window displayed.

- A confirmation window appears when the repackaging is complete. Verify that a user wrapper is created in the current directory.
- Launch the user wrapper software for the user wrapper generated by the reseller to verify that while the underlying content of the newly created user wrapper is the same as the original reseller wrapper, the price is different which is determined by the content creator’s markup (2 in this test case) and the reseller’s markup (1 in this test case).

7. CONCLUSIONS AND FUTURE WORK

We have proposed a model and successfully developed a working prototype for legal and reasonably secure content distribution for P2P networks. P2PRM includes many features to deter piracy and enable content creators and distributors to get profits. Some of these features are security related, such as Break-Once Break-Everywhere (BOBE) resistance, while others are more general, like reselling. The novel features included make the system less susceptible to attacks and more conducive to legitimate users. For example, through the reselling model of P2PRM, resellers can also profit through content distribution. This makes the resellers less likely to indulge in piracy as they can potentially make a profit.

Although the embedded features make P2PRM reasonably secure and a candidate suitable for large-scale deployment over existing P2P networks, no system can be deemed totally secure unless it withstands the scrutiny of hackers and researchers over an extended period of time. While we do not claim that the security of the system is impregnable, we do believe that each security feature adds a useful

layer of security requiring more effort by an attacker. Consequently, we believe that all of the features together render the system reasonably secure. In case determined hackers do succeed in compromising individual pieces of content (which is inevitable), the BOBE resistance feature should enable the system to continue surviving as a whole.

Our basic P2PRM model opens up several avenues of possible extensions. Here we outline some more features that could be added to P2PRM. Note that the following list is not exhaustive, and there could be other possible extensions.

- *Advanced payment system integration* – P2PRM can be integrated with advanced payment systems such as:
  a) Micro-payment system – Digital goods could be offered for a very low price (example, 50 cents). Even the level of access could be determined by the level of payment. For example, the seller could require that the buyer pay five cents to view each page. Both low price and level of access can be solved by adding on a micro-payment capability [24, 37, 38] to P2PRM, which allows the fees for one piece of content to be extremely low. The buyer could pay the fees for one piece

of content or could pay an accumulated larger sum of money.

b) Subscription based – Another possible approach for implementing advanced payment capability in P2PRM is that any potential buyer could pay fixed amount of money monthly to purchase access rights to a fixed number of contents (or number of accesses allowed) from a particular content creator. This would decrease the payment related traffic in the system.

- *Role-based Access Control* – We could incorporate role-based access [25, 39] in the system whereby all users could be classified by their roles in the system and access decisions could be based on the roles users are assigned to. Roles in the system could be granted by the SSP or the creator based on the level of payment received from buyers. More advanced level buyers are granted more access rights in the system (like more number of times he can access or more number of copies he could make). This role-based access control could be combined with a subscription based payment strategy to significantly reduce access and payment related traffic in the system.

- *Advanced DRM capability* – P2PRM could include a finer granularity of access control techniques. A creator could specify the number of pages he wants a particular buyer to be able to read. Alternatively, a creator could specify how many times can a reseller can repackage and distribute a piece of content.
- *Email support* - SSP could send identifier files to creator and reseller through email. This would create stronger form of authentication (so-called two-factor authentication) by adding an extra layer of security since creator (or even reseller) has to be able to check his email to obtain the ID file.
- *Content identification techniques like audio fingerprinting* – Implementing content identification techniques will ensure that an attacker cannot compromise a piece of content, treat the content as his own, and go on to collect profits by packaging and selling the content as his own.
- *Profit through content hosting* – In order to further reduce the risk of content piracy, an additional feature could be included in the system whereby even a person hosting content wrappers at his terminal is entitled for a share of profits when the wrapper is

downloaded from his terminal and purchased by the downloader. This will give more incentives to peers to be a part of the system, rather than attacking it.

- *Integration with Next-Generation Secure Computing Base (NGSCB)* - P2PRM could utilize the inherent security offered by NGSCB [26] through use of protected process modes dedicated to secure media-access related processes and safe storage of cryptographic keys. Integration with NGSCB will offer a high degree of security when the content wrapper resides at a buyer's terminal or when a buyer accesses decrypted content.

APPENDIX A: REFERENCES

[1] How P2P and Kazaa software works. Retrieved January 2006 from http://www.kazaa.com/us/help/new_p2p.htm.

[2] Gnutella clients. Retrieved January 2006 from http://www.gnutella.com/connect.

[3] How the old Napster worked. Retrieved January 2006 from http://computer.howstuffworks.com/napster.htm.

[4] Distributed Computing Industry Association. (2004). Proposed business models for digital music distribution. Retrieved January 2006 from http://www.dcia.info/model.ppt.

[5] iPod + iTunes. Retrieved March 2006 from http://www.apple.com/itunes/.

[6] iTunes: 1 billion served. Retrieved February 2006 from http://abcnews.go.com/Technology/story?id=1653881&technology=true

[7] Stamp, M. (2006). Information Security Principles and Practice, *John Wiley and Sons.*

[8] Einhorn, M., A., & Rosenblatt, B. (2005). Peer-to-Peer networking and digital rights management. Retrieved January 2006 from http://www.cato.org/pub_display.php?pub_id=3670.

[9] Rosenblatt, B. (2003). Integrating DRM with peer-to peer networks. Retrieved January 2006 from http://www.giantstepsmts.com/drm_p2p.htm.

[10] Kalker, T., Epema, D., H., J., Hartel, P., H., Lagendijk, R., L., & Steen, M., V. (2004). Music2Share – Copyright-compliant music sharing in P2P systems. *Proceedings of the IEEE.*

[11] Berket, K., Essiari, A., & Muratas, A. (2004). PKI-based security for peer-to-peer information sharing. *Proceedings of the Fourth International Conference on Peer-to-Peer Computing.*

[12] Iwata, T., Takehito, A., Ueda, K., & Sunaga, H. (2003). A DRM system suitable for P2P content delivery and the study on its implementation. *Proceeding of the 9th Asia-Pacific Conference on Communications (APCC 2003.*

[13] Judge, P., & Ammar, M. (2003). CITADEL: A content protection architecture for decentralized peer-to-peer file sharing systems. *Proceedings of IEEE GLOBECOM 2003.*

[14] Reti, T., & Sarvas R. (2004). DiMaS: Distributing Multimedia on Peer-to-Peer File Sharing Networks. *Proceedings of ACM Multimedia*.

[15] Gehrke, N., & Schumann, M. (2002). Constructing electronic marketplaces using peer-to-peer technology. *Proceedings of the 36th Hawaii International Conference on System Sciences.*

[16] Wierzbicki, A. (2005). Peer-to-peer direct sales. *Proceedings of the Fifth IEEE International Conference son Peer-to-Peer Computing.*

[17] Michiels, S., Verslype, K., Joosen, W., & Decker, B., D. (2005). Towards a software architecture for DRM.

Proceedings of the Fifth ACM Workshop on Digital Rights Management.

[18] Java Secure Sockets Extension (JSSE). Retrieved January 2006 from http://java.sun.com/products/jsse/.

[19] Advanced Encryption Standard. Retrieved March 2006 from http://csrc.nist.gov/CryptoToolkit/aes/.

[20] Wheeler, D., & Needham, R. (1994). Tea, a tiny encryption algorithm. *Fast Software Encryption pp 363-366.*

[21] Extensible Markup Language (XML). Retrieved April 2006 from http://www.w3.org/XML/.

[22] Project Swing (Java Foundation Classes). Retrieved April 2006 from http://java.sun.com/j2se/1.5.0/docs/guide/swing/.

[23] Violet. Retrieved October 2006 from http://horstmann.com/violet/.

[24] Common markup for micropayment per-fee-links. Retrieved October 2006 from http://www.w3.org/TR/Micropayment-Markup/#Basic.

[25] Ferraiolo, D., F., & Kuhn, D., R. (1992). Role Based Access Control. *15th National Computer Security Conference*.

[26] Next Generation Secure Computing Base (NGSCB). Retrieved October 2006 from http://www.microsoft.com/resources/ngscb/default.mspx

[27] keytool – Key and Certificate Management Tool. Retrieved March 2006 from http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html.

[28] Fairplay. Retrieved February 2006 from http://en.wikipedia.org/wiki/FairPlay.

[29] Class UID. Retrieved May 2006 from http://java.sun.com/j2se/1.4.2/docs/api/java/rmi/server/UID.html.

[30] Model-View-Controller. Retrieved March 2006 from http://ootips.org/mvc-pattern.html.

[31] Packaging programs in jar files. Retrieved April 2006 from http://java.sun.com/docs/books/tutorial/deployment/TOC.html#jar.

[32]Java Secure Sockets Extension (JSSE) Reference Implementation. Retrieved March 2006 from http://java.sun.com/j2se/1.4.2/docs/guide/security/jsse/JSSERefGuide.html#SSLOverview.

[33] RSA. Retrieved February 2006 from http://en.wikipedia.org/wiki/RSA.

[34] Youtube: Broadcast yourself. Retrieved October 2006 from http://www.youtube.com.

[35] Gnutella. Retrieved January 2006 from http://en.wikipedia.org/wiki/Gnutella.

[36] Biddel, P., England, P. , Peinado, M.,& Willman, B. (2002). The Darknet and the Future of Content Distribution. Microsoft Corporation. Digital Rights Management Conference.

[37] Dai, J. (2006). QuickPay: Protocol for online payment. Graduate Project. Retrieved October 2006 from http://www.cs.sjsu.edu/faculty/stamp/students/QuickPay Report.pdf.

[38] Stamp, M., & Dai, J. (2006). Micropayment token making schemes. Retrieved October 2006 from http://www.cs.sjsu.edu/faculty/stamp/students/dai.htm.

[39] Stamp, M., Mathur, A., & Kim, S. (2006). Role based access control and the JXTA peer-to-peer network. Proceedings of 2006 International Conference on Security & Management.