

ANALYSIS AND ENHANCEMENT OF APPLE'S FAIRPLAY DIGITAL
RIGHTS MANAGEMENT

A Project Report

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science, Computer Science

by

Ramya Venkataramu

May 2007

© 2007

Ramya Venkataramu

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Mark Stamp, Department of Computer Science, SJSU

Dr. Teng Moh, Department of Computer Science, SJSU

Dr. Robert Chun, Department of Computer Science, SJSU

APPROVED FOR THE UNIVERSITY

ABSTRACT

ANALYSIS AND ENHANCEMENT OF APPLE'S FAIRPLAY DIGITAL RIGHTS MANAGEMENT

by Ramya Venkataramu

Digital Rights Management (DRM) technology is used to control users' access to copyrighted digital content. Apple Inc.'s DRM system, called Fairplay, is used by the iTunes music store to place restrictions on the use of digital content purchased from the online store. Users communicate with the centralized iTunes server to purchase, play, preview digital content, etc.

The existing iTunes music store has the disadvantage of a bandwidth bottleneck at the centralized server. Furthermore, this bandwidth bottleneck problem will escalate with increasing popularity of online music and other digital media (such as movies and TV shows) downloads.

This project studies the Fairplay DRM used by iTunes (version 5.0 and version 6.0) and enhances it to be used over a peer-to-peer (P2P) network. The aim is to garner all the benefits of a decentralized P2P network while still providing DRM content protection similar to Fairplay.

TABLE OF CONTENTS

1	Introduction.....	1
2	Background	4
2.1	Apple's Fairplay DRM	4
2.1.1	Fairplay DRM Restrictions.....	4
2.2	MPEG-4	5
2.2.1	Structure of MPEG-4 Files	6
2.3	The Fairplay DRM Design.....	8
2.3.1	Encryption Scheme	9
2.3.2	Purchasing a Song from the iTunes Store.....	9
2.3.3	Playing a Purchased Song using iTunes	10
2.3.4	Watermarking in Fairplay	11
2.4	Reverse Engineering iTunes	13
2.5	Introduction to P2P Networks.....	15
2.6	Security Aspects of Sharing over P2P	17
2.7	Goals of this Project.....	18
2.8	Existing Research.....	18
3	Design.....	20
3.1	A Hybrid Approach.....	20
3.2	Entities of P2PTunes.....	21
3.3	Architecture of P2PTunes	21
3.3.1	Description of the iTunes Metadata.....	22
3.3.2	Steps Involved in Purchasing and Playing a Song.....	23
3.4	Benefits of the Proposed System	28
3.5	Potential Issues.....	29
3.6	Functional Architecture	30
4	Implementation	32
4.1	Underlying P2P Network.....	32
4.2	P2PTunes - Class Diagrams.....	33
4.2.1	Originator	34
4.2.2	Responder	36
4.2.3	P2PTunesServer (PTS)	38
5	Security	39
5.1	Security Analysis	39
6	Testing.....	44
6.1	Creation of a façade P2P Network.....	45
6.2	Client Browser Windows.....	46
6.3	Purchasing a Song.....	47
6.4	Playing a Purchased Song.....	53
7	Conclusion	55
8	Future Work.....	57
	References.....	59

INDEX OF FIGURES

Figure 1: MPEG-4 Components	5
Figure 2: MPEG-4 File Structure (Anonymous, 2005)	6
Figure 3: Protected AAC File Structure (Anonymous, 2005)	7
Figure 4: iTunes Protocol for Purchasing and Playing a Song	12
Figure 5: JHymn GUI Interface	14
Figure 6: Key Atoms in an m4p File	22
Figure 7: A Sample P2PTunes Network	23
Figure 8: P2PTunes Functional Architecture.....	31
Figure 9: Originator Class Diagram.....	35
Figure 10: Responder Class Diagram	37
Figure 11: PTS Class Diagram	38
Figure 12: Atoms in a Song	44
Figure 13: FacadeP2P Creation GUI	46
Figure 14: P2PTunes client UI.....	47
Figure 15: Request a Song through P2PTunes	48
Figure 16: Responses for Content.....	48
Figure 17: Details of Content.....	49
Figure 18: Request Payment for Downloaded Content	50
Figure 19: Transaction Success Message	50
Figure 20: Transaction Failure Message.....	51
Figure 21 : Download Error Message.....	51
Figure 22 : Integrity Check Error Message.....	51
Figure 23 : Payment Required to Play Content	52
Figure 24 : Downloaded Content on User's System.....	53
Figure 25 : Message indicates Song Playing	54

INDEX OF TABLES

Table 1: Metadata Information	8
-------------------------------------	---

1 Introduction

The success of Apple's iTunes music store, together with its iPod music range has made Apple Inc. a dominant company in both the online media store and digital media player markets (Chandak, 2005). The iPod is a portable digital media player, designed by Apple Inc., which supports the Advanced Audio Coding (AAC), MP3 or Moving Pictures Experts Group (MPEG-1) Audio Layer-3, Waveform Audio (WAV), and Audible formats (Apple Inc., 2007). The iTunes online store allows users to purchase digital media content. A proprietary software application, called iTunes, is used to connect to the iTunes online store to download digital content. The iTunes software is used to manage play lists, share content with different computers, and play digital content on Windows computers, Macintosh computers, and the iPod.

Fairplay is a digital right management (DRM) technology used to protect digital content purchased from the iTunes online store. Fairplay places restrictions on purchased digital content to restrict uses of copyrighted content. Users of purchased digital content are restricted by providing persistent protection, i.e., protection that stays with the digital content after it has been delivered to the user (Stamp, 2006).

The iTunes online store and Fairplay DRM employ a centralized server to distribute content and enforce copyright on downloaded media. Such centralized server based content distribution has the disadvantage of a bandwidth bottleneck at the central server(s). Furthermore, as the number of users accessing the online store grows and the size of digital content increases (movies require much more bandwidth than music) additional strain is placed on the central server (Kalker, 2004).

The centralized iTunes online music service may be better organized as a peer-to-peer (P2P) network from the viewpoint of efficient storage and use of bandwidth (Kalker, 2004). A P2P network is a distributed system that can harness idle storage and network resources from client machines that voluntarily join the network (Rodrigues, 2002). Each workstation or node has equivalent capabilities and can initiate or service requests. This may be contrasted to the client-server based iTunes where only the iTunes server services requests.

P2P systems have emerged as a popular way to share huge amounts of data since they offer the benefits of self-organization, load-balancing, fault-tolerance, and the ability to pool together and harness large amounts of resources (Daswani, 2003). Additionally, P2P networks are quite scalable and easy to deploy (Tanin, 2005). Unfortunately, distributing digital content over existing P2P networks is rife with violation of copyrights and other security risks such as viruses, spyware, and unwanted software (Microsoft Corp., 2006).

This project is focused on critically analyzing the existing iTunes distribution system and scaling it for future needs. The aim is to integrate the existing iTunes with the strong points of a P2P system thus developing a practical and feasible solution. Adequate DRM support will be incorporated in an effort to protect copyright owners, benefit legitimate customers, and limit illegal use of copyright content.

This report is organized as follows:

- Section 2 describes the background of Apple's Fairplay DRM, the existing iTunes design and presents an overview of P2P networks.

- Section 3 describes our proposed design and architecture.
- Section 4 presents a prototype implementation.
- Section 5 deals with security features of the proposed system. We analyze the strengths and weaknesses of the proposed system.
- Section 6 details the testing of the prototype system by following a step-by-step approach through the use of sample test cases.
- Section 7 draws conclusions of the project.
- Section 8 presents additional enhancements that could be used to extend the proposed system.

2 Background

2.1 Apple's Fairplay DRM

The Fairplay DRM, built into the QuickTime multimedia technology, is used to protect digital content purchased from the iTunes online music store. Fairplay has several strong security features. For instance, purchased files are encrypted to prevent copyrighted content from being misused and digital watermarking is used to embed information into the purchased file. However, this software-based DRM does have numerous restrictions and limitations as described in the following section.

2.1.1 Fairplay DRM Restrictions

Some of the restrictions that Fairplay attempts to enforce include:

- Protected tracks can be copied to a certain number of “authorized” computers.
- A protected track may be burned into a play list certain number of times.
- Protected tracks may be burned into an audio CD a certain number of times.
- Purchased digital content can be played on Apple's iPod alone and not any other non-Apple digital music devices.
- Fairplay limits usage of its digital content to the Windows and Macintosh Operating Systems. Downloaded media is not playable on other major operating systems such as Linux.

- Most audio-editing software used for editing or splicing tracks are not interoperable with iTunes purchases.

These restrictions have angered many users who believe purchased content should be free to use in any legal manner without undue restrictions (Futureproof, 2006).

2.2 MPEG-4

The QuickTime file format is a “container” that can handle audio, video, images, text and other digital formats (Apple Inc., 2006). QuickTime is adaptable: new capabilities can be added and new versions maintain backward compatibility (Apple Inc., 2006).

QuickTime is the file format of choice for Moving Pictures Experts Group (MPEG-4). The MPEG-4 standard covers the entire digital media task of capturing, authoring, editing, encoding, distributing, playback, archiving, and delivering professional-quality digital media. MPEG-4 inherits QuickTime’s stability, extensibility, and scalability (Apple Inc., 2006). Figure 1 illustrates MPEG-4 components.

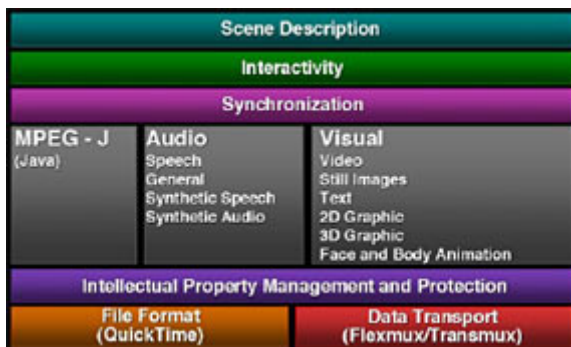


Figure 1: MPEG-4 Components

Source: Apple Computers Inc., MPEG-4: The Container for Digital Media, 2006

Advanced Audio Coding (AAC) is the audio layer in the MPEG-4 files that compresses audio data more efficiently than older formats such as MP3 (Apple Inc., 2006). Apple uses Fairplay to encrypt the AAC or audio data inside an MPEG-4 file resulting in what is known as a protected AAC files. Protected files carry an “m4p” extension.

2.2.1 Structure of MPEG-4 Files

MPEG-4 files are built up of *atoms* each of which store specific information pertaining to the digital content. Every atom has an 8-byte header, an 8-byte field indicating the atom type, followed by the data field. The atom type indicates how to process the atom data. An MPEG-4 file structure is illustrated in Figure 2.

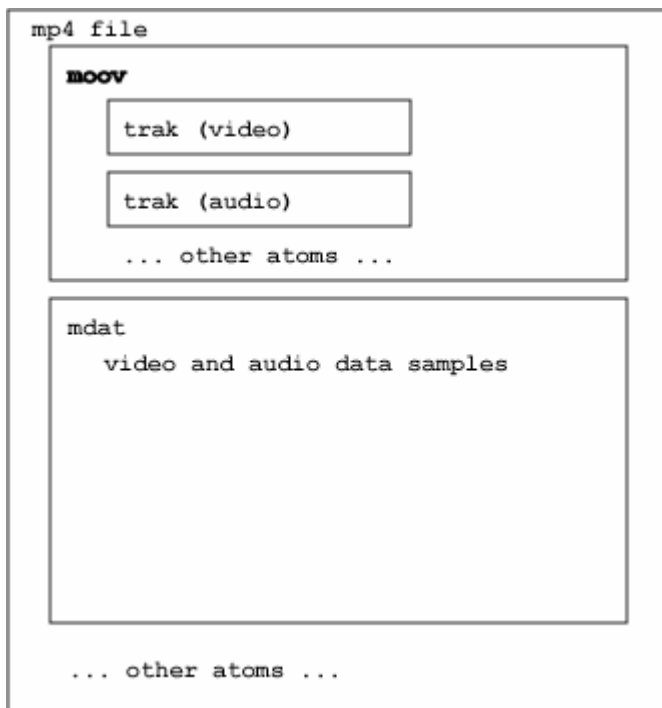


Figure 2: MPEG-4 File Structure (Anonymous, 2005)

The song in the AAC layer is encrypted using the AES (Rijndael) encryption algorithm which is a published standard (Anonymous, 2005). Atom types are depicted in Figure 3. For clarity, some atoms have been omitted in Figure 3.

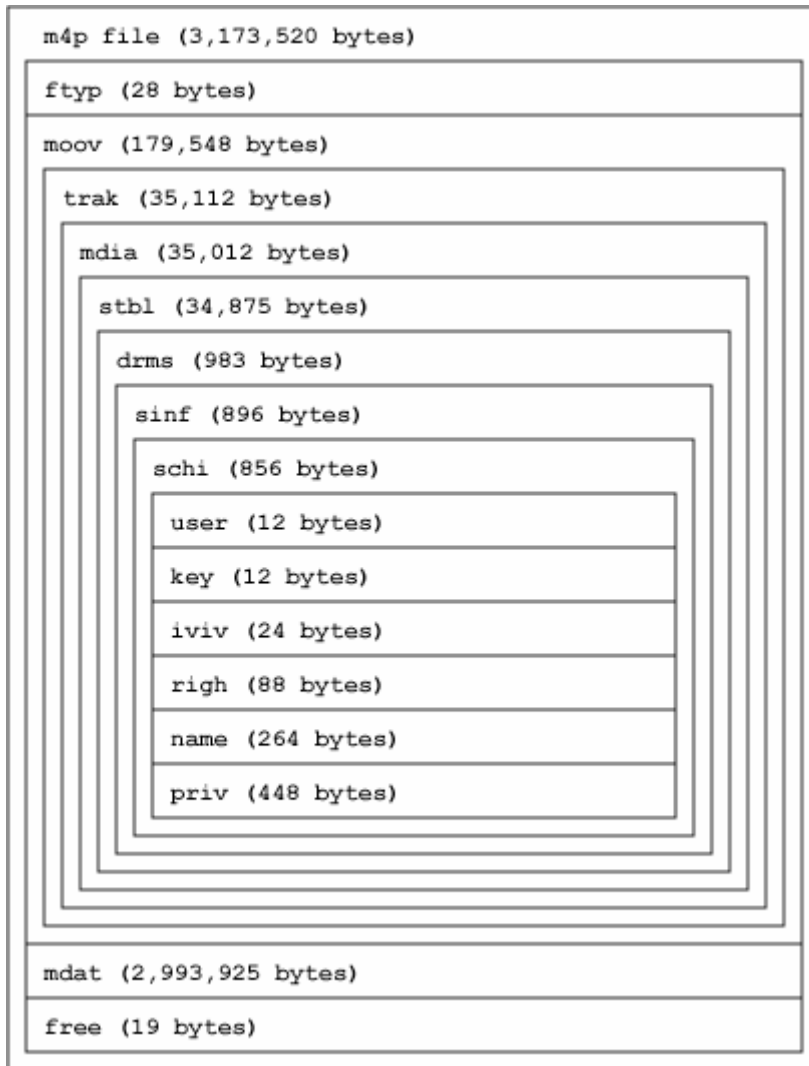


Figure 3: Protected AAC File Structure (Anonymous, 2005)

Atoms generally present in a protected file are presented in Table 1 (Anonymous, 2005).

Table 1: Metadata Information

Atom Name	Atom Data
mdat	stores the encrypted song
drms	stores information about the song, identity of the purchaser, and cryptographic information needed to decrypt the song.
user	stores the iTunes user ID
key	stores the iTunes user key number
iviv	stores the AES initialization vector
name	stores the iTunes user name
geID	stores watermarking information

The main difference between the protected file format discussed above and an unprotected file format is that DRM specific atoms such as drms, user, geID, priv, name, etc., are absent in unprotected files. Unprotected file carry an “m4a” extension.

Atomic Parsley is a lightweight command line tool that can be used to view the differences between protected and unprotected files and to parse and set metadata atoms in MPEG-4 files (SourceForge, 2006).

2.3 The Fairplay DRM Design

This section describes the Fairplay design used in iTunes version 5.

Fairplay DRM has three levels of encryption and uses different keys. The different types of keys used by iTunes are explained in this section.

A *system key* is a shared symmetric key that is used by the iTunes server to encrypt user keys. It is generated on a Windows system as a hash of items from the registry including Bios Version, Processor name, and Windows Version (Anonymous, 2005). The system key hash for Macintosh machines has not been reverse engineered (Anonymous, 2005).

A user key database stored on the iTunes server contains user keys that are needed in the decryption process. Apple uses a few user keys per iTunes music store account (Anonymous, 2005). This implies that different media purchased by one user might use the same user key for the decryption process. The encryption scheme is explained in detail in the following subsection.

2.3.1 Encryption Scheme

The AAC audio data is encrypted with an AES key using the AES (Rijindael) algorithm. This encrypted AAC audio data forms the *mdat atom*. Further, the AES key used in this encryption is encrypted with a user key (stored on the iTunes server database) and stored in the *priv atom*. The user key is encrypted with the system key while being transferred from server to client. The scheme used in purchasing and playing content is explained in the following subsections.

2.3.2 Purchasing a Song from the iTunes Store

The following steps detail an iTunes client purchasing a song from the online music store (Anonymous, 2005). Figure 4 illustrates the interaction between the client and server.

- 1) The user chooses a song from the iTunes online music store and makes a download/purchase request to the iTunes server.
- 2) The iTunes client sends the song download request and user's system information to the iTunes server.
- 3) The iTunes server sends the download URL and a download key to the iTunes client.
- 4) The iTunes client downloads the file from download URL and decrypts the file using the download key. This decrypted file is the protected song and is stored on the client computer.
- 5) The client sends a message to the server indicating success of the transaction.

2.3.3 Playing a Purchased Song using iTunes

The following steps detail an iTunes client playing a purchased song (Anonymous, 2005). Figure 4 illustrates the interaction between the client and server.

- 1) The user ID and the user key index from the protected (m4p) file at the client are sent to the iTunes server along with the system information.
- 2) The iTunes server uses the user ID and the user key index to retrieve the user key from its key database. It encrypts the user key using a system key generated from the system information and sends this encrypted key to the client.
- 3) Upon receiving this key, the client decrypts it using the system key to get user key.
- 4) The client hashes the name and iviv atoms of that specific m4p file to obtain an initialization value.

- 5) The key from step 3 and initialization value from step 4 are used to decrypt the priv atom which retrieves the AES key.
- 6) The key from step 5 and the initialization value is used to decrypt the mdat atom, which gives the audio stream that will be played.

2.3.4 Watermarking in Fairplay

Apple inserts some watermarking in protected files as an indicator of legitimate content. Apple's iTunes software looks for these watermarking indicators to verify the authenticity of the digital content. Tampered files, which do not have watermarking features, are rendered "unplayable" on iTunes software. However, such content can be played on any AAC compatible hardware or software which does not look for Apple's watermarking (Wen, 2005). Additional watermarks are cached outside the protected file, in the iTunes library database, and on the iPod to make it harder for reverse-engineering tools to detect (Wen, 2005).

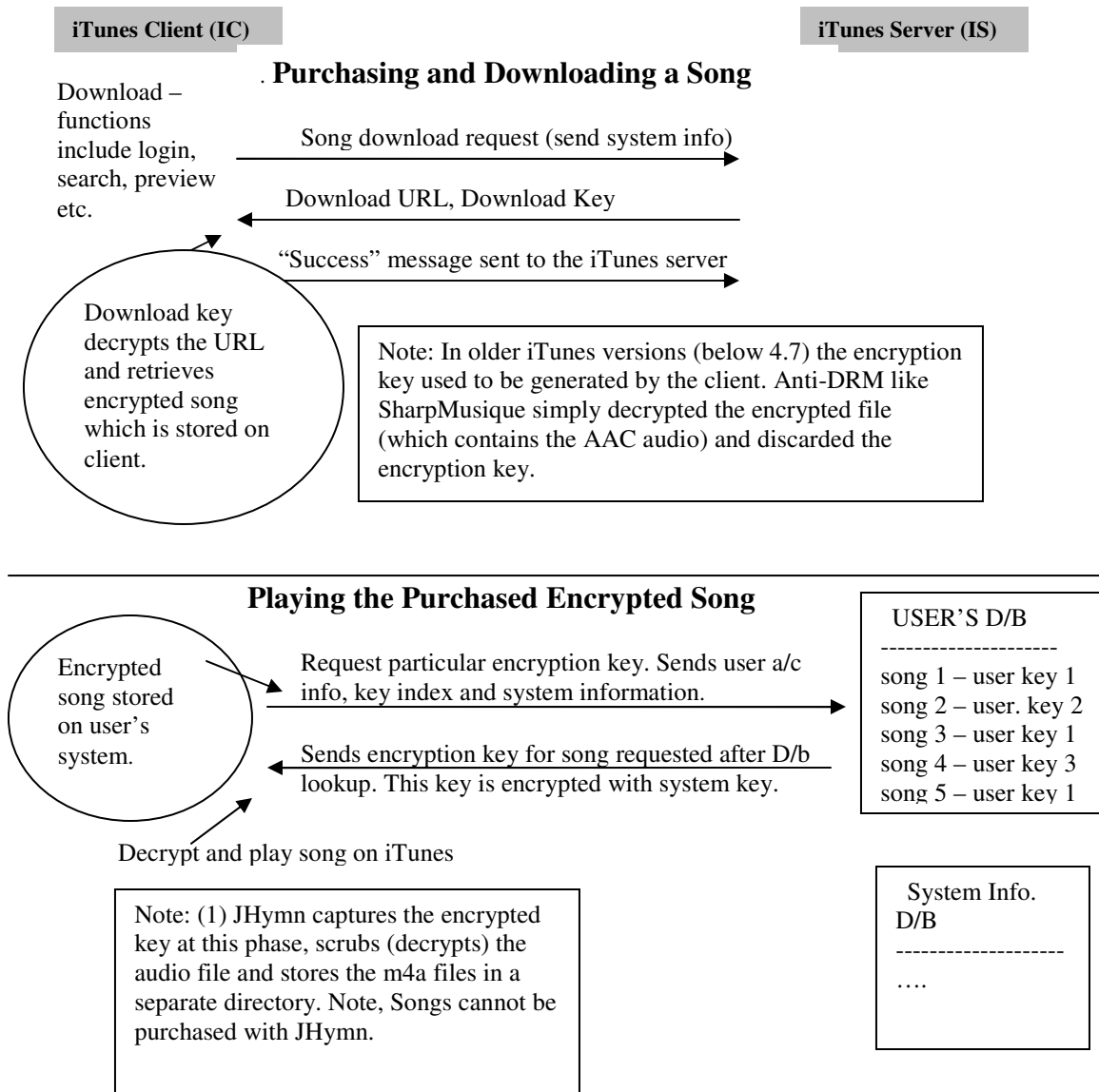


Figure 4: iTunes Protocol for Purchasing and Playing a Song

2.4 Reverse Engineering iTunes

Apple's Fairplay DRM technology is a closed source system. Reverse-engineering a closed source system is generally a difficult task since considerable effort might be needed to determine the functionality.

Jon Lech Johansen who cracked the Content Scrambling Scheme (CSS) encryption used to protect DVD movies, is credited with reverse engineering the Fairplay DRM scheme (Wen, 2004) (Indigo Group, 2005). PlayFair, developed by Johansen, is the first anti-DRM tool for Fairplay. Other anti-DRM software that exploited Fairplay include PyMusique, SharpMusique, JHymn, and QTFairUse6 (Indigo, 2005) (ipod news, 2006).

The iTunes client uses HTTP XML messages to communicate with the iTunes music store and these messages are encrypted using AES Cipher-block chaining (CBC) algorithm to prevent third parties from eavesdropping (Indigo, 2005) (Bornstein, 2005). However, the biggest weakness of any DRM system is that the user has to be given all of the "pieces of the puzzle" to play the digital content. This leaves us with two possible ways to exploit Fairplay:

- 1) Interface directly with the music store using a phony client similar to iTunes.
- 2) Get the decryption key from the user's system since Apple must give a user any keys needed to play a song.

The 'Hymn' (Hear Your Music Anywhere) project, based on Johansen's work, is a phony client that interfaces with the online music store (Indigo, 2005) (Futureproof, 2006). JHymn, authored by someone who goes by the alias FutureProof, is a graphical

implementation of the original command-line “hymn”. Figure 5 illustrates the JHymn interface.

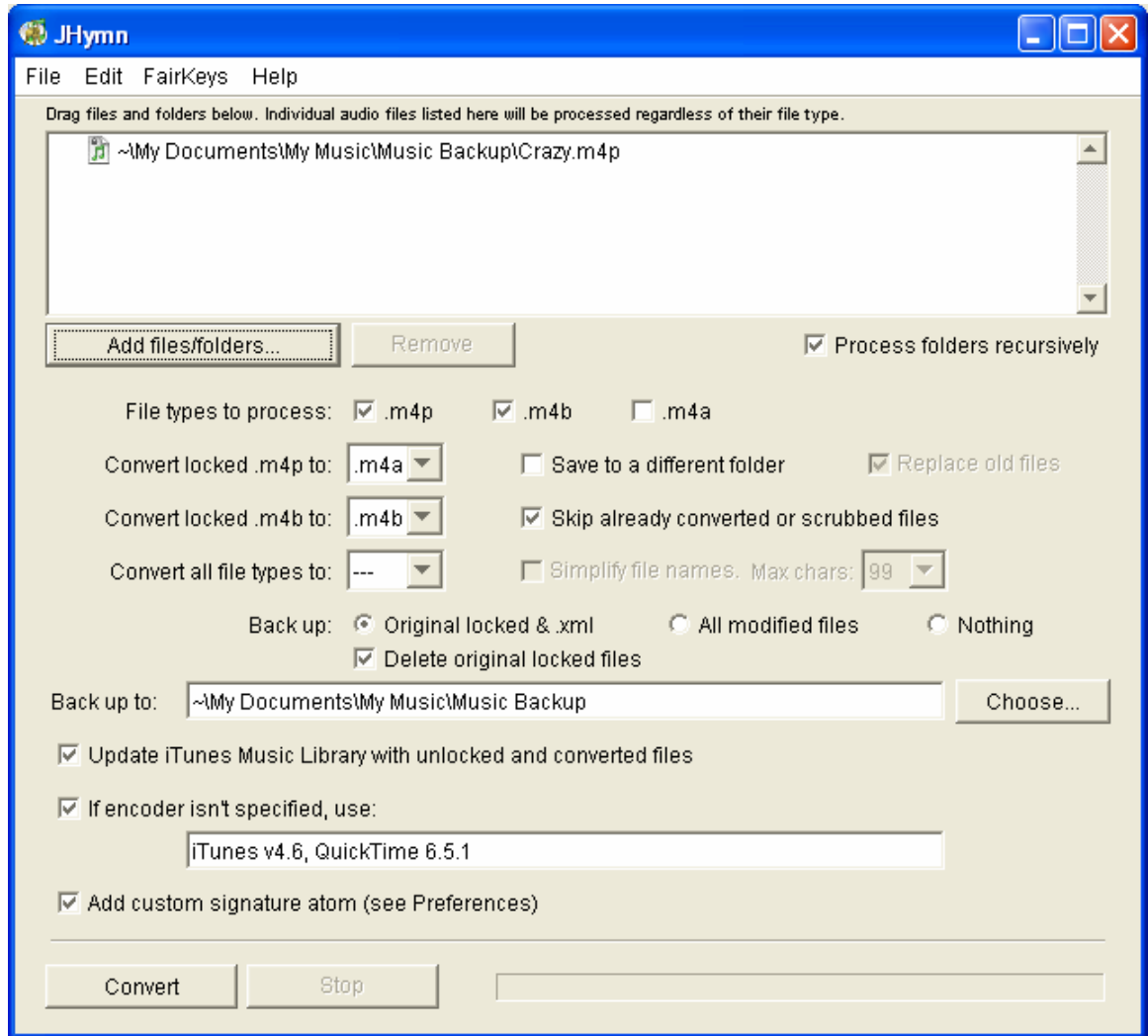


Figure 5: JHymn GUI Interface

JHymn “scrubs” protected AAC files (m4p) into an unprotected file format (m4a) (Wen, 2004). Scrubbing removes Fairplay DRM data from the metadata atoms and leaves unprotected files free of any DRM restrictions. Scrubbed files can be played on any AAC compatible software or hardware. Files scrubbed using JHymn are also

playable on iTunes since watermarking information that the Apple's iTunes software looks for is left intact (Wen, 2004).

JHymn can be used on files that have been purchased with any iTunes version older than iTunes 6.0 (Futureproof, 2006). However, if the user performs any activity such as authorizing a new computer, purchasing content using iTunes 6.0 or later versions, JHymn will not be able to scrub any more files, including files purchased using earlier versions (Futureproof, 2006).

The iTunes version 6 remained unbroken until anti-DRM software called QTFairUse6 was released in August 2006. QTFairUse6 successfully reverse engineered iTunes version 6.0. It captures AAC frames after the song has been decrypted, but before the decoding step and copies it to a file (ipod news, 2006).

Hence, no matter how strong the encryption scheme may be, the real vulnerability is the point at which the song is being decoded to a format understood by a soundcard. A decoded song can be captured by inserting breakpoints in the iTunes client and copying the decrypted song from the computer's memory into a new file. Anti-DRM software can find the proper place to insert the breakpoint and copy the data from memory. Such software was developed by reverse engineering the iTunes client software.

2.5 Introduction to P2P Networks

A P2P network consists of a large number of computers or nodes networked in an ad-hoc fashion (Wikipedia P2P, 2006)(Daswani, 2003). Each peer can function as both a content provider and/or a consumer (Han, 2005)(Mannak, 2004).

P2P systems make file sharing optimal since the main cost of sharing data, namely, bandwidth and storage are distributed across peers of the network. This ensures scalability and eliminates the need for powerful and expensive servers (Daswani, 2003).

There are 2 major P2P architectures (Han, 2005):

- centralized
- decentralized

One of the most widely known centralized P2P system is Napster which uses a central server to index into peers. Kazaa is another example of a centralized P2P system. In general, centralized P2P systems rely on centralized servers for specific tasks such as bootstrapping, obtaining global keys for data encryption, adding new nodes to the network, etc. Nodes in centralized systems perform tasks such as locating and caching content, searching for other nodes, routing messages, encryption, decryption and verifying content, etc. independently (Androutsellis-Theotokis, 2004). Centralized P2P systems are vulnerable to denial of service attacks (Han, 2005).

P2P networks that share computer resources such as CPU cycles, storage, content without requiring intermediation from a centralized server are called decentralized P2P systems. Gnutella, OceanStore, Seti@Home are few examples of decentralized P2P systems. Decentralized P2P systems enjoy high fault-tolerance, scalability, ability to self-organize in systems with highly transient node populations, increased access to resources, etc. (Androutsellis-Theotokis, 2004).

However, there are several challenges that prevent widespread acceptance of P2P systems. Such challenges include security, efficiency, and performance guarantees like atomicity and transactional semantics, unreliable peers, etc. (Daswani, 2003).

2.6 Security Aspects of Sharing over P2P

Some of the security pitfalls of sharing information over a P2P network are the following:

- 1) Installation of malicious software code.
- 2) Ports opened to transmit files are subject to malicious attacks.
- 3) Denial of service attacks.
- 4) Client nodes in a P2P network should ideally share information and service requests. Unfortunately, not all nodes in a P2P network service requests even if the node is able to do so. Nodes may misreport information such as bandwidth to get fewer requests (Saroiu, 2002). This might result in fewer server nodes and more client nodes.

Possible ways to prevent malicious attacks on a client node would be to use anti-virus software and enable firewalls (CERT, 2005).

There are several approaches that address security issues generally seen in P2P systems. Cryptographic algorithms such as self-certifying data and protocols such as information dispersal and Shamir's Secret Sharing scheme are employed to secure content published and stored in P2P networks. In the self-certifying data algorithm, nodes compute cryptographic hashes and an integrity check is performed (the hash value is verified) by the node retrieving the data (Androutsellis-Theotokis, 2004). Protocols such as the information dispersal algorithm and Shamir's Secret Sharing scheme employ

techniques that distribute files such that the file information cannot be obtained by any intermediate nodes in the P2P network.

2.7 Goals of this Project

This project analyzes Apple's Fairplay DRM to understand its design and extends it to function over a P2P environment. This new design, which we call *P2PTunes*, is intended to accomplish several goals, including:

- Ensure that P2PTunes provides same level of security as the current Fairplay DRM.
- Ensure that P2PTunes provides file sharing and bandwidth related advantages present in P2P networks.
- Ensure confidentiality of transactions.
- Provide integrity for digital content transferred over the underlying P2P network.
- Provide authenticity for purchased digital content.

2.8 Existing Research

Music2Share proposes a P2P protocol to manage legitimate music tracks using technologies such as watermarking, fingerprinting. The authors admit that the application of these technologies "has to be worked out and refined" (Kalker, 2004).

Napster, a centralized P2P file sharing system, was infamous for facilitating illegal music downloads and is now a legal music download service which uses Windows Media DRM. Downloads are encoded as high-quality 192Kbps Windows Media Audio (WMA) format. Napster allows registered users to stream tracks a certain number of times from

its catalog to Windows, Macintosh, and Linux machines. Streamed tracks cannot be downloaded to the user's system and are encoded with a low bit rate. A Napster client is required for song purchases and downloads. Napster's online streaming works well on Windows although there are some performance issues on Macintosh. Napster's client software controls "hang" at times, not allowing users to adjust the volume, pause, or skip ahead. But, Napster provides an innovative feature that allows users to explore other member's collections by genre. However, this feature needs to be refined since a search for songs in a particular genre results in songs from different genres displayed in the search result (France, 2006)(Chandak, 2005).

Rhapsody 3.0, owned by Real Networks, offers an on-demand streaming service and download access from its music catalog. Purchased tracks are 192Kbps AAC files wrapped in Real Network's Helix DRM. Rhapsody allows users to listen to a certain number of tracks free each month; these tracks are encoded at 128Kbps. However, Rhapsody 3.0 works on Windows XP, Me, 200 or 98SE platforms and there is difficulty "logging onto" the service which may be attributed to high server traffic (France, 2006)(Chandak, 2005).

Other examples of P2P-based download services include eMusic, Yahoo Music Unlimited 1.1, Sony, etc. eMusic offers a catalog of independent labels' songs for download in the unprotected "mp3" format encoded at Variable Bit-Rate (VBR). There is a maximum download limit per month after which songs may be purchased (France, 2006)(Chandak, 2005).

3 Design

Our proposed DRM system, P2PTunes, adapts the existing Fairplay DRM over a P2P network. This section describes the design and architecture of P2PTunes.

3.1 A Hybrid Approach

The strength of P2PTunes lies in deriving the benefits of a P2P system while ensuring the security aspects that are needed in a reliable DRM are incorporated into P2PTunes. In addition, the design of P2PTunes ensures that it functions over any type of P2P system.

The P2PTunes DRM allows users to purchase encrypted digital media such as audio and video content over the P2P network. In the P2P network, each peer functions as both a content provider (server) and a consumer (client). Client nodes broadcast queries for specific digital content while server nodes service these requests. A node acts as a server if it has the response to the broadcasted query. Nodes acting as client nodes in one transaction, could act as server nodes in another transaction.

In P2PTunes transactions, digital content is always transmitted in encrypted form over the P2P network. This prevents intermediary nodes and client nodes from “grabbing” the content before a payment is made.

P2PTunes should as decentralized as possible by enabling peer nodes to share encrypted digital content. However, certain transactions such as payments, billing, and content authenticity verification must be performed at a centralized server to ensure legality of P2PTunes transactions.

A secure connection between the client node and centralized server is used for payment processing. In addition, the centralized server validates the authenticity of the purchased digital content.

The introduction of the centralized server in combination with the decentralized P2P makes P2PTunes a hybrid model. The hybrid model enables achieving a reliable and secure P2P network for sharing digital content and is the key to extending the existing iTunes functionality to facilitate seamless functioning of P2PTunes.

3.2 Entities of P2PTunes

The main entities of this P2PTunes are listed below:

- Originator (O): a client node that initiates a request for digital content over the P2PTunes network.
- Responder (R): a server node that can service the originator's request.
- Intermediary Node (P): a node that forwards O's and/or R's messages.
- P2PTunes Server (PTS): a centralized server that completes a transaction by handling the financial aspects. In addition, it verifies the authenticity of purchased digital content. Necessary action is taken in case the content is not authentic. Note PTS is a trusted component of this system.

3.3 Architecture of P2PTunes

This section describes the architecture of P2PTunes. The interaction between the various entities of P2PTunes and how to securely distribute encrypted digital content over the P2PTunes network is described in this section.

3.3.1 Description of the iTunes Metadata

Every protected m4p file contains priv and iviv atoms enclosed within the moov atom. The priv and iviv atoms are pivotal to the decryption process. Other atoms such as user, key, and name contain user specific information: iTunes user ID, user key index, and user name respectively. Figure 6 depicts these atoms. For a more detailed view of atoms refer Figure 3.

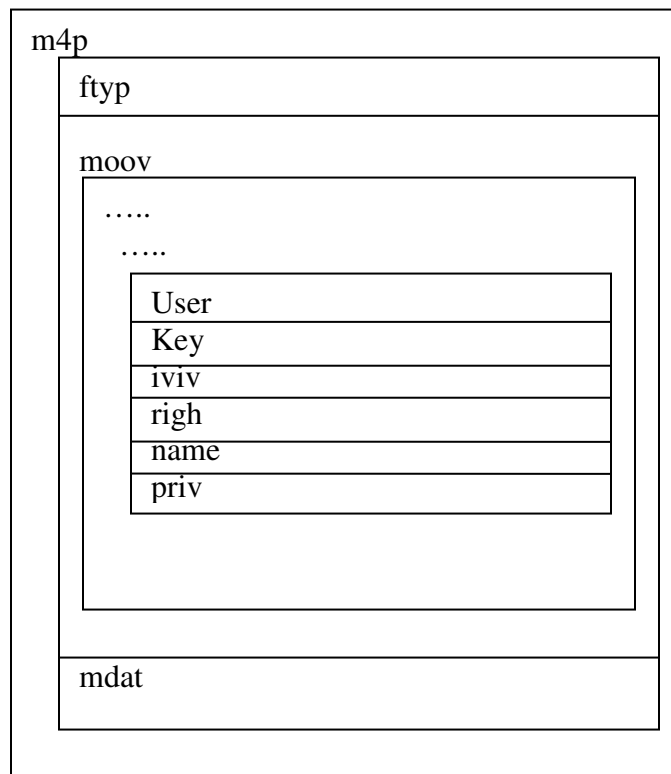


Figure 6: Key Atoms in an m4p File

Different digital content use the same AES private key to decrypt the mdat atom. However, the AES key is encrypted with a user key (which is user specific and differs from user to user) and stored in the priv atom. The user key in turn is encrypted with a system key while it is transmitted from the server to the client. The system key is both

user and system specific, since each user can authorize different system to play the same content. The P2PTunes Server (PTS) uses system information from the user's system to generate the system key. PTS stores different user keys per account and uses a user key index (found in the m4p file) to retrieve a user key from its user database. Hence user-specific atoms play a role in retrieving the user key from the server's database.

3.3.2 Steps Involved in Purchasing and Playing a Song

Figure 7 illustrates a network of Originator, Intermediary and Responder nodes in a sample P2P network.

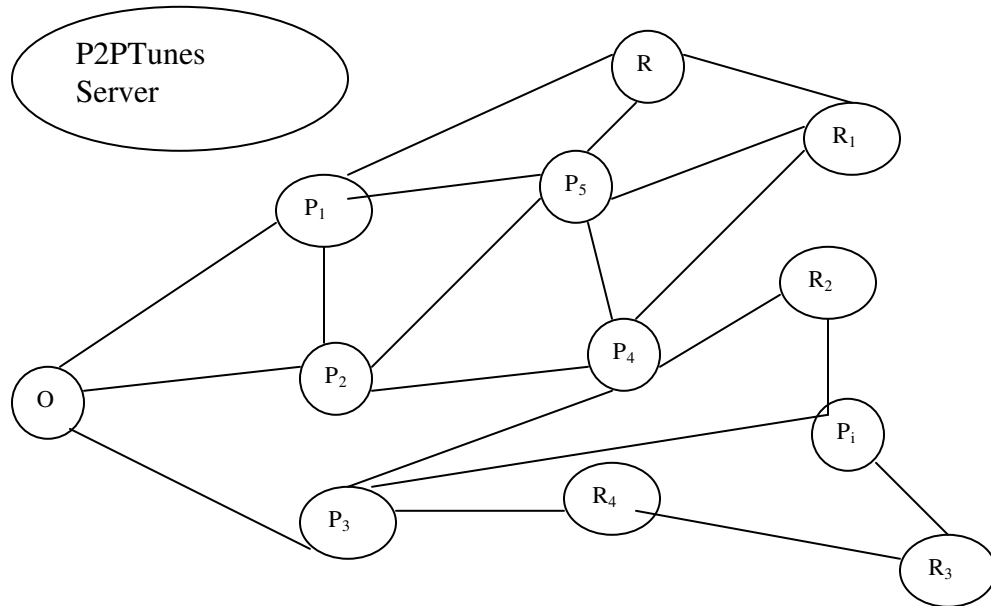


Figure 7: A Sample P2PTunes Network

The steps involved in purchasing digital content are explained below.

Step 1: Initiate Request – Originator O (a client node) initiates a request for a specific digital content, say a song ‘S’. The request S along with O’s identifier is broadcast over the P2PTunes network. The identifier helps tie a request to a node.

Step 2: Respond to Query – Responder R (a server node) possesses the required digital content. Upon receiving request S, R sends its identifier and song information (which include title, song number, version number, etc.) to O. The identifier helps tie the response to a responder node, similar to the previous case. Note there could be more than one responder to a query.

Step 3: Select Response – Node O receives responses from ‘n’ responder nodes ($R_1 \dots R_n$). O chooses any one Responder R_i .

Choosing a responder is based on certain parameters such as:

- The responder node may be a known and trusted by the originator.
- Actual digital content information such as title, version, singer, etc.
- Speed of connection between client and chosen server node.

Step 4: Inform Responder – Node O requests the chosen R_i to deliver the song S by sending a download or “request-confirm” message to R_i .

Step 5: Transmit Encrypted Content – Upon receiving O’s message, R_i encrypts its account information and a timestamp with the PTS public key. Note the PTS uses a

public key cryptosystem such as RSA to generate a public and private key pair. Node R_i computes the hash of the encrypted song and the encrypted account information.

Node R_i strips user-specific atoms such as name, user ID, etc. from the m4p file and sends the content byte stream to O which consists of:

- encrypted song (excluding the user-specific atoms)
- node R_i 's encrypted account information and timestamp
- checksum of the encrypted song and the encrypted account information..

The purpose of encrypting R_i 's account information with the current timestamp is to prevent misuse by O. Since it is encrypted with the PTS public key, O will not be able to decrypt this message. Additionally, the timestamp enables the PTS to ensure that the transaction is contemporary.

Intermediate nodes cache this content and can later forward the same content to similar requests over the P2P network.

Step 6: Verify integrity at O – Node O receives R_i 's encrypted message.

Node O verifies the integrity of the song by computing the checksum of the encrypted content and verifying it against that received from R_i . Once verified, O is left with the encrypted song received from R_i .

Step 7: Verify Authenticity – It is important that O verifies the authenticity of the m4p file it received from R_i over the P2P network.

Node O computes the hash value/HMAC of the non-user specific atoms. User specific atoms that vary from user to user such as priv, user, name, etc. are not present in the byte stream and hence not used in the HMAC computation. However, other non-user specific atoms such as mdat (which contains the song) that is encrypted with the same AES private key for every user will be included in the computation. This computed hash value and the hash key along with the unique song number are sent to the PTS for verification. To verify the authenticity, the PTS simply compares the received hash value with the computed hash value of the content. This enables PTS to verify if the hash value received from O corresponds to the content.

In this way, the PTS verifies the authenticity of the content (section 5.1 explains the benefit of this method). If the song is not authentic, the PTS informs O and O can try another source.

An alternate approach would be for the PTS to send the hash key of the content encrypted with node O's system key to O. Node O can compute the HMAC using this key and send the computed value to PTS (encrypted with its system key). The PTS can verify if the computed HMAC for that content. This approach reduces the computation task of the PTS.

Step 8: Payment Processing – To be able to play the song, O needs to be able to decrypt the song, which it can after making a payment to the PTS.

At this stage, O connects to the PTS though a secure connection and sends the following to PTS:

- node O's account information
- song information (including a unique song number)
- node R_i 's encrypted user information

A secure link is setup between PTS and O by the PTS to complete the payment.

Step 9: Include User-Specific Atoms – On completion of payment, the PTS has to ensure that all of O's user specific atoms are generated.

Node O sends its system information to the PTS, which uses this to generate a system key. Node O's user-specific atoms are encrypted with the system key by the PTS. Note that the system key is stored on the client (i.e. O's system).

The PTS sends node O's priv atom for the content downloaded in step 5 to O. Further, the PTS generates and sends O's user-specific atoms pertaining to information such as user key index, name atoms, etc. to O. The iTunes software functionality is enhanced to be able to add these user-specific atoms received from PTS to the m4p file downloaded in step 5. This step is very important since when O plays the song, the PTS looks up information from these atoms to retrieve the user key.

Step 10: Play Purchased Content – Each time node O wants to play a track, it provides the song request, user specific information (from the m4p atoms), and system information to the PTS. The PTS uses the song and account information sent by O to index into its database and retrieve the appropriate user key. The user key is then encrypted with a system key generated by the PTS and sent to O. Three levels of decryption are done:

- First level: O uses the system key (stored on the client) to decrypt the encrypted user key received from PTS.
- Second level: The decrypted user key is used to decrypt the priv atom. This retrieves the AES private key.
- Third level: The AES private key (from the second level) and initialization vector (obtained by hashing the name and iviv atoms) are used to decrypt the audio data in the mdat atom.

This is similar to the way digital content is played in the current iTunes.

3.4 Benefits of the Proposed System

This section outlines the benefits of the proposed P2PTunes system:

- Presently digital content purchases are made to a centralized iTunes server and iTunes services each request by providing the song. This consumes a lot of bandwidth and creates a potential bottleneck at the centralized server. In our new design, the download bandwidth is reduced at the server as user's share content over the P2P network.
- The new design has all of the advantages of a P2P community with active user participation and community networking (Mannak, 2004).
- Each m4p files has user-specific metadata tags pertaining to a specific user which is identical to the present iTunes system. Also, playing purchased content is

identical to the existing iTunes system where users need permission from the centralized server to play content.

- The new design seamlessly extends the current iTunes system over a P2P network.
- Confidentiality is enforced to a certain extent since the responder node removes its user-specific atoms from the protected file before sending it over the P2P network.

3.5 Potential Issues

It is well-known (Saroiu, 2002) that nodes may simply misreport information such as bandwidth in order to service fewer requests, or no requests at all. This creates an environment of large number of client nodes and fewer server nodes which is not an advantageous P2P scenario. Taken to the extreme, this would result in P2PTunes degenerating into the existing iTunes model with a centralized server distributing the content.

Another issue is that participants in P2PTunes may simply share unencrypted content over P2PTunes or offline. Note that this scenario is also possible in the present iTunes system, where a tool can be used to scrub content and the resulting unprotected content can be shared by users. To encourage users to share encrypted content, it is suggested that the server node from which the client made a purchase be rewarded with a small financial incentive from the PTS. This would create a financial incentive for users to “play by the rules”.

No matter how strong the encryption scheme may be, the real vulnerability lies in the point at which the song is being decoded to the format understood by a soundcard. At this point, it is possible for a user to capture the unencrypted digital content and simply copy it to a file. The key to preventing such attacks lies with using a trusted computing base (TCB) enabled OS which can prevent inserting breakpoints in the application and also by using dedicated tamper-proof hardware to decrypt the encrypted content and play it.

3.6 Functional Architecture

This section describes the interaction between the P2PTunes entities. The three major entities of the system include the Originator (O), Responder (R) and the centralized P2PTunes Server (PTS). Figure 8 illustrates the functional architecture.

The process of buying digital content over P2PTunes proceeds as follows. An originator requests digital content through the underlying P2P network (step 1). Responder nodes, which are able to service the request transmit an index containing content information through the P2P system (step 2). Originator O chooses a responder from the client browser based on its discretion and sends a download request message to the chosen R (step 3). Responder R encrypts its account information with the PTS public key, strips off user-specific atoms from the m4p file, performs an integrity computation, and transmits the digital content to O through the P2P system. Node O proceeds to verify the integrity of the transmitted content (step 6). The transaction is complete once O communicates with the PTS to authenticate the content (step 7), complete the financial payment (step 8), and finally receive user-specific atoms pertaining to the purchased content (step 9). The PTS is entrusted with the responsibility of ensuring that R is the

legal owner of the digital content. This is done by decrypting the encrypted account information (sent to O from R) and verifying ownership against the PTS database. At the end of the purchase transaction, the PTS provides R with an incentive to encourage participation in the P2PTunes system.

Playing a purchased song requires communicating with the PTS to request a user key specific to the content. This is explained in Section 2.3.3 above.

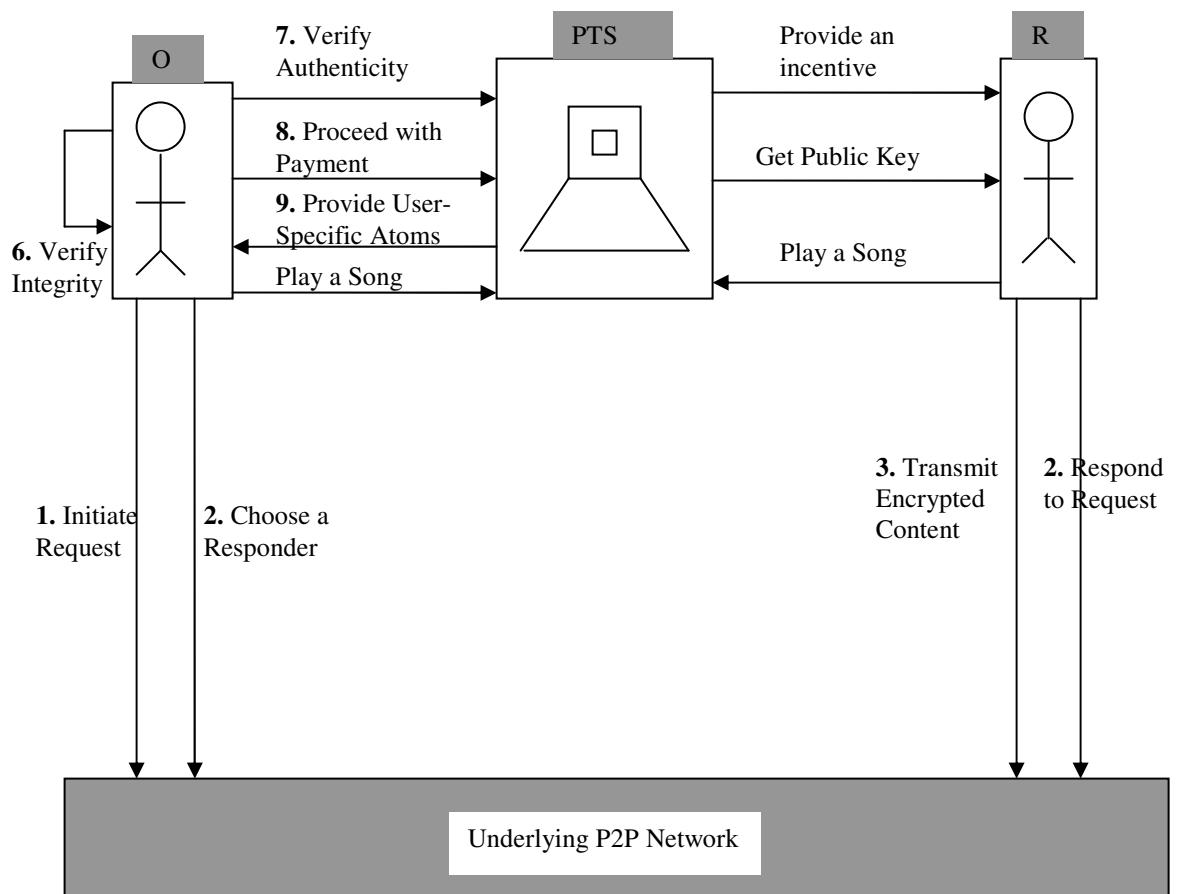


Figure 8: P2PTunes Functional Architecture

4 Implementation

This section discusses the implementation details of the P2PTunes prototype. The underlying idea of this prototype implementation is to demonstrate the functionality of P2PTunes over a simple P2P network.

Java was used as the development language, with Eclipse as the development environment, and Windows XP was the chosen development platform. All user interfaces were implemented using Java swing.

The following sections present details of the implementation of the originator, responder, and the centralized PTS server of P2PTunes as well as the creation of a basic façade P2P network. A security analysis of the implementation is conducted in section 5.

4.1 Underlying P2P Network

The façade P2P network created consists of a simple fully-interconnected network of nodes. Each node is built as a client-server pair and functions as an independent node in P2PTunes. Nodes in this network run as separate Java Virtual Machines (JVM). Even though the P2P network is implemented as a fully-interconnected network, the design concepts of P2PTunes are flexible enough to be implemented over any other type of P2P network.

Each client communicates with the other nodes of the P2PTunes network by communicating with the servers corresponding to those nodes. The number of nodes in P2PTunes can be specified by the user through a facadeP2P GUI. An online/offline

functionality simulates a realistic P2P setting where nodes can log-in or log-out of the system.

Java Remote Method Invocation (RMI) facilitates communication between the nodes. Java RMI enables a programmer to create distributed Java technology-based to Java technology-based applications in which remote methods of Java objects can be invoked by other JVM's, possibly on different hosts (Sun Developer Network,2006). In other words, this enterprise-level solution enables a virtual machine to execute remote methods of another machine's class as if it were residing on that same machine. A local object called stub, present on the client side of the JVM, has methods of the remote machine that can be invoked by the client. On invocation of a method, the stub sends the method call to the skeleton which is present on the server side. The skeleton then implements this method on the server and sends the results back to the stub.

4.2 P2PTunes - Class Diagrams

The class diagrams depict the relationship between the various system classes. The class diagrams of P2PTunes are presented below:

- P2PTunes Server (PTS): is a centralized server system responsible to complete a purchase transaction by performing billing tasks, authenticity verification, and generation of user-specific atoms. Additionally, PTS provides user keys to enable decryption of purchased digital content for playing the content.
- User Components: consists of the originator and responder classes. Methods include client functions that search for digital content, display search results,

compute checksum, initiate payment, replace user-specific atoms, and log-in or log-out of P2PTunes.

4.2.1 Originator

Figure 9 illustrates the relationship between different classes related to the originator. The “UserInterface” class displays results and receives inputs from the user. The “RequestContent” class broadcasts user requests over P2P. The “ReceiveContent” class performs various computations like integrity checks, HMAC calculation for authenticity verification, etc. Communication with other nodes in the P2P takes place through the “facadeP2P” class.

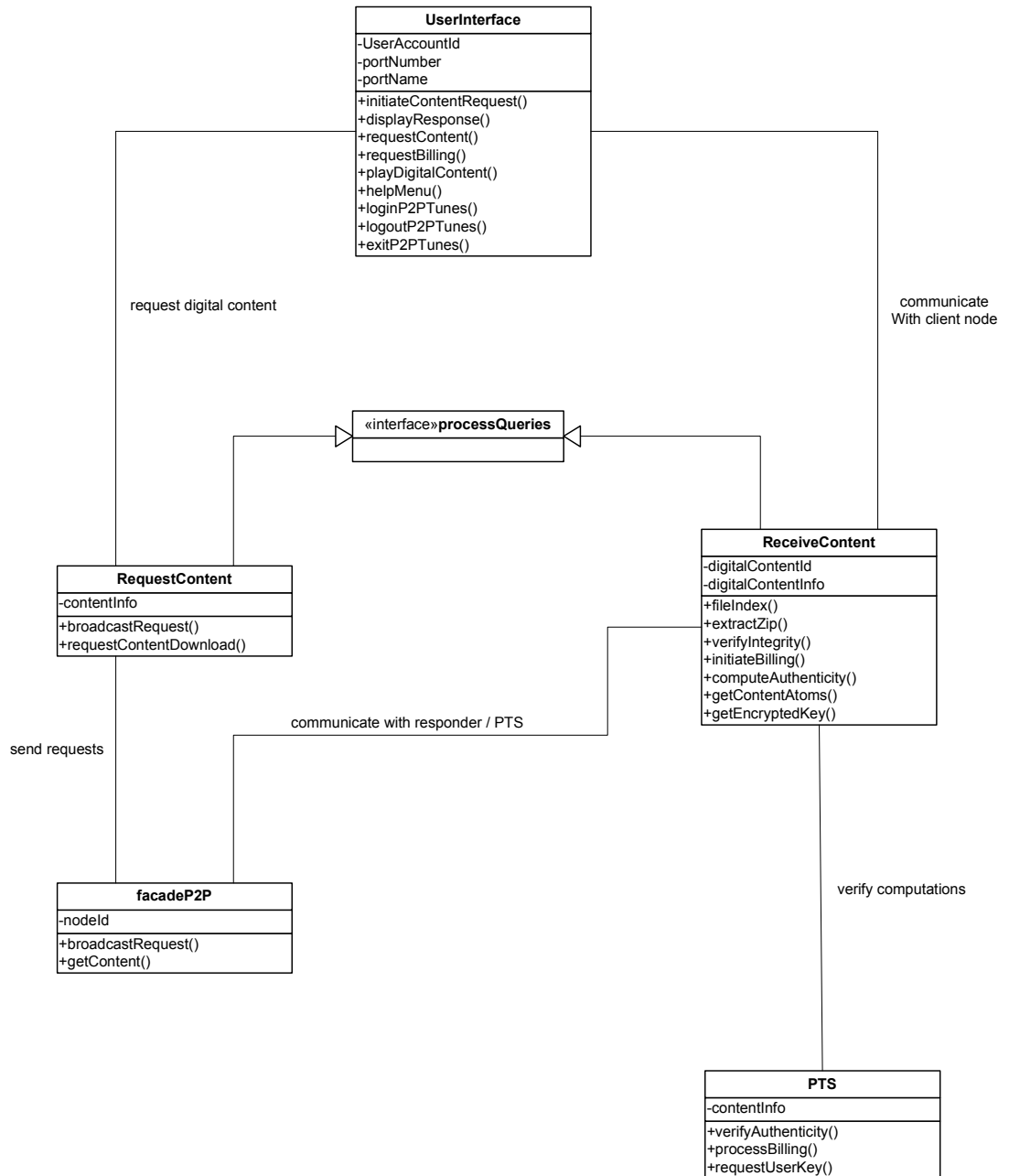


Figure 9: Originator Class Diagram

4.2.2 Responder

The function of the responder is to service requests for digital content. Figure 10 illustrates the responder class diagram. The responder “listens” to all content requests made over P2PTunes network and determines if it is able to service a request. If so, the “PrepareResponse” class prepares a file index containing details of the content and sends this to the originator through the “facadeP2P” class. Once an originator node chooses a responder, “ReceiveRequest” class will obtain the download request and the “PrepareResponse” class will “bundle” up the content and transmit it to the originator through the “facadeP2P” class. Note that in the process of creating the output content byte stream, the responder communicates with the PTS to get the PTS public key which is required to encrypt its account information.

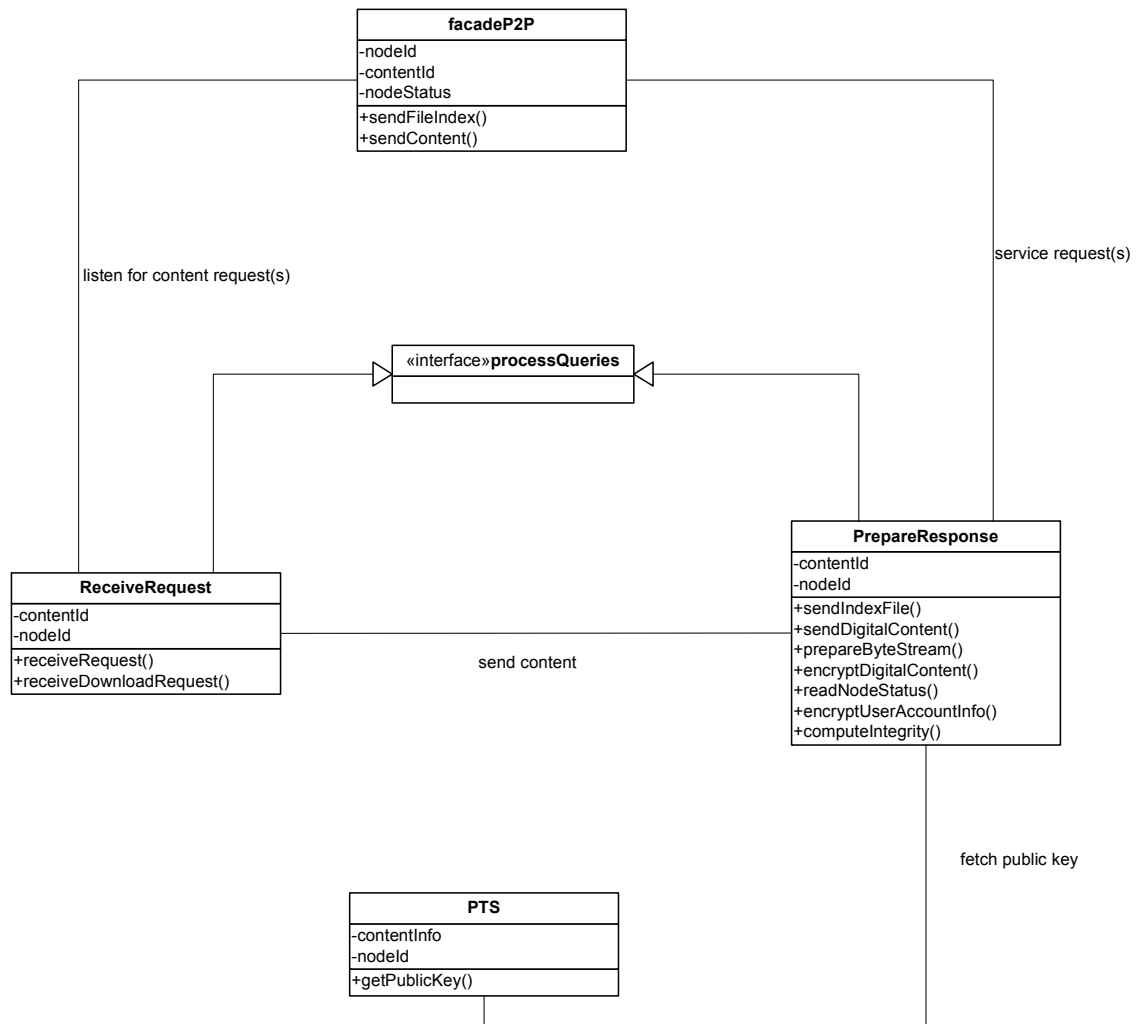


Figure 10: Responder Class Diagram

4.2.3 P2PTunesServer (PTS)

The centralized server is the key element involved in completing transactions over P2PTunes. Figure 11 illustrates the PTS class diagram.

The centralized PTS server accepts requests for authenticity verification, payment processing, playing a song, etc. from the originator and gets the public key for the responder. Communication between all other nodes in P2PTunes and the PTS is a standard client-server communication.

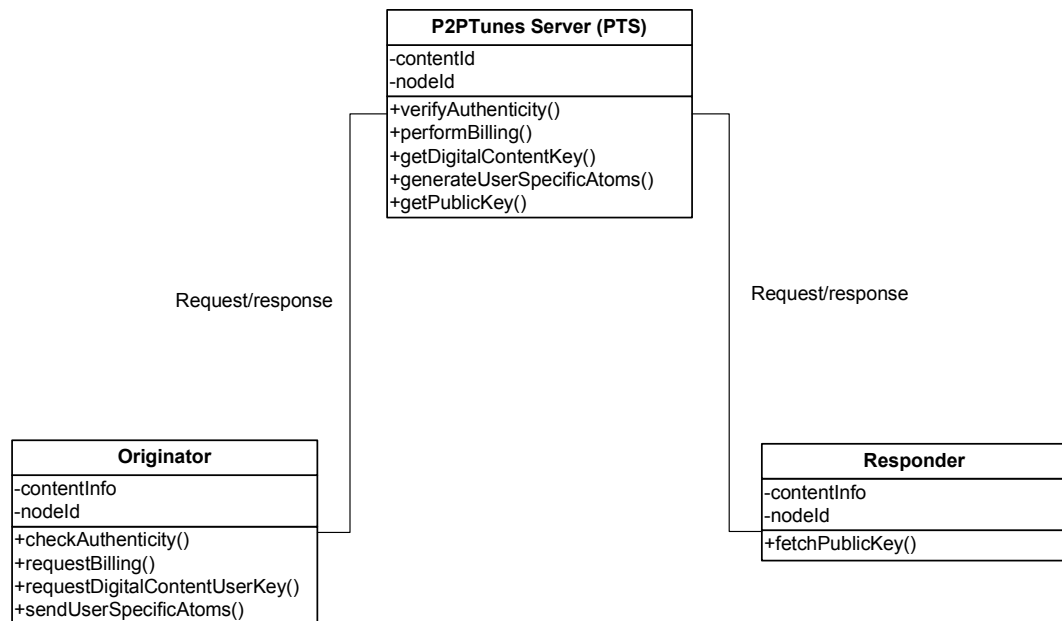


Figure 11: PTS Class Diagram

5 Security

File-sharing capabilities of P2P technology threaten the privacy and security of individuals and businesses through the disclosure of network IP and MAC addresses, and could also potentially distribute viruses. Novice users who possess little technical knowledge may accidentally breach systems by sharing certain folders or even the entire hard drive (Bailes, 2004).

In spite of the many known weakness P2P networks are a formidable means of social networking which itself has been gaining a lot of momentum in recent times. The strengths as well as weakness of each aspect of P2PTunes are thoroughly analyzed in this section.

5.1 Security Analysis

This section scrutinizes each step involved in purchasing content over P2PTunes.

Step 1: Initiate Request – An originator node broadcasts requests for content over the P2PTunes network. At this stage every intermediary and responder node is aware of the identity of the originator node. Anonymity is not considered and the responder is aware of the originator's IP address in order to respond to the query. An alternative would be to use a secret-sharing-based mutual anonymity protocol (SSMP) to allow peers to issue queries and responders to deliver responses anonymously (Han, 2005).

Step 2: Respond to a Request – Responder nodes send a file index containing details of the content and the identity of the responder node to the originator. The size of the file index sent is in the order of a few kilobytes which is much smaller than that of the actual song. Since the actual content is not sent to the originator node it prevents throttling the P2P bandwidth. The potential drawback at this stage is malicious nodes could bombard an originator with a large number of dummy file indices that might result in the originator node's system hanging.

Step 3: Choose Specific Content – The originator node makes a choice for a particular content, based on details indicated in the response. A download request is sent to the corresponding responder node over P2PTunes. The advantage at this stage is that the originator can make an informed choice based on the content information and the responder's ID. This can be enhanced by devising a rating system in which trusted responders can be certified by other users and the PTS.

Step 4: Chosen Responder's Content Transmitted – At this stage, the requested content from a responder is transmitted to the originator over P2P network. The content file (protected m4p file) is made up of atoms each of which contains information pertaining to the content (ex. artist name), identification information (ex. iTunes user ID), the encrypted song, and cryptographic information (ex. encrypted keys). The responder strips the user-specific atoms from the m4p file and prepares a byte stream. The absence of the identification atoms prevents intermediate nodes from identifying the responder and

hence provides a certain degree of anonymity. In addition, atoms essential to the decryption process namely the priv atom (contains the encrypted AES key), name atom (identification atom), etc. are excluded thereby preventing intermediate nodes from trying to decrypt the mdat atom and play the song. Additionally, since user-specific atoms are not present in the byte stream, intermediate nodes cannot fake acting as the responder node and request user keys from the PTS. If a node adds its user-specific atoms to the protected file and requests the PTS for the user key, the PTS would not comply since it could easily verify from its database that the particular node has not purchased the song and can take appropriate action. Hence, these preventive measures thwart unscrupulous intermediate nodes from playing content involved in a transfer.

It must be noted that intermediate nodes involved in the routing are aware of the content details since the remaining atoms of the m4p file include artist name, purchase date, album name, copyright information, mdat atom (which contains the actual song) etc.

Nodes cannot decrypt the responder node's account information (consisting of its user name and user ID) from the byte stream since it is encrypted with PTS/centralized server's public key. Hence, neither the intermediary nodes nor the originator node can view or alter the responder's account information.

The responder computes a checksum of the byte stream using CRC-32 with which the originator node verifies the integrity of the content.

Subsequently, the intermediate nodes can cache the content involved in the transaction and later forward the same digital content to other similar requests originating

from different nodes. This will make efficient use of the P2P bandwidth (Wikipedia, 2007).

As in the original system, content cannot be played from unauthorized machines since the PTS requests system information from nodes which should match that in its database to verify ownership and provide decryption keys (Apple Computers Inc, 2006).

Step 5: Originator receives song content – The originator verifies the integrity of the content against the received checksum to be certain that the data has not been corrupted. Subsequently, the originator verifies the content authenticity with the PTS (by computing the HMAC) to ensure that an unscrupulous node did not tamper with the content.

A huge part of the computation steps such as integrity computations, authenticity computation, “bundling” of the content is assigned to the participating P2PTunes nodes. This leaves the centralized PTS system with fewer computation tasks and makes resourceful use of the P2P bandwidth.

Step 6: Role of PTS – The role of the PTS is to verify the authenticity of the song and throw appropriate error messages allowing the originator to choose a different source in case of a failure. Furthermore, the PTS is responsible to complete billing and generate user-specific atoms to be added to the purchased m4p file by the originator. Once the originator receives the user-specific atoms, it can request the user keys from the PTS to play any purchased content.

Authenticity verification is done between the originator and the PTS. The originator calculates the hash value of non-user specific atoms in the digital content using HMAC. The calculated HMAC, HMAC key along with the content ID and encrypted account information is sent to the centralized PTS which validates the received HMAC for the specified content. In addition, the PTS verifies if the responder has purchased the song by decrypting the responder's account information and confirming ownership against its database. This guarantees that the digital content is from an authentic source.

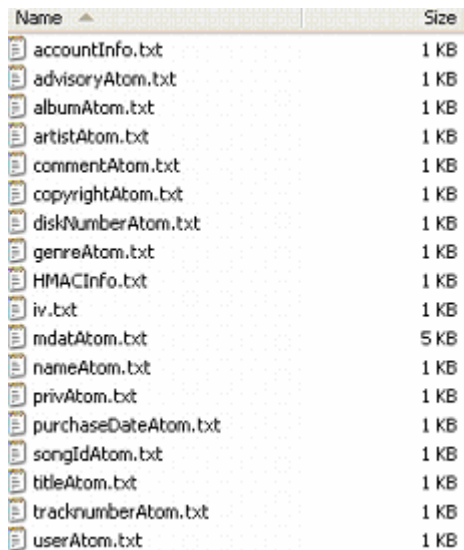
The user-specific atoms generated by the PTS are small in size compared to the size of the entire m4p content file. The PTS generates a priv key and encrypts the AES key with the newly generated priv key forming the priv atom. Additionally, the name atoms (consisting of the user name) and the user atom (consisting of the user ID) are generated and sent to the originator through a secure channel. The originator inserts these atoms into MPEG-4 file thus forming a "complete" protected file which can be played.

Step 7: Playing purchased content – is not different from the present iTunes system, and is described in section 2.3.3.

6 Testing

The P2PTunes system provides a fresh approach to distribute digital content over a P2P network. Most of the computation tasks are accomplished by the client nodes leaving the PTS with minimal tasks. Additionally, this system makes efficient use of P2P bandwidth to distribute content among peers.

The test cases involve distribution of content in the form of songs over P2PTunes. Each song folder consists of various atoms normally found in an m4p file such as mdat atom, name atom, artist atom, title atom etc. Each atom contains either encrypted information or plain text indicating user or song details. Figure 12 shows a snapshot of a song folder's atoms.



Name	Size
accountInfo.txt	1 KB
advisoryAtom.txt	1 KB
albumAtom.txt	1 KB
artistAtom.txt	1 KB
commentAtom.txt	1 KB
copyrightAtom.txt	1 KB
diskNumberAtom.txt	1 KB
genreAtom.txt	1 KB
HMACInfo.txt	1 KB
iv.txt	1 KB
mdatAtom.txt	5 KB
nameAtom.txt	1 KB
privAtom.txt	1 KB
purchaseDateAtom.txt	1 KB
songIdAtom.txt	1 KB
titleAtom.txt	1 KB
tracknumberAtom.txt	1 KB
userAtom.txt	1 KB

Figure 12: Atoms in a Song

Creation of tests cases involved generation of system keys, priv keys, and encryption of atoms such as priv and mdat. All mdat atoms consist of the encryption of the text

“Playing song ...<song name> by <artist name>” where the output would depend on the song and artist name. This message is interpreted as the equivalent of an audio file playing after the appropriate decryption steps have been performed. A stand-alone function was used to create keys and to encrypt or decrypt data required to test P2PTunes.

The testing procedure validates the design objectives while maintaining an eye on the system’s functions in order to confirm to the intended functionality. All the steps involved in creation of the P2PTunes façade network, buying a song, playing a song, emulating online and offline functionalities is demonstrated in this section. Sample test cases are used to verify the implementation.

6.1 Creation of a façade P2P Network

This step involves creation of nodes in the P2PTunes network. The network created is fully-interconnected allowing every node to communicate with all other nodes.

To create the P2P network go to Façade P2P->create and specify required number of nodes. Figure 13 illustrates the GUI. Every node created will need virtual memory space. Due to the memory limitation in the test system, creation of a large number of nodes will result in the system slowdown or crashing as system gets low on available memory. For testing purposes up-to 10 nodes can be created. On clicking “CreateP2P” button, 10 RMI registries and client UI interfaces are created.

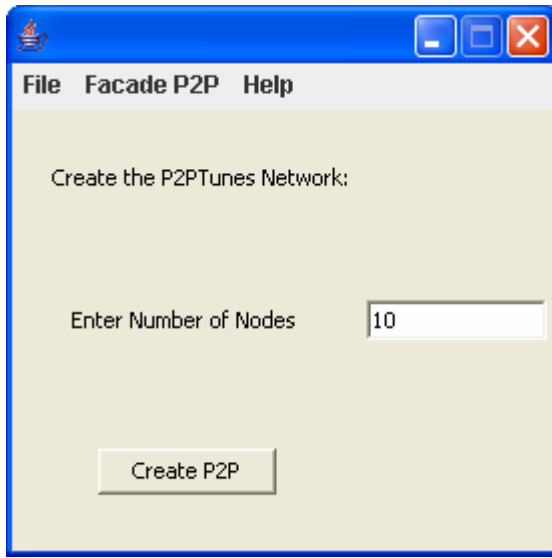


Figure 13: FacadeP2P Creation GUI

The 10 client nodes can interact with each other through the client GUI. An additional PTS node which functions as the centralized server is automatically generated.

Other features included in the GUI are:

Help->about menu which indicates the purpose of this frame.

File->exit menu allows users to exit P2PTunes.

6.2 Client Browser Windows

Figure 14 shows a client window. Each browser UI functions independently and can purchase and play content over P2PTunes. Figure 14 shows one such client node.

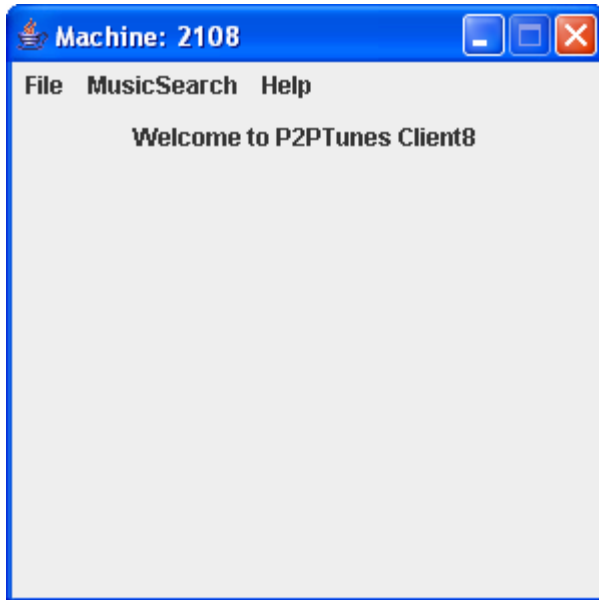


Figure 14: P2PTunes client UI

Other features included in the client GUI are:

File->exit menu which allows a user to exit client browser.

File->logout in which user can go into the offline mode thereby disabling all client functionalities and disallowing other nodes to purchase content from this node.

File->login which allows a user online status in P2PTunes, enabling the client to play or purchase content.

Help->about menu explains features of the client UI.

6.3 Purchasing a Song

Purchasing songs over P2PTunes requires a node to communicate with other nodes in the system to request a song. To search for a song through the client UI, go to

MusicSearch -> Search. Enter a song name and click on Search button as shown in Figure 15.

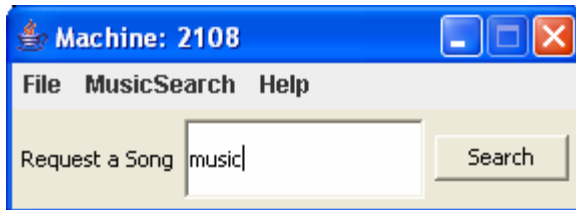


Figure 15: Request a Song through P2PTunes

A list of matching responses is displayed on the client UI as seen in figure 16. Note that a single responder node can have many partial matches to a song name. Details of all matching songs from all nodes are transmitted to the originator node.

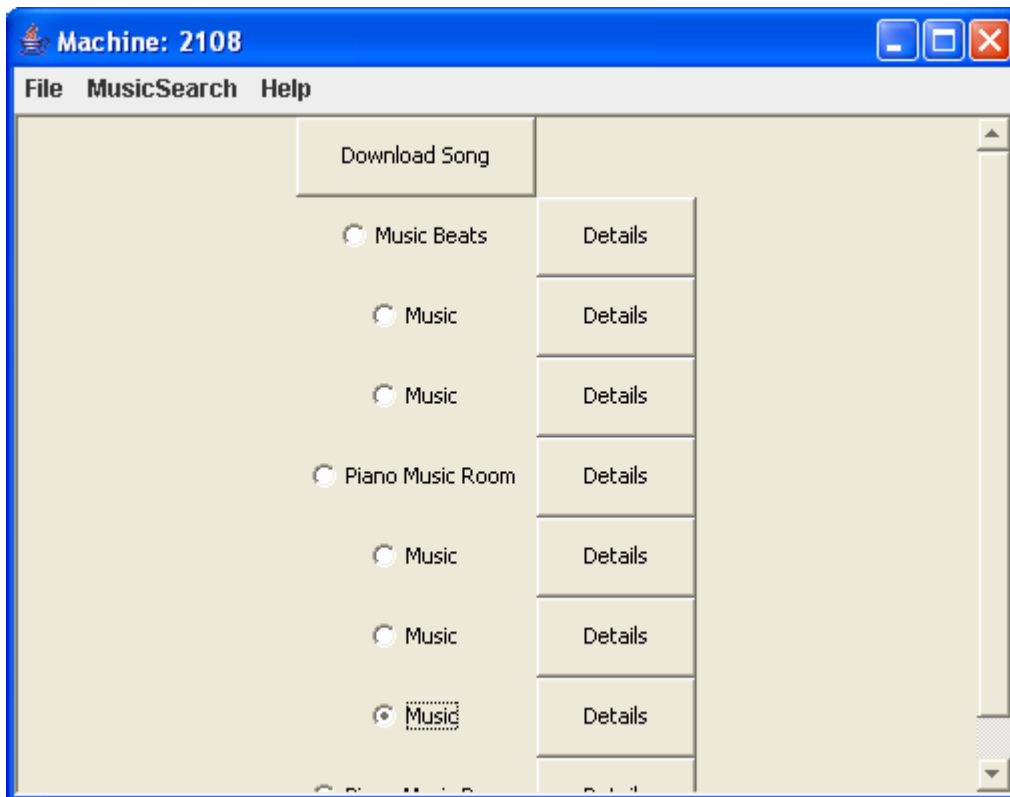


Figure 16: Responses for Content

Clicking on the “Details” button will present content/song details such as artist name, track name, etc. as shown in Figure 17.



Figure 17: Details of Content

The user can make a choice at her discretion and pick any song by clicking on the adjacent radio button and then initiate download by pressing “Download Song” button. This will send a download request to the chosen responder node.

At this stage there are two possible scenarios:

- The responder node is able to service the request: It sends the song as a byte stream to the originator after stripping off the user-specific atoms, encrypting its user account information, and computing the integrity. Once the originator receives the song, the client node requests payment from the user.
- The responder is unable to service the request: since it may be offline. In this case an error message (Figure 21) is displayed on the client UI and the song is not transmitted. The originator may request the song from another source. A node which is offline will not communicate any matching requests. On the other hand,

a node may go into offline mode after communicating matches with other nodes. In this case too, an error message shown in Figure 21 is displayed. However, if the responder node happens to come back into the online mode before the originator makes the download request, the originator will be allowed to access content from this responder node.

If the transaction is successful, the user is given two options: continue with payment or purchase the song later as shown in Figure 18. If the user chooses to continue with payment, integrity, and authenticity checks will ensue.



Figure 18: Request Payment for Downloaded Content

Once PTS completes the transaction successfully, a success message informs the user that the transaction is complete as seen in Figure 19.

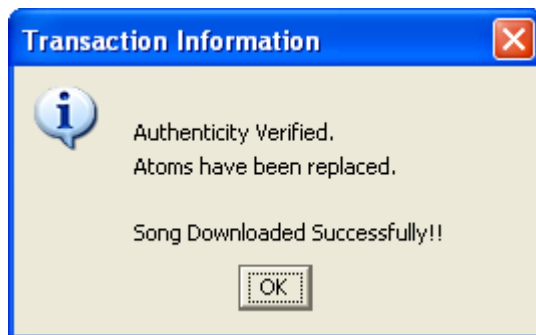


Figure 19: Transaction Success Message

On the other hand there could be issues if the authenticity of the song is incorrect or the responder is not a legal owner. In these cases the originator is informed of the error and she can pick another source. A transaction failure message due to authenticity failure or if the responder is not the legal owner of the content is shown in Figure 20. Figure 21 indicates the error message if the chosen responder node is offline. Figure 22 shows the error message received if the integrity of the byte stream is compromised by any node.

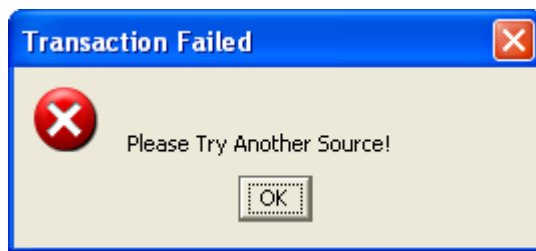


Figure 20: Transaction Failure Message



Figure 21 : Download Error Message



Figure 22 : Integrity Check Error Message

No payment is deducted from the user's PTS account in cases of failed transactions. The second option "Ask me Later" allows the user to purchase the song at a future point in time. Later, when the user clicks on MusicSearch -> PlaySong and chooses the unpaid song, a prompt initiates the payment as explained in the first case. This is shown in Figure 23. If the transaction is successful, a dialog shown in Figure 19 is displayed and the user can play the purchased content. Note, choosing "Ask Me Later" option will not allow the user to play the content.

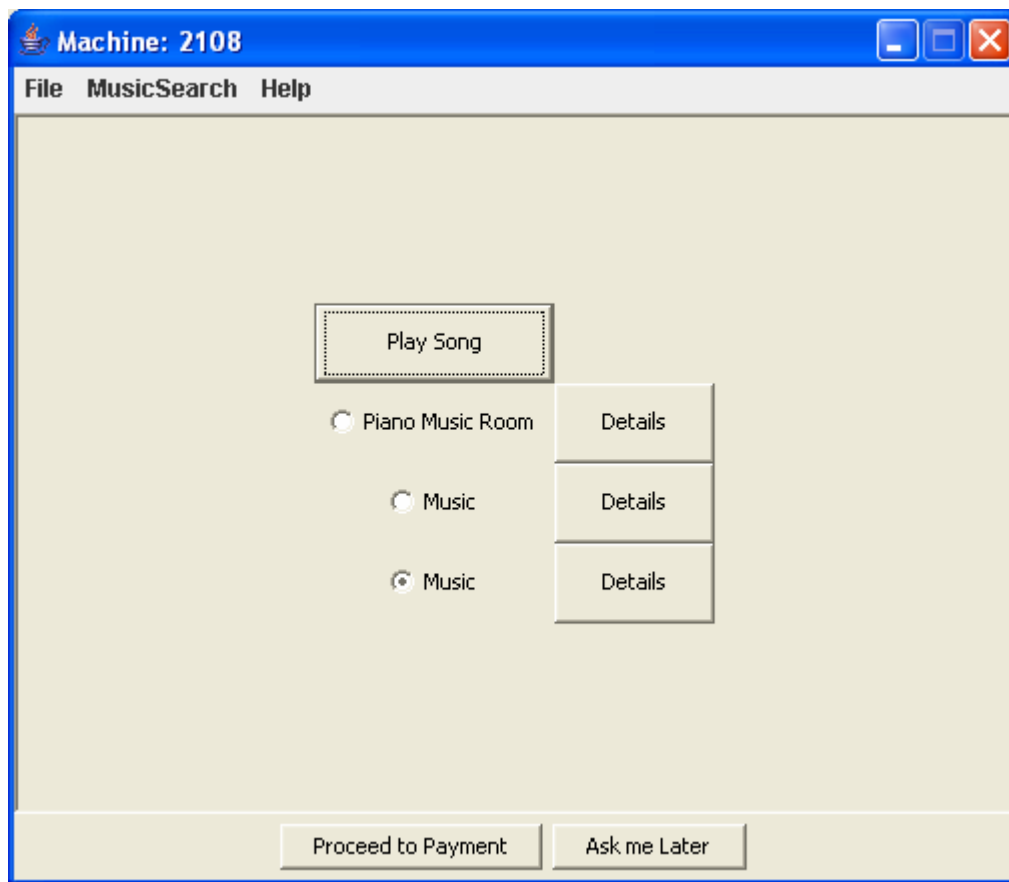


Figure 23 : Payment Required to Play Content

6.4 Playing a Purchased Song

Go to P2PTunesSearch->Play a Song. A list of songs the user has purchased or downloaded is displayed on the UI as seen in Figure 24. Choose the song to be played and click the “Play Song” button.

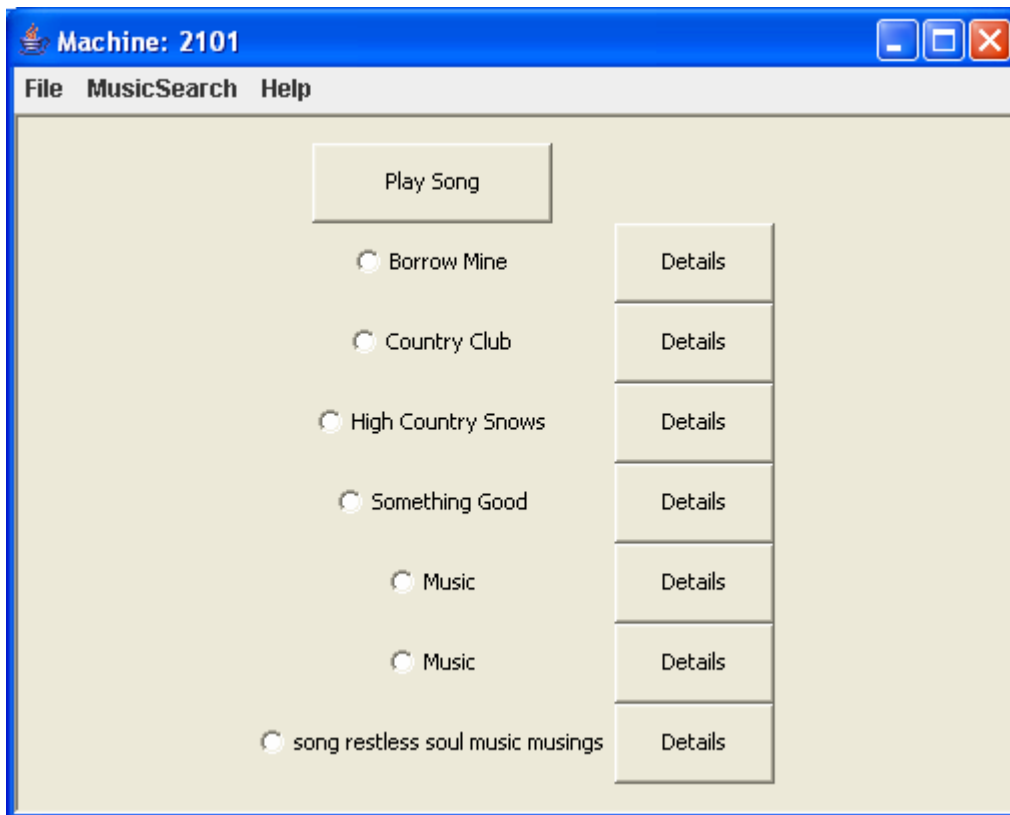


Figure 24 : Downloaded Content on User’s System

If the decryption process is successful, the “audio file” as seen in Figure 25 is displayed indicating the song is successfully playing.



Figure 25 : Message indicates Song Playing

If the user downloaded the song but did not complete the payment, a request for billing will be made as in Figure 23. On successful completion of the payment procedure, the song can be played.

7 Conclusion

The P2PTunes design is geared to efficiently utilize P2P bandwidth for content transactions and effectively delegate as many computation tasks as possible to client nodes. The role of the centralized system is reduced to validating data, performing small operations, and handling financial transactions. This set-up significantly reduces the load on the centralized server since the size of data involved in transactions between client nodes and the PTS is small when compared to the original iTunes system. In addition, P2PTunes provides a forum for users with similar interests in digital content to interact and share content, fostering social networking. Rewarding participating nodes with royalty payments or other forms of incentives such as a point-system wherein accumulation of points in exchange for certain digital content, can generate user interest and could possibly reduce attacks while increasing community participation in the P2PTunes network.

The present centralized system handles multiple functionalities and its bandwidth is occasionally under strain. For instance, swarms of online shoppers armed with iTunes gift cards overwhelmed the iTunes music store over the holidays, prompting error messages and slowdowns in downloads (Montgomery Advertiser, 2007). Additionally, digital distribution of movies, TV shows, etc. requires more bandwidth and hence increases the load on the server. The bandwidth bottleneck can be avoided in the P2PTunes system wherein users in the P2P network can share music over the P2P network instead of approaching the centralized system.

At present there are many security challenges in sharing digital content over a P2P network. The prototype developed here demonstrates it is feasible to extend iTunes over a P2P system while employing sufficient security measures to deter users with malicious intent. However, the model is not impregnable to all security attacks. Reverse-engineering tools such as JHymn can be used to scrub protected files. Users could make use of JHymn and share the resulting unprotected files with each other since it would be a hard task to monitor the type of content exchanged over P2PTunes. Installation of watermarking features such as geId atom (present in m4p file) that are detected by iTunes before playing content is one way of deterring attackers. However, reverse engineering tools such as JHymn take care not to strip information that iTunes detects before playing content. So, this makes it a difficult task for the iTunes player to recognize if the content is in the protected or unprotected format. Note that the above scenario is a challenge faced by the present iTunes implementation as well.

Care has been taken to make P2PTunes reasonably secure and suitable for deployment over P2P networks. The P2PTunes DRM system is strong enough to withstand selected attacks but cannot be deemed totally secure, as is the case for all DRM systems available today.

8 Future Work

The P2PTunes DRM system can incorporate additional security measures and several additional useful features. Some useful features and security measures that can be considered for the future are discussed in this section.

The case of the anti-DRM tool “QTFairUse6” which captures AAC frames after the song has been decrypted by iTunes version 6.0, but before the decoding step and copies it to a file (ipod news,2006) illustrates that the real vulnerability lies at the point in which the song is being decoded to a format understood by a soundcard and not in the encryption scheme. Only a vigilant or trusted Operating System could prevent such an attack. A suggestion for increased security measures would be to rely on trusted hardware and use a trusted operating system. For example integration of P2PTunes with Next Generation Secure Computing Base (NGSCB) offers secure storage of cryptographic keys and secures media-access related processes (Microsoft, 2007).

Some additional features that could enhance P2PTunes further are outlined below:

- Client nodes of P2PTunes can invite other users in their “friends” list to buy digital content from the each other. A user knows her friends’ music or content preferences and can suggest appropriate purchases. This could increase the marketability of content.
- There can be provisions for users to rate digital content and message boards to discuss content and features of P2PTunes. This will increase awareness of P2PTunes as well as that of digital content available for purchase/download. In

addition participation of users in P2PTunes will grow since more users will be aware of P2PTunes functionalities. P2PTunes could well become a widely known forum to discuss viewpoints on digital content and make informed purchases.

- In the present system, the PTS server is able to identify if responder nodes have illegally sent content to service originator's request. This could be used by the PTS to blacklist nodes. Then originators can avoid purchasing content from blacklisted responder nodes.

References

- Androutsellis-Theotokis, Spinellis (2004).
A Survey of Peer-to-peer Content Distribution Technologies. [Electronic Version].
ACM Computing Surveys (CSUR), 36, 335-371.
- Anonymous (2005). Hymn Manual. Retrieved on August 30, 2006
at <http://hymn-project.org/documentation.php>
- Apple Computers Inc. (2006). MPEG-4. The Container for Digital Media.
Retrieved on August 21, 2006
at <http://www.apple.com/quicktime/technologies/mpeg4/>
- Bailes, Templeton (2004). Managing P2P Security. [Electronic Version].
Communications of the ACM, 47, 95-98.
- Bornstein, Neil (2004). Hacking iTunes. Retrieved on April 05, 2007
at <http://www.xml.com/pub/a/2004/11/03/itunes.html>
- Chandak, George, C.E (2005). Can iTunes be weTunes? - Is FairPlay Playing Fair?
[Electronic Version]. 20th BILETA Annual Conference, 2005.
- Daswani, Garcia-Mollia & Yang (2003). Open Problems in Data-Sharing
Peer-to-Peer Systems. [Electronic Version]. Proceedings of the
9th International Conference on Database Theory, 2572, 1-15
- France, Moore, Dreier (2006). Napster. Retrieved on April 07, 2007
at http://reviews.cnet.com/Napster/4505-3669_7-31302303.html
- France, Jasmine (2005). Rhapsody 3.0. Retrieved on April 07, 2007
at http://reviews.cnet.com/Rhapsody_3_0/4505-9239_7-20050753.html?tag=also
- France, Kim (2006). eMusic. Retrieved on April 07, 2007
at http://reviews.cnet.com/eMusic/4505-9240_7-30974740.html?tag=also
- Futureproof (2006). JHymn project. Retrieved on August 24, 2006
at <http://hymn-project.org/jhymndoc/>
- Han, Liu, Xiao et al (2005). A Mutual Anonymous Peer-to-peer Protocol Design.
[Electronic Version]. Proceedings of the 19th IEEE International Parallel
and Distributed Processing Symposium, 68a-68a.

- Indigo group (2005). Fairplay: Effectiveness and Weaknesses of Apple's Digital Rights Management Technology. Retrieved on August 20, 2006 at http://www.simson.net/ref/2005/csci_e-170/p1/indigo.pdf#search=%22indigo%20fairplay%22
- iPod FAQ (2007). iPod: Frequently Asked Questions. Retrieved on April 02, 2007 at <http://docs.info.apple.com/article.html?artnum=60920#faq16>
- iPod news (2006). QTFairUse6 circumvents iTunes DRM. Retrieved on August 30, 2006 at <http://www.ipodnn.com/articles/06/08/30/itunes.drm.circumvented/>
- Jobs (2007). Thoughts on Music. Retrieved on April 04, 2007 at <http://www.apple.com/hotnews/thoughtsonmusic/>
- Kalker, Epema, Hartel et al (2004). Music2Share-Copyright-Compliant Music Sharing in P2P Systems. [Electronic Version]. Proceedings of IEEE, 92, 961-970
- Mannak, Ridder, Keyson (2004). The Human Side of Sharing in Peer-to-Peer Networks. [Electronic Version]. ACM International Conference Proceeding Series, 84, 59-64.
- Microsoft (2007). Next Generation Secure Computing Base. Retrieved on March 17, 2006 at <http://www.microsoft.com/resources/ngscb/default.mspx>
- Microsoft (2006). Peer-to-peer file sharing: Help avoid breaking copyright laws and getting unwanted software. Retrieved on April 03, 2007 at http://www.microsoft.com/athome/security/online/p2p_file_sharing.mspx
- Montgomery Advertiser (2007). iTunes Slowdown. Retrieved on January 03, 2007 at <http://www.montgomeryadvertiser.com/apps/pbcs.dll/frontpage>
- Rodrigues, Liskov, Shriram (2002). Peer-to-peer: The design of a robust P2P system. [Electronic Version]. Proceedings of the 10th workshop on ACM SIGOPS European workshop: beyond the PC, 117-124.
- SourceForge (2006). AtomicParsley. Retrieved on September 16, 2006 at <http://atomicparsley.sourceforge.net/>
- Saroiu, Gummadi, Gribble (2002). A Measurement Study of Peer-to-Peer File Sharing Systems. [Electronic Version]. Multimedia Computing and Networking, 153, 156-70.

- Stamp, Mark (2006). Information Security: Principles and Practice. Wiley-Interscience, ISBN: 0-471-73848-4
- Sun Developer Network (2006). Remote Method Invocation. Retrieved on March 12, 2007 at <http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp>
- Tanin, Nayar, Samet (2005). An efficient nearest neighbor algorithm for P2P settings. [Electronic Version]. Proceedings of the 2005 national conference on Digital government research, 89, 21-28.
- United States Computer Emergency Readiness Team (2005). Risks of File-Sharing Technology. Retrieved on November 03, 2006 at <http://www.us-cert.gov/cas/tips/ST05-007.html>
- Wen, Howard (2005). JHymn Goes Behind Atoms and Apple to Bring DRM-Free Music. Retrieved on September 14, 2006 at <http://osdir.com/Article3823.phtml>
- Wikipedia. Fairplay. Retrieved on August 21, 2006 at <http://en.wikipedia.org/wiki/FairPlay>
- Wikipedia. Peer-to-peer. Retrieved on October 07, 2006 at <http://en.wikipedia.org/wiki/Peer-to-peer>