UNIVERSITY OF BIRMINGHAM

An Online Cinema Application with Digital Rights Management

Author
Yingjing Feng

Supervisor
Eike Ritter

Student ID
1374873

Degree
BSc in Computer Science

Date
April 10, 2015

School of Computer Science
University of Birmingham

Abstract

This project is the result of an integration of the state-of-art digital right management research into the development of the Online Cinema application. The Online Cinema application is a web-based cross-platform application for video sharing, real-time movie playback synchronization and live comments between remote users. Online Cinema enables different users to watch the same media frame of a video at the same time, and uses cryptographic challenge between the client and the server for authentication purposes. The goal is to provide a useful service that at the same time prevents the users to whom the content is being shared from infringing copyright regulations.. The software system is also resilient against known web security attacks such as SQL injection, Man-In-the-Middle Attacks and Cookie Stolen.

Acknowledgments

I'd like to express my thanks and appreciation to my supervisor Dr. Eike Ritter for supporting me and giving me valuable suggestions throughout the project. I would also like to thank my lecturer Ian Batten, and the staff from the IT department in Computer Science school for helping me with the networking and the configuration of virtual server. I would also like to thank my friends and my family for their support.

Contents

Chapter 1

Introduction

The Online Cinema is a multi-platform movie sharing software with media content protection. This project aims to provide an easy-to-use video playback synchronization software with features of a socialized software. The project also embraces the issue of content protection of the movies shared among the Internet, using a cryptographic challenge before media caching and other security deployment, which are transparent to the user experience.

After a user uploads a movie file to the group, a movie-viewing event will be made available to all the members in the group on the date specified by the video owner. During the day when this movie-viewing event is available, when the user enters the movie screening room by clicking the "watch" button, the progress would be first initialized to the current frame of the video, and then the video playback will be synchronized automatically in real time. For example, if a user presses the the pause button, the media viewed by the other members in the same group will be paused immediately; if a user decides to go back to some point in the movie, the group members would see the video of that moment. The user can also see the current online users in the room, and who are controlling the movie playback. The user can choose to hide the additional configuration information for a clearer movie-viewing experience.

Digital rights management is a class of technologies that allow rights owners to set and enforce terms by which people use their intellectual property. The Online Cinema application uses cryptography and other security deployment, which are transparent to the user, to ensure the owners of the video files only share their right to view their owned video copies with other users. The other rights of the video file, such as copying, altering, or selling, would not be shared among the group. This feature makes the software eligible to be widely used on the software market.

1.1 Motivation

This project was motivated from the experience of trying to watch movies synchronously with friends locating in different countries.

Most of the existing video synchronization software only synchronize videos in the mainstream video-sharing websites such as Youtube and Vimeo. The user experience of the software greatly depends on the performance of video rendering in these video-sharing websites, and the accessibility to these video-sharing websites is also subjected to the locations of the users. Besides, the media content uploaded to these mainstream video-sharing websites is also censored, making it extremely hard to share a movie file using these softwares.

The other exiting technological approach for synchronize the movie play-back is to synchronize each frame of the video file. This includes screen-sharing technology such as Skype. Unfortunately this method requires a stable and fast networks environment of both ends throughout the synchronization. The project would reduce the requirement on the networks bandwidth.

This project is developed in hope that technology will eliminate the distance of remote friends. The socialized feature of the software extends the concept of friends in reality, with supports to messaging service among group members and friends.

Chapter 2

Background Research

2.1 Mathematics Primitives in Cryptography Research

A field is an algebraic object with addition and multiplication operations. There are a number of different infinite fields, including the rational numbers, the real numbers and the complex numbers. Cryptography focuses on finite fields to avoid overflow and decision-loss that other infinite fields may cause when the computation is performed on a machine.

A Galois Field $GF(p^n)$ (named after the French mathematician Évariste Galois) is a finite field. In $GF(p^n)$, for any prime integer p and any integer $n >= 1$, there is a unique field with p^n elements. All the results of addition and multiplication are also within this field.

The book [19] provides an easy-to-use algorithm for the addition and multiplication operations. The addition can be implemented as "exclusive-or". The multiplication of two numbers can be done by looking up the entries in the *table of logarithm*, and then looking up the sum of the two corresponding entries in the *table of exponential*. The implementation was used in the project to calculate the secret in 4.6.

2.2 Previous Work in Broadcast Encryption

2.2.1 Secret Sharing Scheme

Shamir [17] has proposed a secret sharing scheme based on $PolynomialInterpolation$ in 1979: given k points in the 2-dimensional plane $(x_1, y_1), \ldots, (x_k, y_k)$ with distinct x_i's, there is one and only one polynomial $q(x)$ of degree $k-1$ such that $q(x_i) = y$ for all i.

Sharmir suggested a secret sharing scheme in which a secret is divided into n parts, giving each participant its own unique part, a subset of k participants are needed in order to reconstruct the secret.

Particularly, Sharmir has adopted a (k, n) threshold scheme for $n = 2k - 1$. The benefits to adopt a (k, n) threshold scheme with $n = 2k - 1$ can be summarized as three:

(i) **Security**: The eligible participants can recover the original key even when $[n/2] = k - 1$ of the n pieces are destroyed.

(ii) **Reliability**: The opponents cannot reconstruct the key even when security breaches expose $[n/2] = k - 1$ of the remaining k pieces.

(iii) **Safety**: Each participant is not given a copy of the secret key to reveal the secret. Each one of them is given a share, which requires other $k - 1$ participants' share.

In [17], the scheme has been proved to work over the field $GF(p)$ for a prime p. [5] has written a document to prove Shamir's secret sharing scheme also works over the field $GF(2^n)$.

2.2.2 Complexity Boundary of Broadcast Encryption Scheme

In [14], the broadcast encryption scheme refers to a system that allows the sender to securely deliver encrypted content to a dynamically changing set of users over an insecure channel, and this scheme

should also be possible to selectively revoke (i.e. exclude) a certain subset of users from receiving the data.

The paper also proposed the concept of *resilient* to a set S: If for every subset T that does not intersect with S , no eavesdropper, that has all secrets associated with members of S, can obtain the secret common to T. A scheme that is called $k - resilient$ if it is resilient to any set $S \subset U$ of size k.

In Fiat's work, the optimization is on the number of keys associated with each user, the number of transmissions used by the center, and the computation effort involved in retrieving the common key by the members of the privileged class. Fiat proved that for a k-resilient scheme, the minimal storage complexity is that each user to store $O(k \log nw)$ keys and the center to broadcast $O(k^3 \log n)$ messages.

2.2.3 A Boradcast Encryption Scheme based on Sharmir's Secret Sharing

Berkovits[11] combined the works of Sharmir's and Fiat's, and suggested a broadcast scheme requiring communication, encryption and decryption complexity linear in the number of recipients $(N - r)$ for N users with r revoked users included.

In his scheme, secret share S from a finite field $GF(p)$ is encoded as the constant term of some polynomial P. The scheme defines k eligible users and j non-eligible users ($j >= 0$). Each user is assigned a pseudo-share, i.e. point (x_i, y_i) in the 2-dimensional space, and all these points are made pairwise distinct. To transfer a message s, the sender selects a random polynomial P of degree $k + j$ that passes through the points corresponding to all the eligible users, but not the points of non-eligible users. Then the server can broadcast $k + j$ randomly chosen shares that are all distinct with all the users' points. Each user then add his point to the $k + j$ common shares.

This scheme ensures that only the k eligible users can recover the polynomial and get the share S, and the other j non-eligible users can not recover S. This broadcast encryption scheme has a merit that the k eligible users can not collude to get the secret S by combining all their k points, as the polynomial required at least $k + j + 1$ points to recovered the secret.

Chapter 3

Specification

The elicited general requirements, both functional and non-functional, as well as the definition of the the relationships between the user and the system via the use-case diagram were critical steps towards a more detailed design of the system. The elaboration of refined definitions is discussed in the following chapters.

3.1 General Requirements

When the project started, we first looked into general requirements. Then we developed more detailed functional requirements and non-functional requirements.

- A group management system where the users can register, login, create group, add other users in the group.
- A video file sharing system that ensures the video files can not be accessed from outside the system.
- A real-time sharing progress system that allows user to synchronize the progress of the film in their devices.
- The tool should be accessible in a global networks environment.

3.2 Functional Requirements

1. User Management
   1.1. User can register with username, password and image provided.
   1.2. User can login with matched username and password.
   1.3. User can log out the system.
2. Friend Management
   2.1. User can see the photo of a friend.
   2.2. User can search for a friend in the system.
   2.3. User can search for a non-friend user in the system.
   2.4. User can send friend request to a non-friend user, and this user will automatically become the friend of the user.
   2.5. User can unfriend a user, while their message is still persistent in the system.
   2.6. User can send a message to a friend.
3. Group Management
   3.1. User can create a group with a group name.

3.2. User can invite a friend to join the group he belongs to, provided that the group has less than 12 members.

3.3. User can leave a group.

3.4. User can create a movie event of group by uploading a movie to the group and specify the date of the event.

3.5. User in the group can choose to like a currently available movie.

3.6. User can get the information of the movie, including movie name, movie brief, movie and which members like the movies.

3.7. User in a group need to wait until the current movie event to end, in order to create another new film event.

4. Current Movie Management

4.1. User in the group can watch the shared movie of this group which is currently available.

4.2. All the users in the same group will watch the synchronized frame.

4.3. User can see which group members are also watching the movie.

4.4. User can chat with other group members while watching the movie.

4.5. User can seek the movie by moving the scroller bar.

4.6. User can control the volume of the video.

4.7. User can choose to hide the unnecessary configuration in the interface.

3.3 Non-Functional Requirements

Performance The application should have a stable performance while it is running.

Scalability The application should be easy for downloading and installation.

Portability The application should be available for the mainstream operating systems, i.e. Windows, Linux and Mac OS X.

Availability The server should be available for 24/7 hours when it is functioning. It should be accessible in locations with networks access.

Reliability It should be 95% time that the latency between each user is less than 3 seconds.

Maintainability The unnecessary code dependencies between different components in the software should be minimized.

Security The web server should resist SQL injection and other common web attacks.

Usability The application should be easily understood and used for the purpose of video synchronizing with friends.

Ethicality The media content must only be accessible within our software and only when film event is not ended. The video file should only retain in the memory, and never be saved to the disk.

3.4 Use Case

A use case diagram can be used to illustrate the relationship between users.

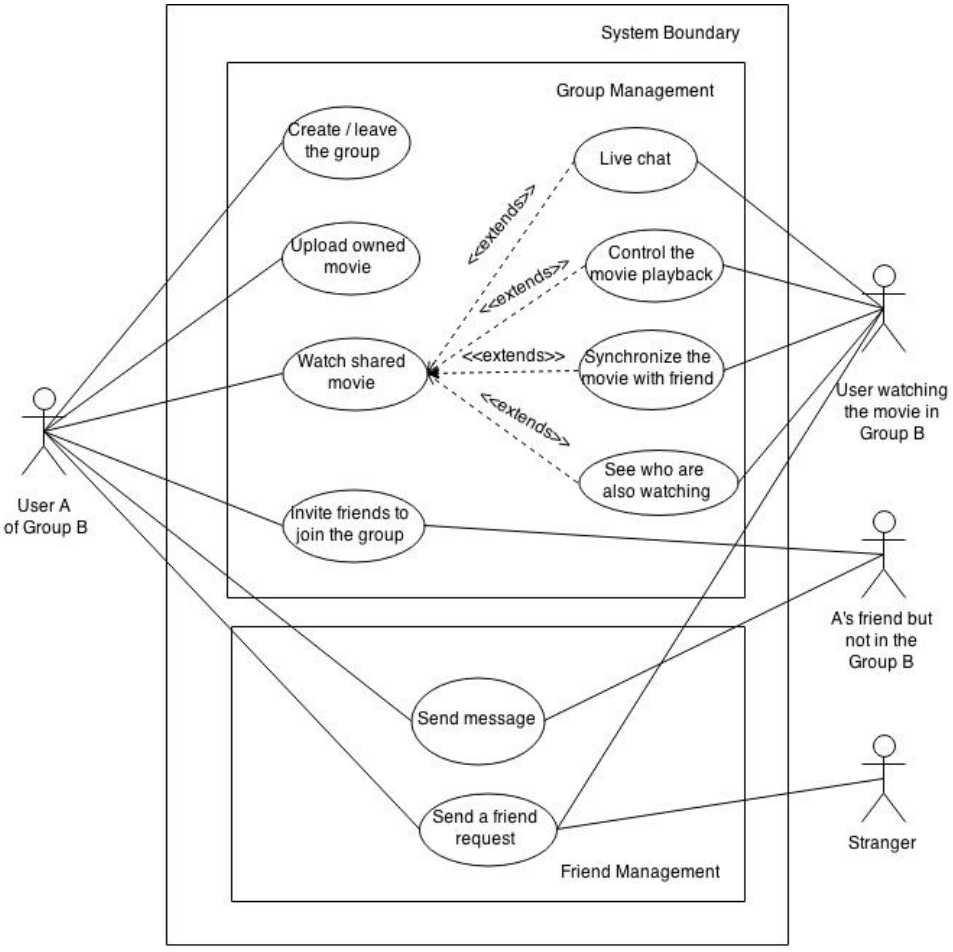


Figure 3.1: Use Case Diagram

Chapter 4

Digital Rights Management

4.1 Background

According to [10], a system with **digital rights management** (DRM) can control the content access and actions that can be performed by users (both human and machine) of the digital content.

A system that fully implements DRM should meet the various required levels of security, from application layer, the operating system layer to the hardware layer. Unfortunately, according to [10], currently there is no such system that provides all these levels of security.

Reviewing the available stand-alone[1] media content distribution software in the market, there is not a piece of software that can stand the criticism on the ineffectiveness on DRM, as none of them can prevent sophisticate attacks on the hardware and operating system level. This is exactly like what [12] had claimed, a software-based scheme can never fulfill all the requirements in DRM thoroughly, because the software must contain all the decryption keys or other necessary information in order to decrypt the file, which an attacker can also extract from the software.

A technical report [16] has made an analysis on the security framework. At the time when the report was written, the DRM of Netflix depended on a Microsoft proprietary DRM component and a Microsoft Silverlight application framework. The critical step to control DRM-encoded video and audio files is during the process of license acquisition. After observing on the HTTPS requests for the license and responses from the server, the author believed that "the Silverlight plug-in is generating a new public/private key pair every time it requests a new license, using perhaps a combination of its individualization code, the device ID as recorded by Netflix, and a time-stamp."

4.2 Overview

After evaluation on research status in DRM, it was decided that the scope for DRM in our software are two: (i) to prevent user piracy over the application layer; and (ii) to prevent the networks layer.

We have also identified the goal of the adversary as gaining the media content uploaded by a group member and stored in the server.

When we designed the scheme for DRM, the first proposal was to save the encrypted video file to the disk, and the decryption is performed every time the user attempts to watch encrypted video using our application. This was excluded immediately as we examined the JavaFX media player in our project only support reading from a file, but not from a byte stream, since it is more difficult to seek[2] from a byte stream. Apart from that, the decryption key to the encrypted video file is both non-revocable and non-renewable. Once it is released, the video file can be completely decrypted to a version that can be redistributed on the market.

[16] indicated that the DRM scheme within Netflix is very likely based on the communication with a license serve upon issuing a request of media caching to the server, and the SSL public/private keys used for communication are changed in every request based on a combination of some artifacts and a

[1] not bound with the support of the hardware or operating system
[2] jump to the specific time of the video stream

time-stamp. Such information helped to design the DRM scheme in our application, and the reasoning process will be explained in this chapter.

This chapter will first introduce the adversary model in 4.3. The goals in DRM were established on the basis of the proposed adversary model. Then we will explain the process of choosing the best key management solution out of four proposals. Finally, the corresponding algorithm used to calculate the shared secret. The application of this scheme will be presented in 5.3.3.

4.3 Adversary Model

In order to achieve the goal, the adversary may use the power on the three levels as described below:

(i) **Web Application Level**

At the web application level, the adversary can use web application security testing tools. For example, using Burp Suite Toolkit [2], the adversary can scan the site in order to get all the web page and the corresponding RESTful API. He can also generate arbitrary HTTP requests to the server. For example, in order to make the request look like it is from the application, the adversary can set the parameters in the content of a POST request, and modify the User-Agent field in the request header. The adversary can also perform typical web application attacks, such as performing SQL injection on the input, creating false identity and making unauthorized access to the resource on the server.

(ii) **Application Logic Level**

At the application logic level, the adversary can become a qualified user of the application.The adversary can issue a valid video file download request, and get a challenge issued by the server. However, if the adversary is not using our application, he has to solve the challenge and provide the answer back, in order to get the response containing movie file.

(iii) **Networks Level**

At the networks level, the adversary can monitor, intercept or modify all the network traffic.To address this problem, all the requests and responses between the client and the server are based on a SSL layer with two-way mutual authentication. We assume that the adversary cannot interpret the content of encrypted traffic.

The protection against the attack of the operating system and reverse-engineering analysis are not implemented in the project. These topics will be discussed in 10.1.

4.4 Goals

The goals of security in this project were made on the basis of the adversary model listed above.

(i) **Web Application Security**

The web application should be resilient to common web attacks. The value of the session cookie should be unpredictable to the adversary, and the cookie should be encrypted. The web application should reveal as few information as possible. This information includes the database design and the file structure of the server.

(ii) **Networks Security**

To protect against the eavesdroppers between the server and the client, the requests and responses should have mutual authentication, and the traffic between two ends should all be encrypted.

(iii) **Logical Security**

With only the information provided by the challenge, or only the client's stored key, the client should not be able to reveal the secret which establishes a valid answer for a movie-file download response. The secret could be calculated only when the adversary combine the stored key and the information in the challenge.

(iv) **Minimal Message Exchange**

As the message exchange is in an insecure networks environment, the project aims to build a secret sharing scheme with minimal transmission complexity and the storage complexity on both of the server side and the client side.

Additionally, a goal of (iv) **minimal message exchange** is introduced. The goal of **logical security** should still remain true even if the adversary is able to copy the private key corresponding to the user certificate and use it in an external program, which breaks the **networks security** goal, the cryptographic challenge would still be reasonably hard for an external program to forge the answer and assess the media content, even for an eligible user in the application.

4.5 Key Management on Secret Sharing

The key to access the video file should be constructed dynamically, so the client has to ask the server for the key. The client program itself must also be able to calculate the key based on the information that server provides. Here a key management scheme is required in order to share a secret between the server and the user, where the secret is the key to access the video file.

In our application, the set of eligible participants is changing dynamically. When a user leaves a group, the key of a group to get the secret shared by the subset should also be also updated, in order to prevent that user from keeping on using the key to get the content of the group.

There are two measurements to decide whether the key management is good or not: (a) the number of keys stored in both the server and the client sides; and (b) the communication between the server and the client when the key is updated. In an unsecure networks, we hope (a) and (b) to be minimized.

4.5.1 Proposals

We define a model with N_g groups and N_u users. To simplify, there are p users in each group. In average, a user will join q groups. The N_u and N_g can both be large, for example, $N_u = 10000$ and $N_g = 5000$. In our application, there is a constraint on the number of members in a group should be less than 12, i.e. $p <= 12$. q is also small compared to N_u and N_g, e.g. $q = 10$.

(i) **Each user is assigned a unique secret key.** In order to transfer a secret of a group, the server should use the public key of each user's to encrypt every single secret.

Key Storage: This algorithm requires key storage complexity of N_u in server, and N_u in client program. An encryption and decryption is required each time.

Key Update: No key is bound with the group. When the user leaves a group, there is no action that the server needs to take.

(ii) **Each group is assigned a unique secret key, and every user in the group owns a copy of group's secret key.** A user needs to store the secrets of the groups which he belongs to. Whenever a user joins a group, he is given the secret key of the group. In order to transfer a secret of a group, the server uses the public key of the corresponding group's to encrypt the secret.

Key Storage: The server needs to store N_g public keys corresponding to each group. Each user stores keys of all the groups he belongs to, i.e. q keys. The numbers of keys stored in each client sums up to qN_u.

Key Update: When the user joins or joins the group, it is necessary for the group to update and inform the other members of the group of the updated private and public key pairs, so that user will no longer be able to get the secret key of the group. The communication complexity for the updated group is p each time.

(iii) **Each user is assigned all the keys corresponding to all the possible subsets of the users.** A group will be a subset of all the users. To broadcast a message to a group, the server will look up the key of the subset which contains all the group members, and use the public key of that subset to encrypt the message.

Key Storage: Each user needs to store 2^{N_u} keys, and the total number of keys stored in each client sum up to $N_u 2^{N_u}$. The server accordingly also needs to store $N_u 2^{N_u}$ public keys corresponding to every possible subsets of the users.

Key Update: When the user U_i leaves a group G_j, there is no need to inform the users about the update of the key. When the server needs to broadcast next message, the server can use the key of the new subset $(G_j - U_i)$ for encryption and broadcast the encrypted message.

(iv) **The secret of each group is a polynomial of degree k, and each user is assigned a point of 2-dimensional.** Upon request, the user would get k pseudo-share points of the polynomial. This scheme is based on Berkovits's scheme introduced in 2.2.3, with the j value proposed in the scheme set to 0. The difference is that each user has exactly one unique identity represented by an x value, and its identity is not associated to the group. So the server only needs to store the group secret, i.e. the constant term of the polynomial of the group, and the points of users.

Key Storage: Storage complexity of each user is just 1, so the total number of keys stored in each client is N_u. The server needs to store N_g public keys corresponding to N_g groups.

Key Update: When the user joins or leaves a group, the key, i.e. the constant term of the polynomial equation of the group, requires an update. But this mechanism does not need to inform the other member of the key change as in the proposal (ii), as the updated k points will be included in the next broadcast message.

4.5.2 Decision

By considering the complexity and the update of the key, we adopt the proposal (iv), as it only demands linear storage complexity in both the client and the server side. It also does not require to inform the update of key to users immediately when there is a change on the groups. The next session explains how the secret is calculated based on the proposal (iv).

4.6 Secret Calculation

The K-out-of-N secret sharing scheme was proposed by [17]. The complexity boundary of a broadcast encryption scheme was studied formally by [14]. [11] had proposed a broadcast encryption scheme based on [17]'s scheme. The implementation over GF(256) field[3] is based on the implementation of [5]. Their works were introduced in the background chapter.

[3]Galois Field

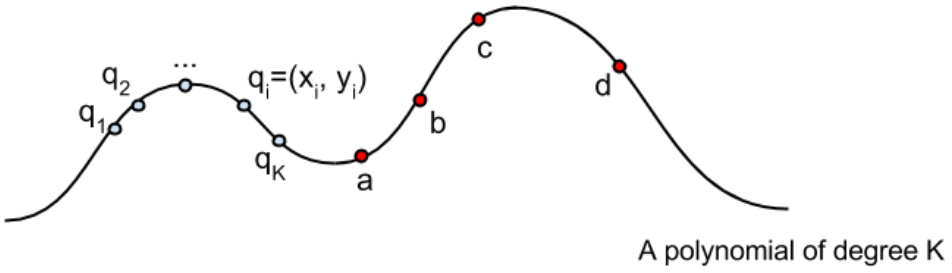


Figure 4.1: Illustration of the Broadcast Encryption based on Sharmir's Scheme

We used $(k+1)$ points to define a polynomial P_k as the polynomial in Figure 4.1. In the scheme, each recipient will get the same polynomial P_k that passes through k common points shown as the blue points q_1, q_2, ..., q_k on the figure.

Each group member will get a distinct point of the same polynomial shown as the red points on the curve. For example, the first group member will get point **a**, and the second member will get the point **b**, but they will never get the same point as any other in the group.

Each eligible recipient will get the k common points and their own point to calculate the group secret using the algorithm shown below. This scheme introduced minimal key management complexity, and the non-eligible recipients can not deduce the secret.

Algorithm 1 Algorithm used to compute the y_{k+1} and secret S

Preconditions:
(a) A Lagrange interpolation polynomial is denoted as P_k is known to both the server and the client.
(b) S as the constant term of P_k. It is known to the server only before secret sharing.
(c) The server chose distinct nonzero and random $x_1, \cdots, x_k \in GF(256)$, and $y_1, \cdots, y_k \in GF(256)$. Each (x_i, y_i) represents a point p_i on the Lagrange Interpolation P_k. They are known to the server only before secret sharing.
(d) The x_{k+1} is known to both the server and the client.

The server can calculate of y_{k+1} of P_k, provided with (a)(b)(c)(d) under GF(256).

$$y_{k+1} = \left[S + \sum_{i \in I, i \neq (k+1)} y_i \prod_{j \in I, j \neq i} x_j (x_i - x_j)^{-1} \right] \left[\prod_{i \in I, i \neq (k+1)} x_i^{-1} (x_i - x_{k+1}) \right]$$

The client can calculate of the constant term S of P_k, provided with (a)(c)(d) and y_{k+1} under GF(256).

$$S = \left[\sum_{i \in I, i \neq (k+1)} y_i \prod_{j \in I, j \neq i} x_j (x_i - x_j)^{-1} \right] \left[y_{k+j+1} \prod_{i \in I, i \neq (k+1)} x_i (x_i - x_{k+1}^{-1}) \right]$$

Because:

$$\left[\sum_{i \in I, i \neq (k+1)} y_i \prod_{j \in I, j \neq i} x_j (x_i - x_j)^{-1} \right] \left[y_{k+j+1} \prod_{i \in I, i \neq (k+1)} x_i (x_i - x_{k+1})^{-1} \right]$$

$$= \sum_{i \in I, i \neq (k+1)} y_i \prod_{j \in I, j \neq i} x_j (x_i - x_j)^{-1} - \sum_{i \in I, i \neq (k+1)} y_i \prod_{j \in I, j \neq i} x_j (x_i - x_j)^{-1} + S$$

$$= S$$

Chapter 5

Software Design

5.1 Client-Server Architecture Analysis

The use case on the specification in Chapter 3 was turned into more specific functional components in the client program. The layered diagram would be used to design the system architecture as well as the user interface. The system architecture would be described in this chapter, and the user interface would be described in the software implementation in Chapter 6.2.2.

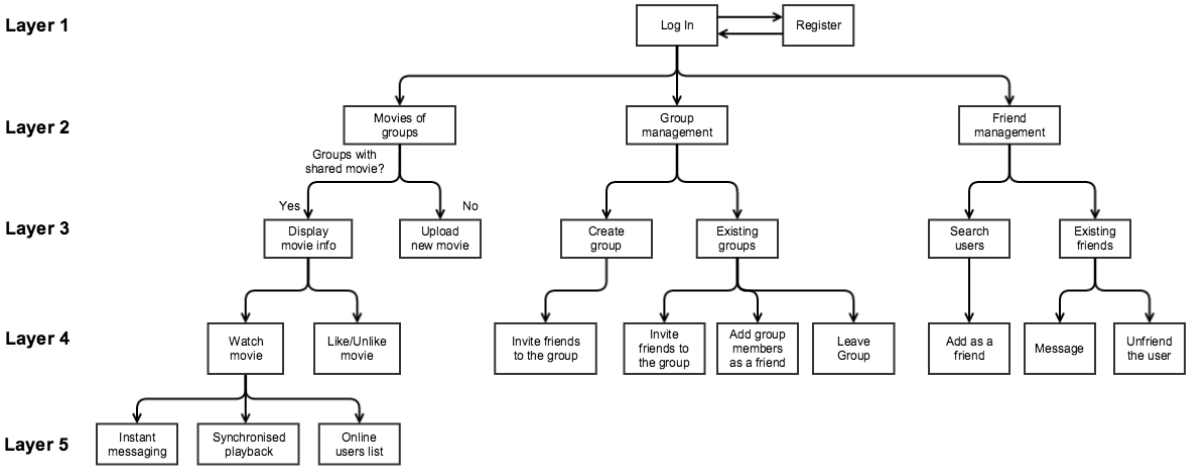


Figure 5.1: Functional Components in Client Side

5.2 Programming Language and Technologies Used

5.2.1 PHP

PHP was used for the server side as it has many good open-source projects developed in PHP to learn from. It's very popular for the server side web application development, and the servers hosting PHP are also widely available, with a lower price compared to ASP and Python which are supported by less hosted servers. The mature language also have many extension libraries to perform a variety of tasks, which speeds up the development process. PHP also has a great database flexibility that it can work well along with MySQL, PostgreSQL, SQLite, etc.

5.2.2 Laravel

Laravel framework was adopted with the PHP development as it has many features to make PHP development easier. The external packages could be all done by composer manager. It also provides

functionalities such as the aritisan command-line program to complete different tasks, database migration, RESTful Routing and Object Relational Mapping. Besides, the popularity of the Laravel framework provide many online resources to learn from.

An application built by Laravel framework is based on Model-View-Controller (MVC) design pattern. Each model is associated with a table in the database under the scheme of Object Relational Mapping, which was explained in the section of database in 5.3.5.2. The view of a web application in this project is implemented in the client side, so the view component is substituted by the JSON string in the response. The controller the connector between model and view. When a request comes, the routing table will find the controller which contains the correct function that handles that request.

5.2.3 Ratchet

Ratchet is a PHP library which implements the WebSockets, and is used in this project to built the components of layered 5. Ratchet provides APIs of *onOpen*, *onMessage* and *onClose*, that can allow the programmer to specify the actions under the situations when a connection opens, a message comes and a connection closes.

5.2.4 JavaFX

The client application is developed in Java as it provides with Object-Oriented Programming which is suitable for the requirement. Each components can be represented in classes. A JavaFX project is shipped with Java 8, which means easy installation, and also fully compatible with other Java APIs. It provides the SceneBuilder tool to design the user interface with JavaFX built-in UI controls.

The media player used in the project is JavaFX media framework. There are some options of media players that can be used to build on a Java application. However, JavaFX has a stable performance and good portability for multiple operating system.

- **Java Media Framework API (JMF)**

  This library is an official Java library that builds on Java 2 to support time-based media such as audio and video. It also supports playback on multiple media formats. However, the last review on JMF project was in 2003, and no update supports were available after that.

- **VLCj**

  This library is the Java framework for using the VLC program in a Java application. It requires VLC programs and other dependencies installed in the client side, and the environment for each operating system varies, which introduces a major problem in the deployment of the Java application.

- **JavaFX Media**

  JavaFX is a set of graphics and media packages for creating and delivering cross-platform rich internet applications. It provides Java APIs for media playing which only requires Java SE Run-time Environment. The Oracle document also claims the media engine as "stable" and "low-latency".

5.2.5 Stunnel

Stunnel is an application designed to work as an SSL encryption wrapper between remote client and local or remote server. It can run as a daemon program in the server side. As the project uses **Ratchet** PHP library to implement the WebSocket application which doesn't support direct SSL requests, **Stunnel** is used in the server to unwrap the WSS encrypted request to plain WebSocket request.

5.3 Server Design

As this project involves a great amount of communication between the client and the server, the first thing to think about in the design of system is the networking model.

We classified the network requirements into two types:

(i) Request-response networks in layer 1, 2, 3 and 4.

(ii) Real-time networks for layer 5.

Additionally, the requests related to media content caching was handled separately. Thus, the architecture are made of three main components.

In this section, we look into the server design of these three components. The component of general requests and responses and the real-time networks would be discussed in 5.3.1 and 5.3.2 respectively. The requests and responses related to media content caching would be presented, together with other security deployment that ensures the web application security. The database design of the system is placed at last.

5.3.1 General HTTPS Requests and Responses

The functional components specified in layer 1, 2, 3 and 4 can be implemented in traditional request-response networks. The client side asks the server to deal with the system logic including data retrieval and update of the database using HTTPS requests and responses. In the server side, a routing table should be implemented to specify the correct functions for different URLs without exposing the file directory of the server. The routing table can be seen in "/app/routes.php" in the code.

5.3.2 Real-time Networks

The real-time networks here refers to soft real-time that subjects to the normal latency in the user's networks environment. There are three proposal for implementing the real-time networks, and the decision was made by comparing the three proposals.

5.3.2.1 Proposal

(i) **Request-Response Model**

As the application uses Apache server and PHP scripting language, it is natural to come up with a naive solution to use traditional Request-Response model for all the traffic between the server and the client. This model works as a general way of communication. However, in order to provide high-quality video synchronization and live-chatting services, it requires intensive real-time communication between server and client. The Request-Response model is not satisfactory as the server is always passively awaiting for the clients, and the client has to probe the server continueously to check any updated status of the media playback, or new messages of the chatting.

(ii) **Broadcast Model Based on UDP Multi-cast**

In UDP Multi-cast, the client in the multi-cast group can send packets to all the other clients in the same group without going through the central server. The other users can get the updated information in time. Such model is an implementation for the prototyping version. However, UDP socket establishment is constrained by the current networks environment. Also, UDP Multi-cast needs to join the UDP multi-cast group by specifying the global IP address of the group. In the global networks, the lossy feature of UDP socket and the difficulty on the routing to the multicast address together make the UDP Multi-cast approach unrealistic.

(iii) **Publish–Subscribe Model based on WebSocket**

WebSocket Protocol is a protocol providing full-duplex communication channels over a single TCP connection, which was standardized by the IETF as RFC 6455 [13] in 2011. The protocol consists of an opening handshake followed by basic message framing, layered over TCP. The goal of the WebSocket protocol is to provide a mechanism for browser-based applications that need two-way communication with servers that does not rely on opening multiple HTTP connections.

Using WebSocket protocol, once the connection between the client and the web server is established, a persistent connection will stay connected until either the client or the web server decides to close this connection. With this open connection, the client or server can send a message at any given time to the other. The running server application can keep track of all the connections, and communicate with any number of open connections at any given time. The web programming implemented by WebSocket will be event-driven and stateful.

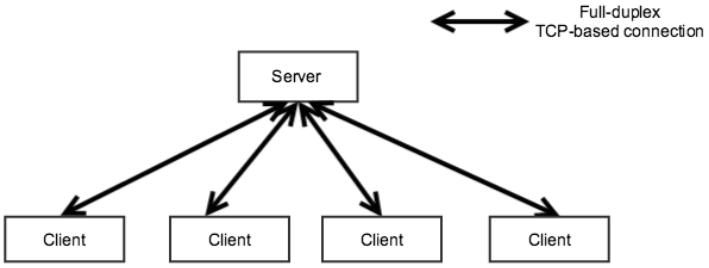


Figure 5.2: WebSocket

5.3.2.2 Decision

Out of the three proposals, only the proposal (3) which uses the WebSocket protocol for communication can satisfy the soft real-time networks requirement. The real-time networks of each event can be established based on the full-duplex WebSocket connection as shown below.

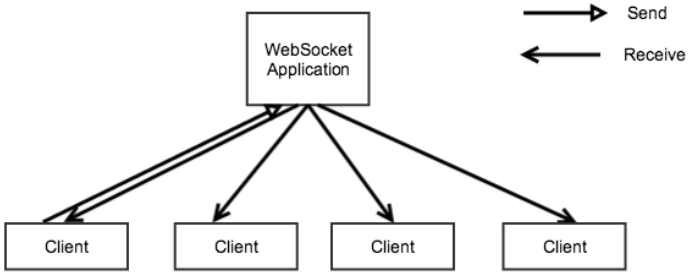


Figure 5.3: Using WebSocket in the Project

When there is an updated in one user, it informs the central server, and the server will broadcast the updated information to all the other users immediately. JSON strings are used to transfer data between the client and the server. Each JSON string has a uniform format for different types of message: **message_type**, **message_text**, **movie_time**, **message_time**, **uname**, **groupname**. It can carry all three types of messages available in the group:

(i) Updating the playback of the shared movie.

(ii) Sending and receiving messages of the group instantly.

(iii) Updating the list of the online users synchronously.

The Stunnel program (5.2.4) is used to strip the SSL layer of the WebSocket, which can then be handled in the server websocket endpoint.

The server websocket endpoint used **SplObjectStorage** in PHP language, which is a hashing table using objects as the key value. The information of each group in the server is stored in using the **SplObjectStorage** with each group's name as the key.

This data storage in the real-time networks does not preserve for a long term, so it does not need to write/read the disk. As each movie event runs for 24 hours, and the storage in the memory will be cleaned after the movie event became invalid, so the memory which stores the configuration of the group can be freed.

5.3.3 Movie Chaching Requests and Responses

For the movie download request, we would use a concept of **challenge** to prevent the media content from being accessed illegally, i.e. accessing the media content uploaded by the other users without using the Online Cinema application. The challenge of the scheme is used for proving the eligibility of a participant. A participant is trusted if it can solve the challenge. Therefore, the role of **challenge issuer** in the server side, and the role of **challenge solver** are defined in the challenge. A challenge issuer issue a challenge by releasing all the information required to solve a challenge. A challenge issuer needs to calculate the answer and return it to the challenge issuer, in order to prove itself as an eligible participant in the scheme.

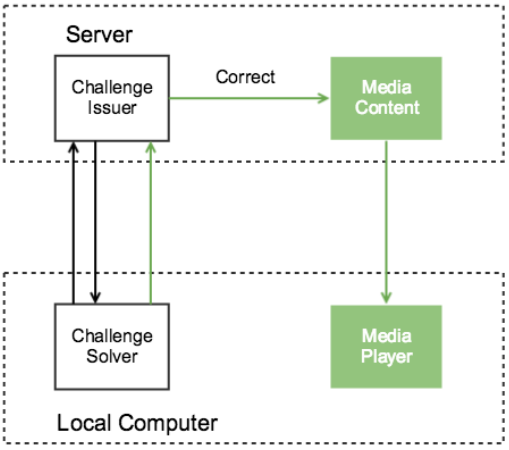


Figure 5.4: Challenge Issuer and Challenge Solver

In the figure, the green content is the critical content which our scheme aim to protect. In the following we will show how the green arrow can be protected from the middle man in the networks from accessing it. The implementation is based on the key management strategy introduced in 4.5.2. The secret calculation algorithm was introduced in Section 4.6.

5.3.3.1 Initialization

The system set the value of k for the Lagrange Interpolate Polynomial P_k in both of the the client and the server sides.

Each client program stores the x_{k+1} of the $(k+1)$th point as the unique identifier in the binary. This x_{k+1} value should be kept safe. The application stores it in the binary code shipped with each piece of software. Alternative approaches include the system sends an email with x_{k+1} stored in an encrypted file,

whose decryption key is in the binary code of the software, and the user needs to apply the encrypted file to the program.

When a group is set up, the server generates k points $(x_1, y_1), \cdots, (x_k, y_k)$ and the secret S randomly for the group. These information is stored in the database. Each group has a limitation of $k = 12$ members.

5.3.3.2 Sequence of the video caching

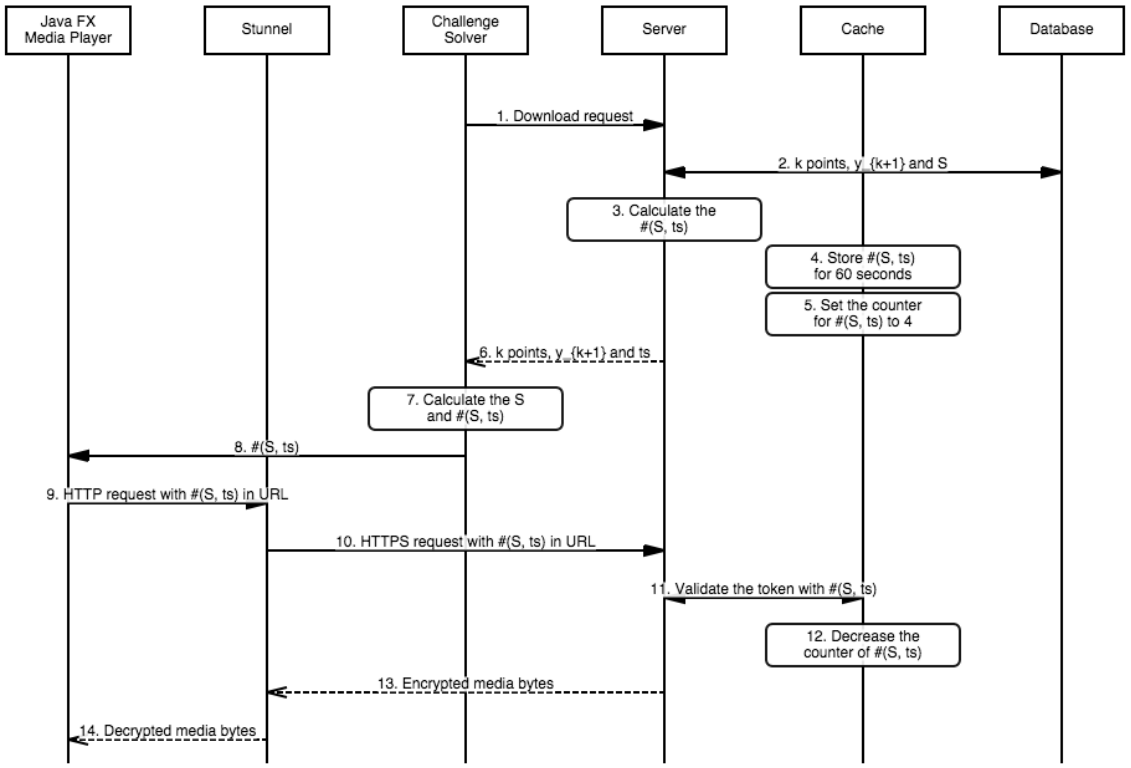


Figure 5.5: Traffics of Video Caching for Eligible Clients

When the user presses the "watch" button in a movie pane, the challenge solver in the client side first issues a download request to the server. The server validates the user's identity, and fetches k points $(x_1, y_1), \cdots, (x_k, y_k)$ of the user's group's and y_{k+1} corresponding to user's x_{k+1} from the database. The server then hashes the secret S with the current time-stamp ts. At the meantime, he server also caches the hash value $\#(S, t_s)$ in the server for 60 seconds, with a counter set to 4[1]. The server responses with a cryptographic challenge, with information of k points $(x_1, y_1), \cdots, (x_k, y_k)$, and y_{k+1} corresponding to client's x_{k+1}, and the time-stamp t_s.

The challenge solver combines the x_{k+1} in its storage, with y_{k+1} to get the $(k+1)$th point(x_{k+1}, y_{k+1}), and the other k points $(x_1, y_1), \cdots, (x_k, y_k)$ received from the server. Using the algorithm in 4.6, the challenge solver can calculate the secret S with these $(k+1)$ points of the P_k, and hashes the secret S with the time-stamp ts to get the token stated in URL.

The JavaFX Media Library use a HTTP URL as a video file source to initialize the Media Player. Since there is a problem that the JavaFX Media Library can not analyze a HTTPS URL, and we want the token and the media bytes to be encrypted on the networks, so a SSL wrapper Stunnel is used in the client side. The JavaFX Media Library sends a HTTP request to the Stunnel program listening on the

[1]explained in the following section

port 8888 on local machine with the token in the URL, and the Stunnel will encrypt the message, and forward it to the web server. [2] The Stunnel then wraps the URL on a SSL layer. [3]

When the server receives the answer, it will compare the token with the stored $\#(S, t_s)$ in the cache. If the two hash value match, it will send back the media bytes in a HTTPS response, the Stunnel unwraps the HTTPS response to an HTTP response, and forward it to the JavaFX Media. The Media framework then begins caching the content in the memory of the client. The server also decreases the counter that keeps a record of how many times left for using $\#(S, t_s)$, and returns the video file in an HTTPS response. When the value of counter of a token is decreased to 0, the token is no longer available.

In this project, we used two characters with two hexadecimal numbers to represent a 8-bit number which denotes the $GF(256)$[4]. In order to get a longer key such as a 96-character key, we calculate 12 different $S_1, S_2, \ldots, S_{12}$ separately, and concatenate them to form a string S of length 96.

5.3.3.3 Rationale

The scheme ensures that all the traffics between the server and the client is on the SSL layer. The media bytes in a HTTPS response are all encrypted. The token in HTTPS request URL is also encrypted, as an attacker who views the traffic between the server and the client can only view host server of the request.

The time-stamp also prevents the replay attack even if the attacker somehow reads the HTTPS URL. Apart from the time limit, there is also a counter for each available token in the cache. As the JavaFX will issue one HEAD request and three GET requests while caching a media, the counter value is set to 4. A token should be invalid after 4 HTTPS requests containing this token in the URL are presented.

5.3.4 Security Deployment

To make sure the cryptographic challenge works properly, the system should also be resilient to other attacks. The application was capable to prevent the five attacks discussed below.

5.3.4.1 Man-In-The-Middle Attack

The application uses mutual authentication on both the server and the client sides. The server has stored all the clients' certificates and the server's own private key. Each piece of the software in the client side has already contained the server's certificate and its own private key. The establishment of a SSL session involves a SSL handshake process that authenticates both sides. In this project, all the requests use HTTPS and WSS protocols, which refer to HTTP protocol and WebSocket protocol on the SSL layer respectively.

5.3.4.2 SQL injection

According to the Laravel 4.2 official document [4], the Laravel query builder uses PDO parameter binding to protect the application against SQL injection attacks. Therefore, the input strings being passed as bindings are automatically sanitized.

[2] An example HTTP URL sent by the JavaFX Media Library will be `http://127.0.0.1:8888/download/{groupname}/{token}`.

[3] An HTTPS URL forwarded by the Stunnel would be `https://vm-yxf373.cs.bham.ac.uk:8078/download/{groupname}/{token}`.

[4] 8 bits can store 2^8 values

5.3.4.3 Exposure of File or Directory

The routing of RESTful requests to its corresponding functions are done by a routing table specified in the **/app/route.php**. For every request, it first goes through an authentication function to identify the user identity, then it will be routed to the corresponding functions in the controllers. The requests with URL not specified in the routing table will get a response with an page-not-found error. Thus, for the file directory where the files and the codes of system logic are stored, it will be very difficult for an attacker to get access by URL guessing.

5.3.4.4 Session Hijacking

When a user logs in the application, its login and password will be sent to the server in a HTTPS request, and a session which binds with the user identity will be created and stored temperately in the server. The server uses a session identifier to identify a session, and encrypts the session identifier with a randomized initial vector. The **laravel-session** value is the field in the cookie that stores this encrypted session identifier. The encrypted session cookie can prevent the attacker from creating fake identity by calculating the session ID.

As all the requests and responses are going on the SSL layer in the application, the attacker can not stole the cookies. Also, the session ID is a randomized string combining by nearly 280 randomized numbers and letters, making impossible for the attacker to guess the session identifier of another user.

5.3.4.5 Fake Identity

To identify a user's identity from a request, the application using the **Auth** component of Laravel framework will look up the session identifier the "laravel-session" field in the header of a request. If the session identifier is associated with a valid user identity, the server will retrieve the user identity from the database.

Comparing to the approach of presenting user name and password in every single request, which the client program does not need to record the user name and password permanently, this method provides higher security. Associating the session cookie with the user identity also has a merit of simplifying the codes in the server side, and improves the maintainability of the project.

5.3.5 Database Design

5.3.5.1 Entity Relationship

In the design of databse, we first identified three main entities in the database are *users*, *groups*, *movies*. The relationship between two entities such as the friendship, the message, group membership and the opinion to a movie can all be implemented by creating new tables, using the primary key of each entity as a foreign key. Other entities such as the movie file, movie image, and the secret of a group, can also be created as a table with foreign key constraint on the *movies* and *groups* table.

Another table *migration* is used to store the migration entries of the database schema. The schema migration is introduced in section.

The tables in the database with their foreign key constrains are shown as below.

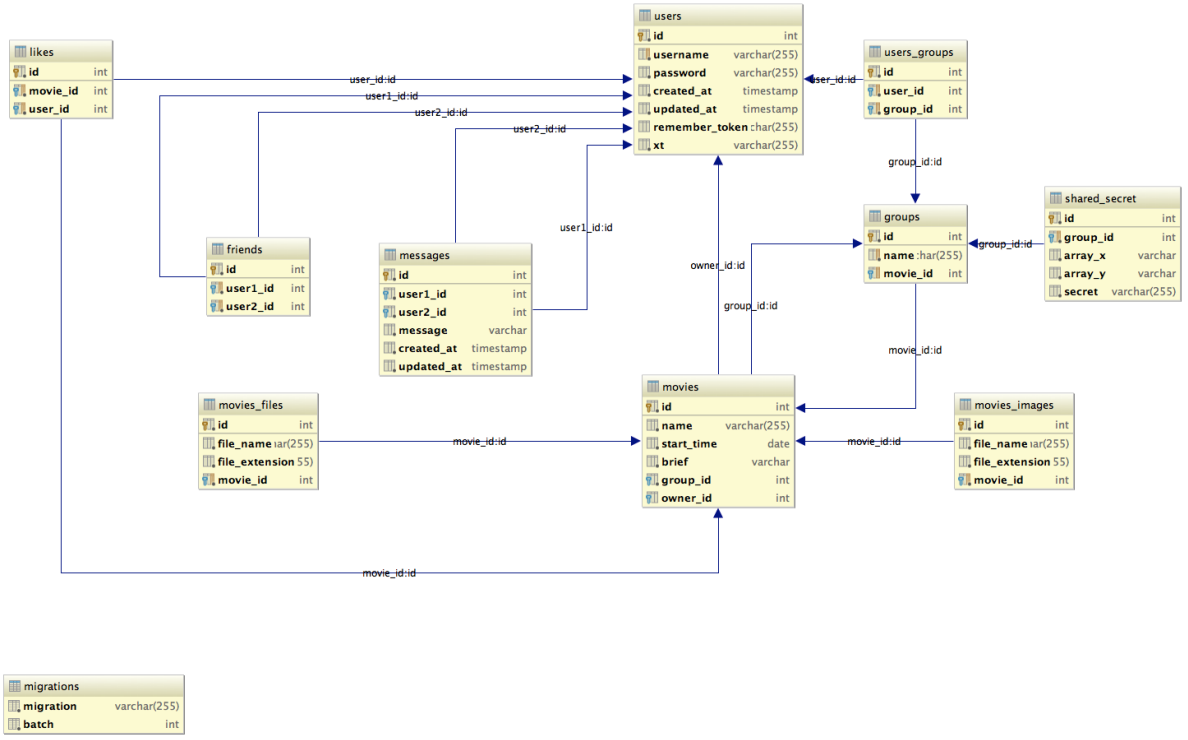


Figure 5.6: Database Diagram

5.3.5.2 Query Builder and Object-Relational Mapping

The **Fluent** query builder in Laravel is a interface for creating and running database queries. It is similar to traditional query, but with a more easy-to-use syntax.

Object-Relational Mapping (ORM), is a technique for automatically converting object-typed data in object-oriented programming to data types acceptable in database system. The in-built **Eloquent ORM**, as an extension of the query builder in the Laravel framework, enables further support of object modeling. A class extending Eloquent class represens a table in the database. Performing CRUD [5] operations on a table in database is made easy. For example, creating an instance of such class will result in adding a new tuple in the database.

The Fluent query builder has slightly better performance over Eloquent ORM for database query, but Eloquent is more capable of the relationship of the tables. A mixture of these two tools for building database were used in the project.

The project used Eloquent to specify the relationship between different entities. For example, to define the 1-to-1 relationship between the group and the movie, we can define a "has one" relationship by calling a *hasOne* function. We can then use this function to find the movie of the group. By describe the relationship in a way that is similar to natural language, the readability of the code is greatly improved.

Eloquent ORM also made the code cleaner. For some functions which involve updating on more than one tables, such as creating a group, requires adding new tuple on the table *groups* and *users_groups*. By using an *attach* function, the updating on both tables will be finished automatically.

```
Auth::user()->belongsToManyGroups()->attach($group);
```

[5] Create, Read, Update, Delete

5.4 Client Program Design

5.4.1 Model-View-Controller Class Design

A design pattern of **Model-View-Controller (MVC)** pattern is used for implementing application with user interfaces. The **model** represents the data storage in the application. The **view** is for rendering and presenting the user interface. The **controller** is the key role, linking the **model** and the **view**. When the user interacts the application by an action on a UI component, the controller will take decisions, such as making a change on the model, or making a networks request to the server.

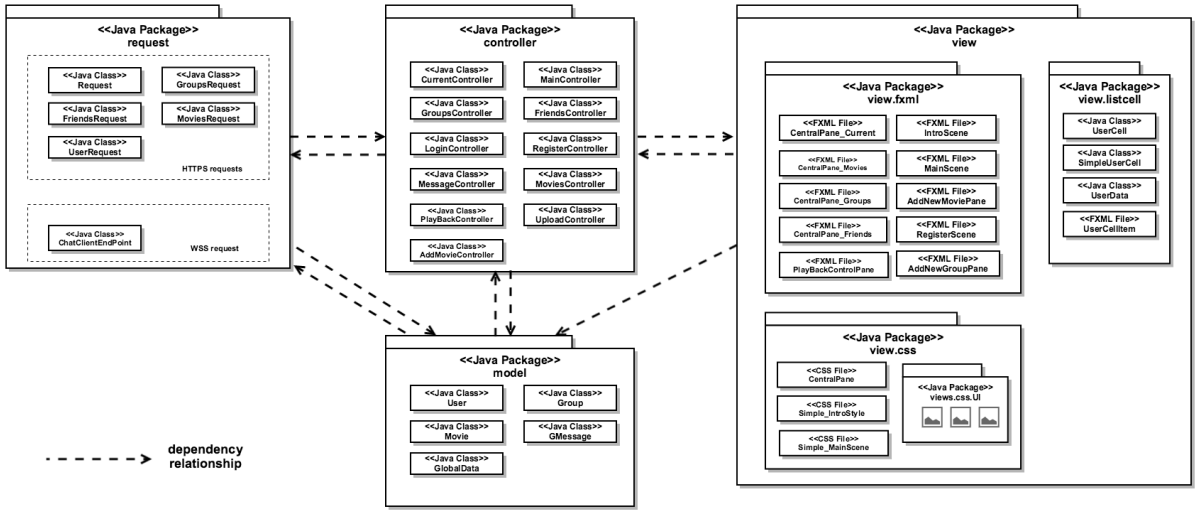


Figure 5.7: Package Diagram of the Client Program

In the package diagram of the client program, the **requests** package represents the request classes that are all public and static functions to be called by the classes in the **controllers** package. The FXML files in the **views** package are XML files used in JavaFX that define the layout of the components, and the styling of the components are done by CSS files in the **view.css** packages. The Controller classes in the **controllers** package will also use the FXML files upon initualization. The content of ListView UI component also comes from the Controller classes. The **models** package defines the data types of the User, Group, Message and GMessage (group message). The GlobalData class stores all the data that can be used throughout the program, such as the array of groups that the currently logged in user belongs to.

5.4.2 Relationship of Controllers

The controllers in the program decides both the background network task and the actions to be performed when receiving an user input. Thus it is important to understand the relationship between each controller and the view. The relationship between different controllers are shown as follow. The private attributes have been removed for simplification.

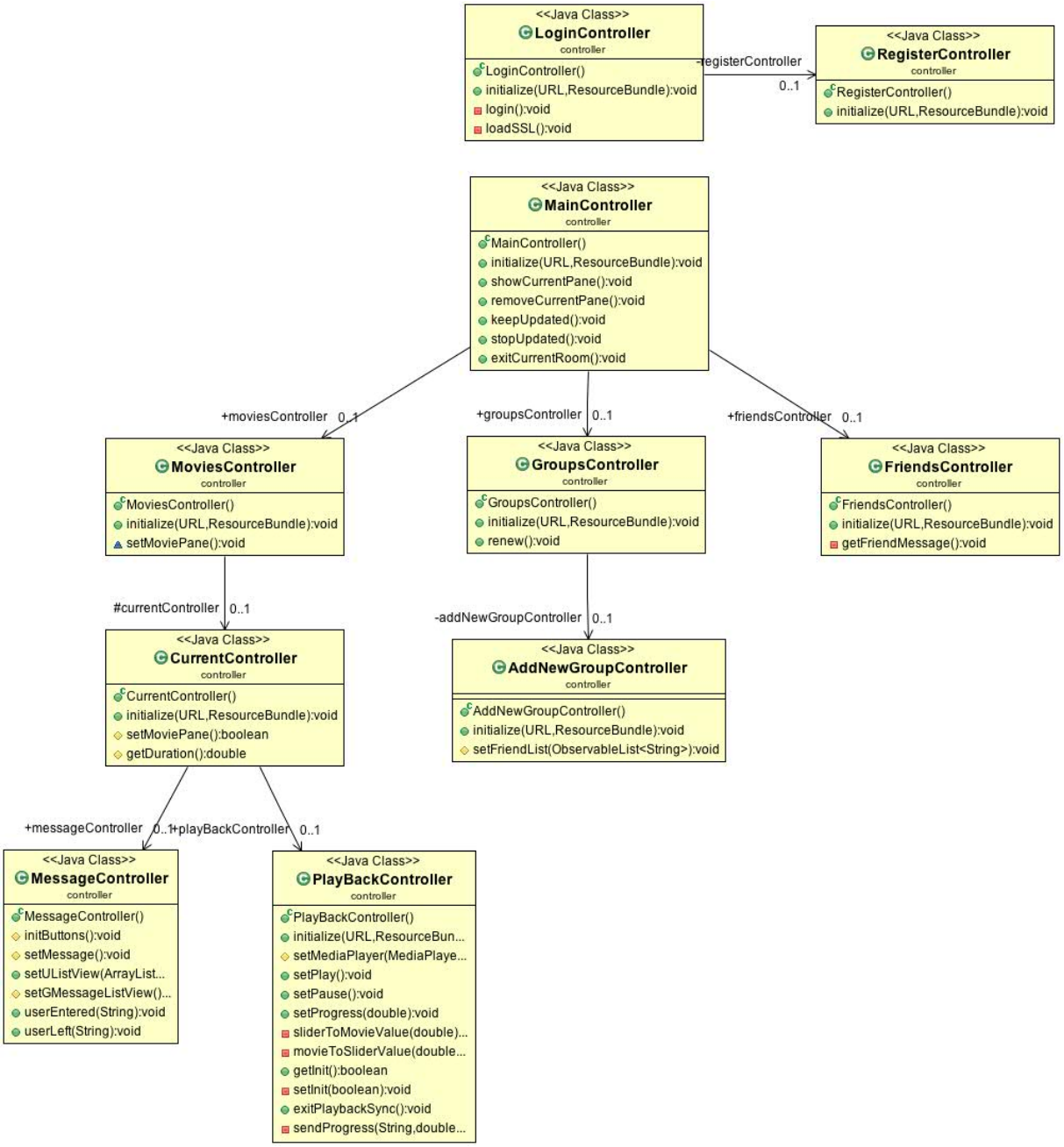


Figure 5.8: Simplified Class Diagram of Controller Classes (without Private Attributes)

The design of controller classes are according to the layered functional components of the application. The **initialize** function of each class registers all the events that can be triggered by the UI interface components.

On the top layer, **LoginController** and **RegisterController** take charge of the user login and register respectively.

The second layer is the **MainController**, which stores **MoviesController**, **GroupsController**, and **FriendsController** on the third layer. It is responsible for the transition between different panes. It also takes the periodical task of checking update of the server in the background.

The third layer, **MoviesController**, **GroupsController**, and **FriendsController** are responsible

for movies pane, groups pane and friends pane respectively.

The **CurrentController** on the fourth layer is used to show all the informaton of the current movie. The current movie is the movie that the user currently selected in the movie pane.

The **MessageController** and **PlaybackController** on the bottom layer are responsible for the live-chat and the movie synchronized playback.

5.4.3 Event-Driven Programming

When the user is successfully logged in, all the necessary information is returned in a HTTPS response, and it is stored in the client program. Then a scheduled repeating task, which sends HTTPS request to detect updates of information in the server, will run in the background. If there is an update, the client program will replace the updated version with the old version.

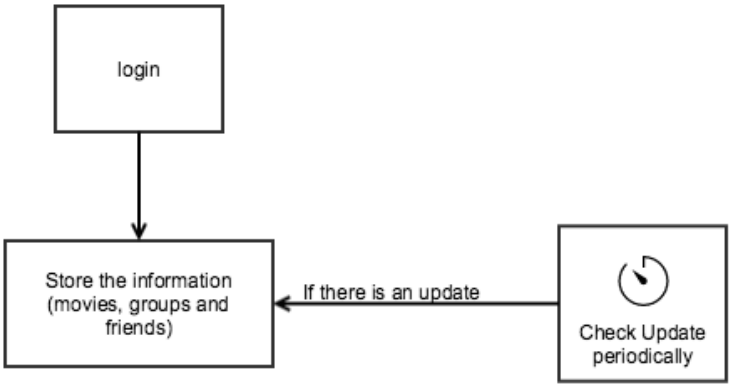


Figure 5.9: Scheduled Request to Check Update

The nature of a JavaFX desktop application is different from a browser-based application. The transition of different pageviews is not done by refreshing the or re-rendering a web page, but the transition between panes. A stack pane, which acts as a central pane manager, will control the visibility of each pane stored by itself. A user will not expect a desktop application to render its view upon request as he is waiting for the web page to be loaded. Therefore, in order to minimize the overloading of the transition of different panes, not every pane transition will triger a HTTPS request. Only request necessary to generate HTTPS request is made.

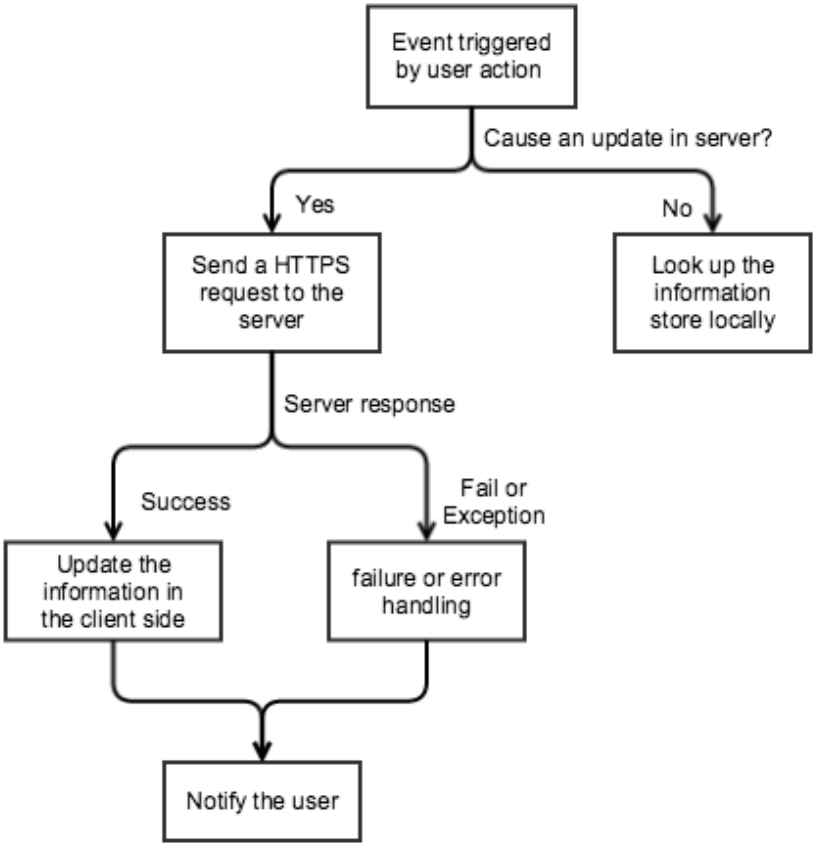


Figure 5.10: Decision for Events Triggered by the User

For an event triggered by the user, the program handles it according to whether it will cause an update in the server. Events that will not cause an update of the information in the server will only look up storage in the local program. Such events include presenting all the groups and the movie of the group. Only the events that will cause an update of the information in the server are necessary for update will send a HTTPS request. Such events include creating a new group, adding a user as a friend. The information in the client will be updated only when the server sends back a success response to the client.

If client's application is disconnected while the program is running, the an exception would be captured, and a dialog indicating connection failure would be prompted to the user.

5.5 Integration

The resulting design of the client and the server architecture is shown below. The white arrow shows the data flow related to the media content. This architecture is an integration of the real-time networks and request-response networks.

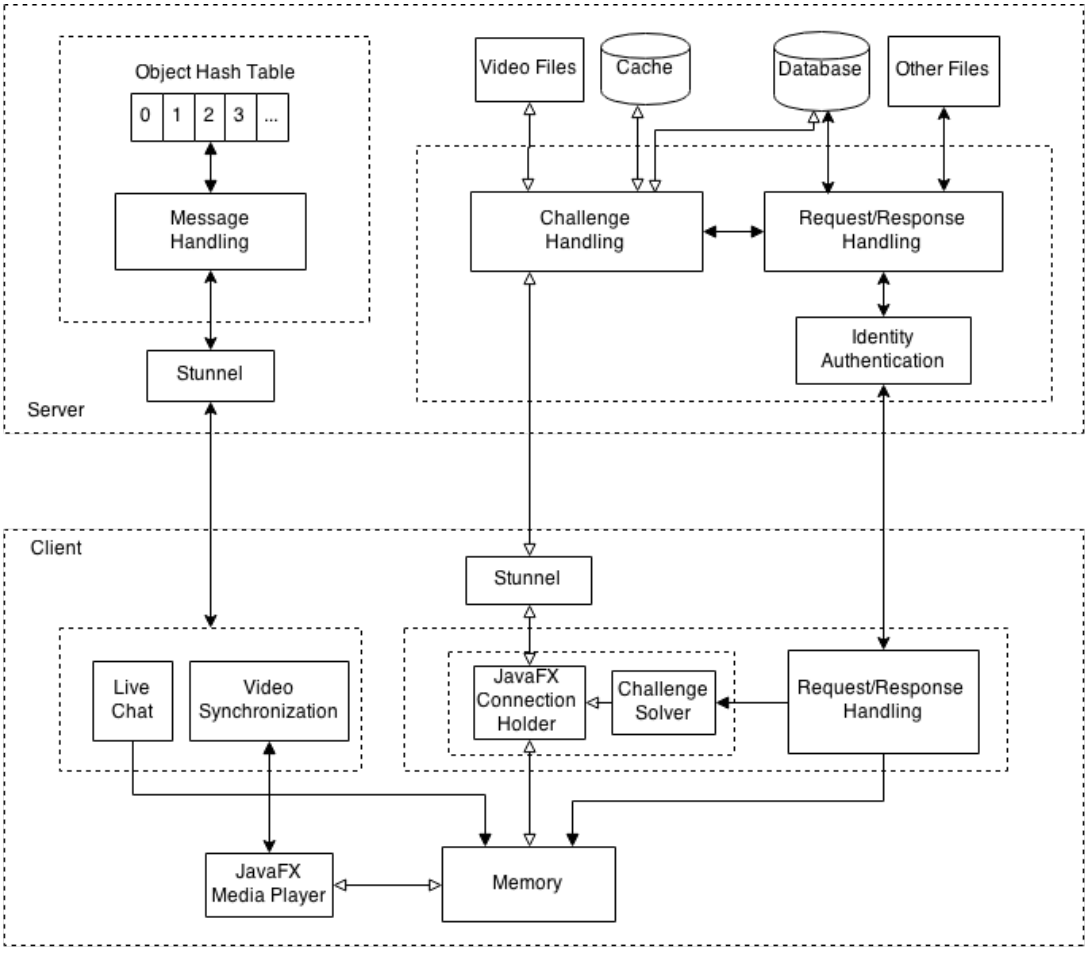


Figure 5.11: Client - Server Architecture Overview

Chapter 6

Software Implementation

6.1 Server Implementation

This section describes actions needed before the PHP scripts is running in the server.

6.1.1 Configuration of Server

The server for development is the localhost, and the server for deployment is `vm-yxf373.cs.bham.ac.uk`. It is a virtual server in School of Computer Science in University of Birmingham[1].

Three ports 8077, 8078, 8079 in the server are open to public for the incoming requests. The server in school is not open to public. To access it requires using the networks of Computer Science School, or using VPN. The port 8078 is open for HTTPS (HTTP with SSL) traffic, and the port 8079 is open for WSS (WebSocket with SSL) traffic. The port 8077 is used for HTTP requests and responses. This port is only for experiment use, and it will be removed for the official deployment. The reason for setting up this port is because the machines in the lab of school does not provide root access for students. We cannot use the Stunnel for SSL wrapping locally which is required to encrypt the traffic between JavaFX Media player and the server, as running Stunnel requires a root privilege. This port will be removed for deployment.

6.1.1.1 Resolving Dependencies

Library dependencies for Laravel PHP frameworks is also solved in the setting up phase. The required software such as composer, a dependency manager for PHP was installed. The version of PHP is required to be >= 5.4. PHP extensions mcrypt and mbstring should also be installed. The access of "app/storage" and "app/file" directory should also be changed to accessible to the Apache user group of the server machine, as the directory of files and session identifiers will be accessed by the web server.

6.1.1.2 Generation of Certificate and Key

To communicate over the SSL layer, the server needs to prove its trusted identity using a digital certificate. A digital certificate of the server is signed by a Certificate Authority (CA), which acts as a trusted third party.

The steps for generating the digital certificate is as follow. The method is applicable for the generation of both the server's and the client's certificates.

- Generate Certificate Signing Request (CSR) from Certificate Authority (CA). The CA for the certificate of the server side is the School of Computer Science in University of Birmingham. The CA for the certificate of the client side is itself.
- The server generates the key and the CSR file using openssl.

[1] This virtual server is accessible until June 2015.

- The server then convert the CSR file to a PEM file, which contains the certificate. The PEM file that represented the identity of the client program contained both the key and the CSR file.

The client side also needs to convert the PEM file into Java Key Store (JKS) file, which can be passed to the $SSLContextConfigurator$ in the system property that can be used throughout the whole life-cycle of the application.

6.1.1.3 Stunnel Configuration

The **Stunnel** program (see 5.2.4) listens on port 8079. A rule of listening incoming SSL packets on port 8079, unwraps the SSL-encrypted packet from port 8079 to a plain-text WebSocket packet, and forwards it to the chat application is added to the configuration file of the stunnel.

The client's key and the server's certifacte are also added to the configuration file of the stunnel in the server, in order to verify the trusted identity of the client, and also prove the server's own trusted identity to the client.

6.1.1.4 Schema Migration

The schema migration refers to the management of incremental, reversible changes to relational database schemas. Each migration represents a version of the schema of the database, so it is also a type of version control for the database. With the schema migration technology, the programmer can update or revert a database's schema to newer or older version. Meanwhile, the dependencies of fields and tables in the database is preserved. The schema migration makes the database development easier to control under the command-line interface to the virtual server.

In the project, the schema migration of the database is done by running a PHP migration script under the **artisan**, a command-line interface included with Laravel. Each migration action is represented by a class that extends the $Migration$ class. Each such class overrides the *up* method to specify the actions needed in current migration, such as creating a table and adding foreign keys. The class also overrides a *down* method which specifies the actions needed in a rollback of the current migration, such as deleting a table. The update and revert of each migration action are done by using **migrate** and **migrate:rollback** commands of artisan.The design of the database is introduced in 5.3.5.1.

6.1.1.5 Database Seeding

Database seeding is the initial seeding of a database with data, and it can be done with *Laravel* frameworks. **Faker**, a PHP library that generates fake data, is also used during the seeding for creating random user names, movie names, available dates, paragraphs etc.

Each seeding action is represented by a class that extends the *Seeder* class. Each such class impements a *run* function to specify the rules of seeding. The seeding actions are called by using **db:seed** command of artisan.

6.1.1.6 Version Control

Subversion (SVN) is used for the version control of the code for codes migration between the development server and the deployment server.The development was mainly done in local server. Each updated stable version on the code for server was first tested in the local serve before being committed to the SVN server. In the deployment server, a SVN **update** command was used. Testing on the deployment server for the stable version was performed. If the updated version of codes was not running under expectation, then a SVN **revert** command was used to rollback to previous version.

6.2 Client Application Implementation

6.2.1 Concurrency Handling

In Event-Driven Programming design, as the UI components are not thread-safe, the scheduling between different threads requires special attention. The client side program paid special attention to concurrency handling.

According to the Oracle official document [6], the update of the UI components should be executed in a separate JavaFX Application Thread.

```
Platform.runLater(new Runnable(){
   @Override
   public void run() {
        // update of UI components...
   }
});
```

For time-consuming background threads, such as networks request that would block the main thread, they should be executed in the call method of a one-time object of the Task class. The Task class can be used to complete tasks of system logic and it will not block the JavaFX Application Thread.

```
Task task = new Task<Void>() {
   @Override
   protected Void call() throws Exception {
        // time-consuming non-UI tasks ...
   }
};
task.run();
```

For the tasks that run in the background, but will also update the UI components, a Service class is preferred. For example, when the user choose to upload a video file to share among the group, a dialog with a progress bar indicating the uploading progress is shown to the user. The uploading request is done in a HTTP POST request, which is time-consuming. The skeleton of the function is shown below.

```
Service<Void> service = new Service<Void>() {
    @Override
    protected Task<Void> createTask() {
        return new Task<Void>() {
            @Override
            protected Void call() throws InterruptedException { // the user can interrupt the
                progress bar
                updateMessage("Uploading . . ."); // update the message in the dialog
                updateProgress(4, 10); // update the progress bar in the dialog
                // time-consuming non-UI tasks ..
            }
        };
    }
};
service.start();
```

6.2.2 User Interface

Adobe Photoshop was used for the initial design of the user interface. The image selected from online source was also edited in Photoshop to suit our program. Then **JavaFX Scene Builder** was used for

implementing the design. The components of each view can be drag-and-drop in the center panel of the **JavaFX Scene Builder**, and the style of each component can be specified by external JavaFX CSS style sheets. The effect can be viewed in the scene builder window in real-time.

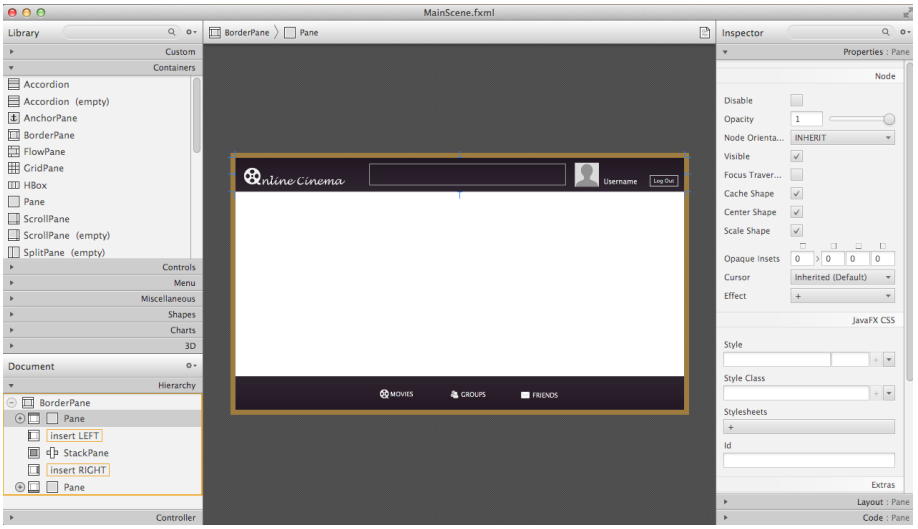


Figure 6.1: JavaFX SceneBuilder

The User Interface takes the target users into consideration. The target users of our software are mostly young (18 to 25 years old) females that would like to watch movies with remote friends or lovers. The theme chosen uses violet as the main color, because the darkness of the violet is similar to the atmosphere in the cinema, and violet color is more neutral than pink colors.

The usability of the software and an aesthetic principal of simpleness are both taken into account for the user interface design. To make navigation easy, the main menu buttons are placed at the bottom of the user interface, which the user can find at all time The semi-transparent design makes the user interface lighter than a filling without no transparency. The layout of components is also straight-forward to the user.

Figure 6.2: Screenshot of the Main Interface

6.2.2.1 Movie Screening Room

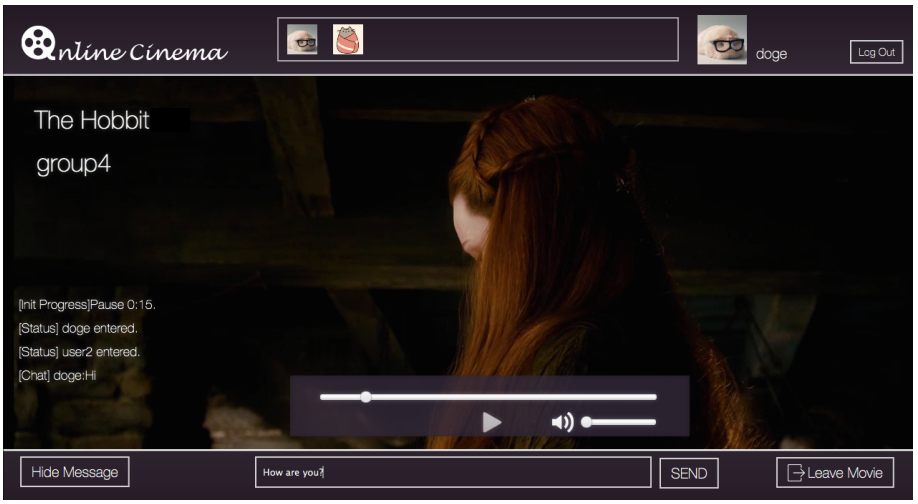


Figure 6.3: Screenshot of the Main Screening Room

There are three layers for the movie screening pane. At the bottom there is a background pane with a central stack pane. A stack pane is a container of UI components that acts like "stack", where a component newly added can cover the components below. The media view and the control pane can both be pushed to the central stack one after another. The filling of the control pane is set to transparent, but a listener of mouse entering is also registered on this control pane. It produces the hovering effect of the movie control pane.

In the original design, the list of group messages is to the left of the whole media view. The user feedback from the user acceptance suggested an improvement that the list of group messages should be moved within the media view, which can provide a larger space for the media view, and minimize the possibility of distraction.

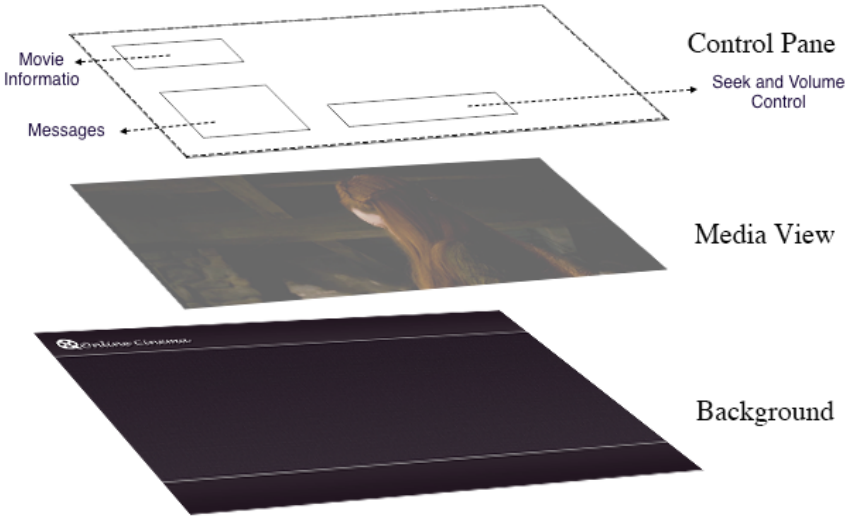


Figure 6.4: Layered Panes for the Movie Screening Room

Chapter 7

Verification and Validation

The process of verification and validation are used together to test whether the product has met the user needs.

The verification process was used to verify the quality of the software, and to what extend does it meet the non-functional requirements. A structural testing procedure is done throughout the project. The Unit testing, integration testing and regression testing are performed for both of the prototyping version and final version. Unit test is used for testing individual functions automatically. After the unit testing is approved, the integration testing that covers all the components will be performed to test sets of functions. If there is any issue being exposed during the integration test, a solution would be proposed for fixing the bugs. The regression testing is then neccessary as the changes performed after the integration test, as the actions produced by integration test may influence the working functions in the previous versions.

The validation was performed during development procedure mainly to see if the user needs are all satisfied. Sufficient user feedback from different background is also the key to the software development procedure. The user acceptance test and usability test are not only used for evaluate the project quality, but it also helps the development procedure. The design of the user interface changes as more user feedbacks are gathered thorught the user-centered development process.

7.1 Verification

7.1.1 Unit Testing

The JUnit Test Suite tool was used for unit testing of Java codes. The functions tested by the JUnit Test Suite are the public functions in **requests** package which issue HTTPS requests to the server and handle the requests from the server. Performing JUnit Testing on these functions would examin both of the PHP codes and the Java codes that implement the main system logic.

Additionally, unit testing on the functions involving user interaction was also performed.

The unit testing results can be found at the Appendix C and D.

7.1.2 Integration Testing

The integration test was performed after all the cases of JUnit test cases were approved in the prototyping phase and deployment phase. The integration test is done accross different functional components.

A bottom-down approach on the components of the system as shown in Figure **??** was used for the integration test in the project. The reason for choosing a bottom-down approach is because the sequence of the actions in the testing would be very similar to the reality.

We selected two integration testing cases for demonstration.

Test No.	1
Date	31/12/2014
Scope	"Live chat" and "video download".
Defect	The TCP socket and WebSocket socket are difficult to manage together because of different granularity.
Action	Change the video file downloaded from listening to a TCP socket with the video owner, to a RESTful request from the central server.

Test No.	2
Date	06/03/2014
Scope	"Friend management" and "group management".
Defect	When a user is added as a friend of the current user in the friend pane, the friend list in the group pane is not updated.
Action	Change the data type of friends list in global list from an array list to an observable list. All listview which set this observable list as item of the view will be notified upon change of the list, and reflect on the view to the user.

Table 7.1: Example Integration Test Cases

7.1.3 Regression Testing

The regression testing is performed after a major code change performed. The purpose is to prevent the code added in the later stage affects functionalities that were implemented earlier. In the project, the regression tests followed two situations: (a) the integration tests that produced actions on change on the system level; (b) the integration of the cryptography research on the project.

In situation (a), the integration test case No.1 demanded redesign of the networks structure, as the TCP socket connection was no longer used in the project. The classes related to TCP connection were removed, and the other components, such as user interface of notification indicating the progress of downloading was also removed, as the downloading would be performed without another user. The other components should be tested. In situation (b), after the cryptographic challenge was integrated to the software implementation, the other components related to the movie-viewing, such as the performance of instant messaging, synchronized playback, and the online users list should also be tested.

7.2 Validation

The validation of the project include usability testing and user acceptance test. User accpetance test is used to evaluate whether the software has met the needs of the users. The usability test which measure the usability from user's feedback is also necessary, as the usability is an important non-functional requirement of a software product.

In validation of the project, 20 randomly-selected students of studying different subjects (with 10 female students and 10 male students) were invited to test the software in the lab of the Computer Science Department in University of Birmingham after the verification has been finished. The participants were invited to use our application installed in the lab machines in the lab to watch movies with other participants. They were asked to fill in the questionnaire after using the software. The questionnaire can be seen in the Appendix [Appendix] for both of the user acceptance testing and usability testing.

The result will be discussed within the chapter of project result in 9.1.2.

Chapter 8

Project Management

8.1 Overview

The development consists of two part: the software development of the Online Cinema application, and the research part of digital rights management. The software development can be planned incrementally in the beginning, but the research part involved unforeseeable risks. Actions on the risks were required in such a time limit. Therefore, the development process of the project used prototyping and the spiral model. The time management of the project was monitored throughout the project. The weekly log is available in Canvas.

Week 1 to Week 3	Problem analysis and background research. The proposal was written in this phase with the planning of project.
Week 4 to Week 11	Fast software development for the prototyping without the constraint of some non-functional requirements.
Week 12 to Week 15	Conducting the research on digital rights management. The process involved a large amount of literature reviews. The scope of digital rights management of the project was also settled in this phase. The software application was also redesigned in order to meet the requirement of the digital rights management. Non-functional requirements were also considered in this stage.
Week 16 to Week 17	Integrating the research into the software. The result of research was integrated to the implementation of the software. Sufficient testing was done in this phase.
Week 17 to Week 18	Migrating server and upgrading the user interface. The server was migrated to the virtual server in Computer Science school. This involved close collaboration with the staff in the IT department of the school. The user interface in the client program was upgraded to a lighter and clearer version.
Week 19 to Week 21	Testing on the final version and deployment. The software was packaged and the verification and validation of the final version was performed.

8.2 Prototyping

The prototyping phase lasted from the week 5 to the week 11. In this phase, the main focus was to build a stable usable version. The local network environment was set for conducting experiments on the networks. A local server which ran simple PHP scripts and database query strings was set up in this phase. Several different types of media player frameworks, such as FMJ and VLCj based on Java were tried in this phase. The tool to build the user interface was also changed from WindowsBuilder on Java Swing frameworks to JavaFX frameworks. In the week 11, the movie synchronization was implemented based on UDP multi-casting, and the live-chat was realized by probing the server every two seconds. This phase only did not implemented the requirement on the system security and system scalability. In the end of the prototyping, the software were required to be evaluated, and redesigned if needed.

8.3 Risk Management

The whole project development used a spiral model, which is based on the incremental mode but puts more emphasizes on risk management. The whole project can be divided into several increments, and each increment composed of four phases: planning, risk analysis, engineering and evaluation.

The first increment is from week 1 to week 6. A very basic prototyping with functions of video playing and user login was produced with all the local networks environment ready. But after quick evaluation on the cost of development over Java Swing and the VLCJ media player, these tools were changed to JavaFX framework in the second increment (lasted from week 7 to week 11). In the end of the second increment, the video synchronization within a local networks environment and a long-latency chatting over the group was implemented. The Christmas break, which was the third increment, a new protocol WebSocket was used to replace the TCP and UDP connection, as it provided the features that can suit the needs of stable and event-driven multi-casting.

In the second term, the fourth increment, which involved the research on DRM as main tasks was started. The idea on the scheme was based on communication with project supervisor and professors in the Computer Science Department. The DRM analysis in this increment was no longer only focus on the cryptography, but also on the overview of the system. The reliability of the server system and the requirement on the accessibility from a global networks called for a change on the server architecture. The risk analysis on deployment on different PHP frameworks led to the choice of Laravel, a component-based PHP framework that is built on top of Symfony, but the development process of Laravel was much easier than Symfony. The web hosting was also migrated to the virtual server in the school with supports from the IT department of school. The last incremented started from three weeks before the demonstration. The suggestion on changing the user interface design which gained from the potential users' feedback, was turned into the upgrading on the user interface.

In conclusion, throughout the software development, the spiral model has provided a good risk controlling, as there always existed a workable version from last increment. However, it also meant more workload than one single version. But as the development process went deeper, the knowledge of the technologies and the expectation to the product was also improving. The overall performance was upgraded in each increment, and it is worthwhile to go through that process.

Chapter 9

Results and Evaluation

9.1 Result

9.1.1 Screenshots

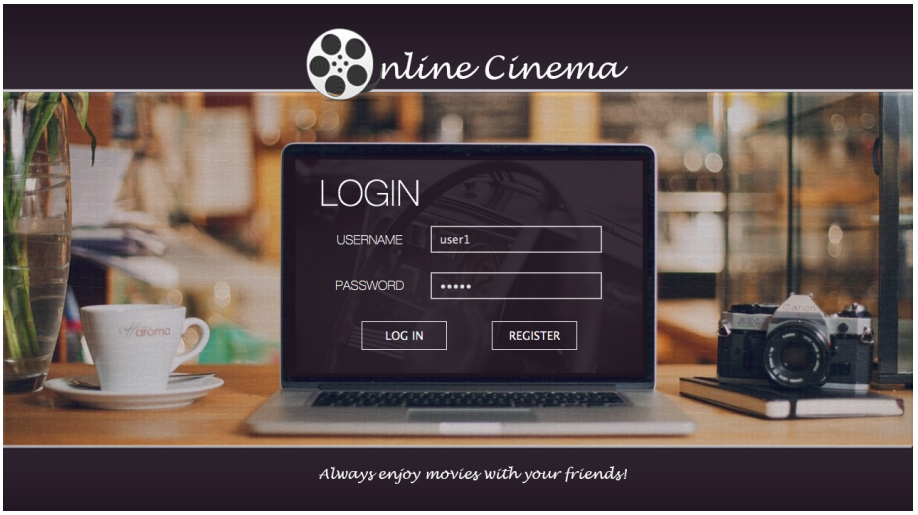


Figure 9.1: Screenshot of the Login Pane

Figure 9.2: Screenshot of the Group Management Pane

Figure 9.3: Screenshot of the Friend Management Pane

The interface design uses image from online source.[1]

9.1.2 User Feedback

20 randomly chosen students was asked to take part in our user acceptance test and usability test. The questionnaire could be found in the appendix. The questionnaire consisted of two parts: marketing survey and the user experience.

If you have watched movies online with your friends synchronously before, which method(s) did you use before? How was the experience? (Multiple choices)	Percentage
Used an online software, and it worked well.	0%
Used an online software, but it didn't work well.	10%
Synchronised the playback manually (eg. while chatting over Skype or telephone.)	20%
Shared the screen remotely.	20%

If you have not watched movies online with your friends synchronously before, what was the main reason?	Percentage
That is a great idea, but I didn't know how to do it.	30%
That is a great idea, but I haven't tried it.	20%
That is a boring idea.	10%

Table 9.1: Marketing Survey

[1]Source of images used to produce the user interface: `http://cdn01.wallconvert.com/_media/wp_400x250/1/2/17747.jpg`, `http://prodstatics3cdn1.purewow.com/images/articles/2014_11/noisy_coffee_shop_400.jpg`, `http://static.wixstatic.com/media/932404_e109c554d47c4e2b911801c4f89c59a4.jpg_srz_1349_646_85_22_0.50_1.20_0.00_jpg_srz`

In the marketing survey, the 12 out of 20 users had never tried watching movies synchronously with friends. However, 10 of these users expressed their interests in doing this. For those participants who had tried watching movies online with friends, they mainly used the screen-sharing or instant messaging tools for communication before sharing. There was not a well-functional automatic tool that can provide the user with a satisfying experience while watching videos synchronously with friends.

The participants were also asked to rate for their experience in using the Online Cinema application.

User Experience	1	2	3	4	5
Overall user experience	0%	0%	10%	10%	80%
Video synchronization functionalities	0%	0%	10%	20%	70%
User interface design	0%	0%	10%	25%	65%
Usability	0%	5%	15%	25%	55%
Likelihood to use in the future	0%	0%	20%	25%	55%

The score of "5" means very positive, and "1" means very negative.

Table 9.2: User Rating on the Software

The majority of the participants ranked the overall experience very positive. The functionalities and the user interface design they experienced were also very good.

A few users still found the software is not very easy to understand and use, and suggested adding more instruction on the software in the feedback, as they need to spend some time in understanding the requirement of adding friends before inviting people to the group. More instructions would be provided in the next version of the application.

9.2 Evaluation

Compared with the similar products which deal with movie synchronization, a large amount of products synchronize the videos on mainstream video-sharing websites such as YouTube and Vimeo. These products include **Sync Video** [8], and **Watch2Gether** [9]. It is difficult for normal users to use in case they want to watch a file from their own computer, because they will have to setup a web server or find one to store their file, and use the link to add the movie to be synchronized in the room.

If users only want to synchronize videos on the mainstream video-sharing websites, it is fine to use these kinds of software. But it would be more worthwhile to synchronize long videos, such as movies, series, or anime episodes, because they last longer and are the type of content people usually watch together. The case is that these mainstream websites also censor the content of the video files to be uploaded, leading the movie sharing impossible. Additionally, in some locations, YouTube is inaccessible because of the firewall policy of the countries.

Sync Play [7] allows the user to synchronize the video playback based on the media players (such as MPC-HC, VLC, mplayer2, mpv) which the user should already have installed in their machine. However, this software is not made for general user. The installation has to be built from the source code under the command-line terminal in Unix-like system. Even for the Windows operating system, the user has to install a visual studio 2008 package in case the installation process fails. This software is just for programmers or IT-curious people, not for normal users. It requires all the users to have the same video file in the disk before sharing.

All the video synchronization software do not provide support of file uploading and sharing. The Online Cinema application embraces the challenge to provide a better user experience. The adversary

model in the project deals with the layers above the operating system using a mechanism of cryptographic challenge. There are also some important security deployments in this project that prevents the system from certain attacks such as man-in-the-middle attacks, SQL injection and session hijacking. The only possible way to break the challenge without our software is to extract the JKS file containing client's secret key and server's certificate for SSL from the program, use them to establish SSL sessions with the user. The x_{k+1} is also required for client side the calculate the challenge returned by the challenge issuer of the server. Therefore, if a user breaks the challenge externally, it can be asserted that this user makes an effort in extracting the JKS file from the directory of the application, and get the x_{k+1} value fby analyzing the binary code of the application, or performing debugging on the program. This user can not claim that he "accidentally" broke the system.

All the four goals of the Digital Right Management proposed in the section 4.4 are web application security, networks security, logical security and minimal message exchange are implemented by the security deployment discussed in 5.3.4 and adopting the secret sharing scheme based on the works of Sharmir's, Berkovits's and McVittie's, which can be found in Chapter 4.

The functional requirements are tested in verification of the software using unit testing, integration testing and regression testing. The feedback of the participants in the user acceptance test has shown that the application have provided a stable performance, and great reliability on the video synchronization functionality. The usability testing of the software was generally positive. Accoding to the users' feedback, more instruction of usage would be added to the future version for improvement.

Distributions of application on Linux, Mac OS X and Windows platforms are also fully available. The requirement of the user knowledge about computer is nearly zero, as the user only needs to double click the executable file, making it very easy for downloading and installation.

The version control and the artisan tool used in the software automates the database migration and seeding, smoothing the process of server migration. In the case of an emergency, the server can be migrated to a new hosting server. Since the application can be routed to any server in a global network environment, during the period when the original server is under maintenance, all the requests and responses can be redirected to the newly available server. Both of the server and the client development used the Model-View-Controller and component-based design. The clear structure between each classes and the clear comments of the codes have improved the maintainability on the software.

Chapter 10

Discussion and Conclusion

10.1 Discussion

The outcome of the project is a piece of user-friendly software combining the video-synchronization and video-sharing. The reliability of the software is fully tested and by the users design of the software system was good and clear. The software development process was well-planned and incremental.

The issue of digital rights management that resides within a video-sharing software was tackled by conducting research on the digital rights management. This software ensured the security of the media content on the networks level, the web application level, the logical level with minimal information exchange. The software product satisfies the ethical and legal requirement of a video-sharing software by applying a broadcast encryption scheme that is resilient against the adversary model proposed in the project.

In the specification of the project, it was claimed that the project should use cryptography methods to ensure the user can not access the video file externally. During the project development, this goal has to make an adjustment to be "building a software that protects against user piracy on the application layer", as the current research results have shown the extreme difficulty of this problem. An adversary model was proposed for the software, and a solution that can work in application layer and networks layer. The model applied to purpose-built hardware, which the user can not give additional input to the system apart from what is constrained by the system, or analyze the internal logic of the system.

There are two security threats are not within the scopes of digital rights management in the project.

(i) **Read from Memory**

The adversary can read the memory location of the media content, and record the media content in that location, provided that it has the root privilege of the operating system. The adversary could also capture the system I/O or the content of the key when it becomes the root user of the system.

A non-root user can not read the virtual memory space that were used by the other users. However, the constraint of non-root users is not realistic in reality. Because even if an application is able to detect the current user of the operating system, as the adversary has full control on the system, he can disguise himself as a non-root user. Support from the operating system which can provide trusted information of the computer is needed.

(ii) **Reverse Engineering**

The adversary can also perform reverse engineering on the binary code. For example, the adversary could use tools such as Java Decompiler [3] to reverse the binary code of the program into the JAVA source code, analyze the algorithm to solve cryptographic challenge upon request.

Some existing proposals for reverse engineering. The first proposal is encrypting the binary codes of the application. However, the key to decrypt the binary codes must also be shipped with the program. If the operating system can find the key, the adversary can also extract it to decrypt the binaries. The second proposal, obfuscating the code such as using random strings for the names of attributes and changeing the sequence of execution, can only slow down the reverse engineering procedure. But provided with a large span of time, a skillful attacker can eventually construct the source code from the binary code. The other proposal of storing the binaries safely in the hardware, may be a promising approach.

The two threats above both required the trusted system environment in the client side. This naturally leads to consideration of specific hardware supports on the trusted system environment.

According to [18], the trusted platform module (TPM) is a computer chip that can securely store small pieces of data. A computer can show its current configuration to a remote machine the sequence of events executed from the boot time, with a hash value that can make sure the the log of events is genuine. TPM is also a secure cryptographic processor which can be used to perform calculation for simple ciphers. In this software, TPM may help to safely store the key required in establishing the SSL with the server and the value of x_{k+1} which is used in the cryptographic challenge. But it also make the software distribution extremely difficult. Every single variation of an operating system configuration that TPM can trust has to be saved in the server that attests the trusted configuration of computers. The dependency on TPM module will disallow Mac OS X users to use the application. It is also very difficult for the other system users to manipulate the TPM to use the application.

Flicker [15], as an extension for TPM, can make sure the application run securely under a compromised system. To run a piece of codes securely, the Flicker stores the artifacts of the operating system and suspend the whole operating system before the application starts running. After the application finishes the execution and cleans the memory usage, the operating system will resume to normal. Even though Flicker can prevent the configuration of the system being read during the piece of codes is running, Flicker still can not solve the two problems listed above, because when the system is suspended, the system I/O which is required by media rendering is also blocked.

It can be seen that the supports from currently available hardware such as TPM still can not solve these two problems wholly. Further investigation on hardware support for trusting computing is awaited for the future for digital rights management.

10.2 Conclusion

The project produces a high-quality software with an aesthetic user interface that satisfies the market needs in watching movies synchronously with friends. The idea of watching movie with a group of users using our piece of software has been proved to be very welcomed in the user acceptance test. It provides a useful service that at the same time prevents the users to whom the content is being shared from infringing copyright regulations.

The software has made a best effort in tackling the issue of digital rights management in a stand-alone software. The solution for digital rights management is by introducing a cryptographic challenge to the client, which is based on the broadcast encryption scheme introduced in [11]. It is very difficult for an external program to pass the challenge without deliberately extracting the key and forging a request exactly the same with the traffics go from our application. The security deployment on the web application also makes the software system resilient to many types of attacks listed in the project.

Further improvement on the project will explore the possibility to combine the security of the operating system levels into the project.

Chapter 11

Bibliography

[1] http://cdn01.wallconvert.com/_media/wp_400x250/1/2/17747.jpg.

[2] Burp suite. http://portswigger.net/burp. Accessed: 2015-03-31.

[3] Java decompiler project. http://jd.benow.ca/. Accessed: 2015-03-31.

[4] Laravel 4.2 official document. http://laravel.com/docs/4.2/. Online; Accessed 31-March-2015.

[5] Linux secret sharing library libgfshare. Accessed: 2015-03-31.

[6] Oracle javafx 2 document. http://docs.oracle.com/javafx/2/threads/jfxpub-threads.html. Online; Accessed 31-March-2015.

[7] Sync play. http://www.syncplay.pl/. Accessed: 2015-03-31.

[8] Sync video. http://www.sync-video.com/. Accessed: 2015-03-31.

[9] Watch2gether. http://www.watch2gether.com. Accessed: 2015-03-31.

[10] Alapan Arnab and Andrew Hutchison. Digital rights management-an overview of current challenges and solutions. In *Proceedings of Information Security South Africa (ISSA) Conference 2004*. Citeseer, 2004.

[11] Shimshon Berkovits. How to broadcast a secret. In *Advances in Cryptology—EUROCRYPT'91*, pages 535–541. Springer, 1991.

[12] Cory Doctorow. Microsoft research drm talk. *Transcript, Microsoft Research Group, Redmond, WA*, 17, 2004.

[13] Ian Fette and Alexey Melnikov. The websocket protocol. 2011.

[14] Amos Fiat and Moni Naor. Broadcast encryption. In *Advances in Cryptology—CRYPTO'93*, pages 480–491. Springer, 1994.

[15] Jonathan M McCune, Bryan J Parno, Adrian Perrig, Michael K Reiter, and Hiroshi Isozaki. Flicker: An execution infrastructure for tcb minimization. In *ACM SIGOPS Operating Systems Review*, volume 42, pages 315–328. ACM, 2008.

[16] LLC Pomelo. Analysis of netflix's security framework for 'watch instantly'service. pomelo. Technical report, LLC Tech Memo, 2009. http://pomelollc. files. wordpress. com/2009/04/pomelo-tech-report-netflix. pdf.

[17] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[18] Nancy Sumrall and Manny Novoa. Trusted computing group (tcg) and the tpm 1.2 specification. In *Intel Developer Forum*, volume 32, 2003.

[19] Neal R Wagner. The laws of cryptography with java code. *Available online at Neal Wagner's home page*, 2003.

Appendix A

Content of the Folder

A.1 Structure

1. dissertation
2. client - Java codes in the client side
3. server - PHP codes in the server side with Laravel Framework
4. release - Distribution of the software
5. Appendix B - Directory of the server
6. Appendix C - Unit Testing Cases
7. Appendix D - Ant Ouput of JUnit Testing
8. Appendix E - User Questionnaire

A.2 How to run the code

- Have Java SE 8 installed in the machine.
- If you are not using the networks within School of Computer Science, you need to connect to VPN of the school.
- If you are using Linux machine, go to the "Linux" directory under the release directory, and double-click the OCJavafx.
- If you are using Mac OS X, go to the "Mac OS X and Windows" directory under the release directory, and double-click the OCJavafx.

Appendix B

Directory of the Server

```
.
├── _ide_helper.php              // syntax highlight in PhpStorm
├── app                          // system logic folder
│   ├── chat                     // websocket application, created and implemented by the project
│   ├── chat-server-wserver.php  // script to run the application, created and implemented by the project
│   ├── config                   // configuration of the framework, changed by the project
│   ├── controllers              // controllers of MVC, implemented by the project
│   ├── database                 // database migration and seed, implemented by the project
│   ├── enc                      // share secret logic, implemented by the project
│   ├── files                    // directory to store files, created by the project
│   ├── filters.php              // application & route filters, not used in the project
│   ├── lang                     // different version of language, not used in the project
│   ├── models                   // models of MVC, implemented by the project
│   ├── routes.php               // routing table, implemented by the project
│   ├── start                    // class loader
│   ├── storage                  // storage directory, used by the framework
│   └── views                    // view, not used in the project
├── artisan                      // command-line tool of Laravel used during the development
├── bootstrap                    // used during the system is loaded
├── composer.json                // composer application for PHP extension management
├── composer.lock                // composer application for PHP extension management
├── public                       // public entry point, used by the project
├── server.php                   // tool for virtual server in the local development environment, used by the project
└── vendor                       // extensions used in the framework
```

Appendix C

Unit Testing Cases

Unit Test Cases

Date: 15/03/2015

No.	Module	Test Unit	Action(s)	Expected Output(s)	Result
1	User	register with valid input	Register with non-empty username, password, and image provided.	A dialog of success is shown, and go back to the main page	Pass
2	User	register with invalid input	Register with empty username.	An error message is shown in the register page	Pass
3	User	register with invalid photo	Register with empty image.	Use default system image as the user's image	Pass
4	User	login with valid input	Login with matched username and password.	Enter the main pane successfully	Pass
5	User	login with valid input	Login with non-matched username and password.	An error message label is shown in the register page	Pass
6	User	logout	In the main page, the "logout" button is pressed.	1. Back to the login pane 2. After logging in with another user, no side effect is produced	Pass
7	Friend	friends' photos download	Go to the friends pane.	The photo of all the friends are shown	Pass
8	Friend	search friends	In the friend pane, enter substring of friend's name in the search text field.	The names of the friend's with the substring are in the list view is shown	Pass
9	Friend	search non-friend users	In the friend pane, enter substring of a non-friend user's name in the search text field.	If no friend's name is matched, the names of non-friend user's with the substring are in the list view is shown	Pass
10	Friend	add a friend	In the friend pane, Select a non-friend user's name in the list.	1. A pane of adding friend is shown. 2. No meeesage pane is shown.	Pass
11	Friend	delete a friend	Select a friend's name.	1. No pane of adding friend is shown 2. A meeesage pane is shown	Pass
12	Friend	messaging a friend	In message pane, send a message to the friend.	The message is shown in the message pane of both of the current user and that friend.	Pass
13	Group	get information of groups when the user does not join a group	Log in as a user that has not joined a group. Go to the group pane.	A pane to create the group is shown.	Pass
14	Group	get the information of groups when the user has joined a group	Log in as a user that has joined a group. Go to the group pane.	A list of groups is shown	Pass
15	Group	create a group	Go to the group pane. Press the "create group". Fill in the group name. Select 2 users from the list of friends.	A group with selected users is created.	Pass
16	Group	select a group	In the group pane, press the "add group" button.	The first group in the list is shown.	Pass
17	Group	leave a group	In the group pane, select a group in the list of groups. Press the "leave group" button.	1. A notification of leaving group successfully is shown. 2. The group is removed from the group list.	Pass
18	Group	get information of group members	In the group pane, select a group in the groups list.	1. A list of members in the group is shown. 2. A list of friends that are not in the group is shown. 3. The button of "leave group" is shown.	Pass
19	Group	invite a friend to the group	In the group pane, select a group in the list of groups. Press the "invite to the group" button.	The friend became a member of the group, which can be seen by both of the current user and that friend.	Pass
20	Group Movie	get the movie event information of a group that has a movie event	In the movie pane, select a group that has a movie from the list of groups.	The information of the movie event of that group is shown.	Pass
21	Group Movie	get the movie event information of a group that has no movie event	In the movie pane, select a group that does not have a movie from the list of groups.	a pane for creating a shared movie event is shown	Pass
22	Group Movie	share a movie to the group with valid input	In the movie pane, select a group that does not have a movie from the list of groups. Attempt to upload a movie without a movie file.	A dialog to remind the user to upload the movie file is shown.	Pass
23	Group Movie	share a movie to the group	In the movie pane, select a group that does not have a movie from the list of groups. Attempt to upload a movie with all the information needed.	A dialog to remind the user to wait for the uploading process is shown.	Pass
24	Group Movie	watch a movie that is currently not available	In the movie pane, select a group with a movie which is currently not available. Press the "watch" button.	A dialog to remind the movie is not available is shown.	Pass
25	Current Movie	watch a movie that is currently available	In the movie pane, select a group with a movie which is currently available. Press the "watch" button.	A pane of the movie screening room is shown.	Pass
26	Current Movie	movie intitualization	Go the the pane of the movie screening room.	1. The movie is loaded into the media pane of the screening room. 2. The slider is set to the current position of the movie. 3. The list of online users shown on the top of the pane show correct information.	Pass
27	Current Movie	synchronized pause/play.	Login as 3 users in 3 machines respectively. Go the the pane of the movie screening room. Put the mouse on the media view. Press the pause/play.	All the 3 users would see the movie is paused/playing synchronizedly.	Pass
28	Current Movie	synchronized seek	After test #27, pressed the right position in the progress slider. Repeat the process a few times.	1. All the 3 users would see the movie is seked to the posistions that are specified by one of users synchronizedly. 2. All the 3 users would see the seeking message shown on the message pane.	Pass
29	Current Movie	live chat during the movie	After test #27, send a message to the users using the text input field shown at the bottom.	All the 3 users would get the message instantly	Pass
30	Current Movie	hide the information of the message	After test #27, press the "hide message" button.	The message of configuration and live chat becoms invisible, and the text on the button is changed to "show message"	Pass
31	Current Movie	show the information of the message	After test #27, press the "show message" button.	The message of configuration and live chat becomes visible, and the text on the button is changed to "hide message"	Pass
32	Current Movie	volume control	After test #27, control the volume using the media control pane	The volume of the media is changed accordingly	Pass
33	Current Movie	hide/show the media control pane	After test #27, move the mouse to enter/exit the media view.	The media control pane is visible while the mouse is in the area, and invisible while the mouse is not in the area	Pass
34	Current Movie	leave the group	After test #27, press the "leave room" button in one user's application.	1. The other 2 users can see the change on the list of online users. 2. The other 2 users can see the status change on the message list.	Pass

Appendix D

Ant Output of JUnit Testing

Unit Test Results.

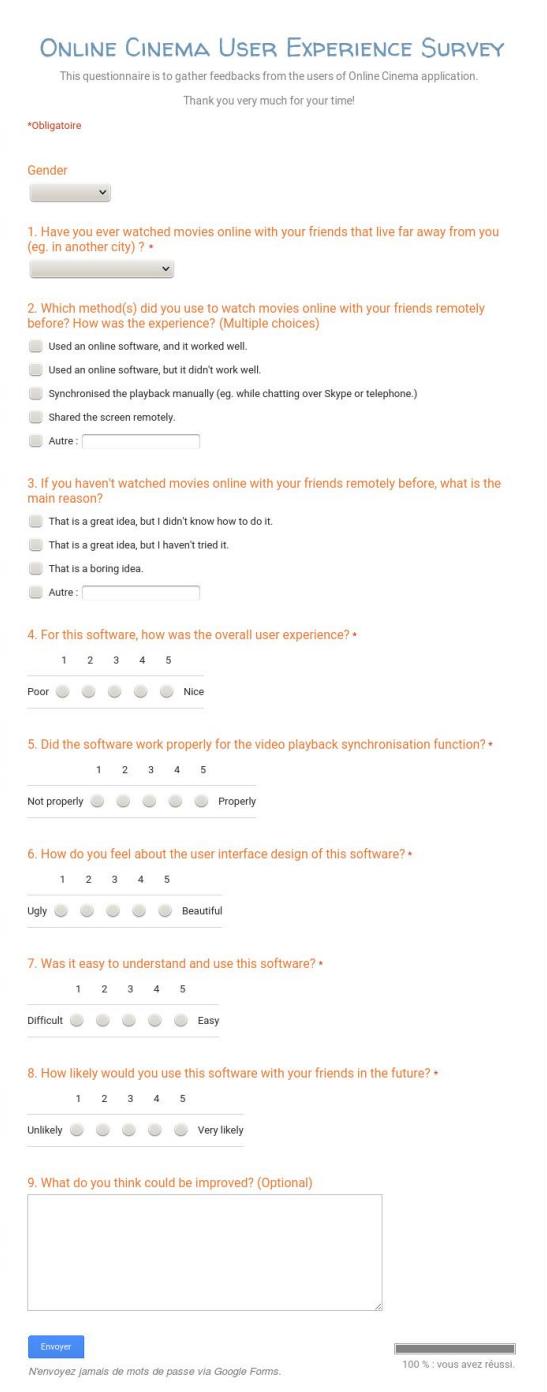
Designed for use with JUnit and Ant.

All Tests

Class	Name	Status	Type	Time(s)
FriendRequestTest	testAddAndDeleteFriend	Success		1.079
FriendRequestTest	testGetFriends	Success		0.232
FriendRequestTest	testSendMessage	Success		0.350
FriendRequestTest	testSearchFriend	Success		0.197
FriendRequestTest	testGetFriendMessage	Success		0.255
GroupRequestTest	testInviteToGroup	Success		1.170
GroupRequestTest	testCreateGroup	Success		0.332
GroupRequestTest	testLeaveGroup	Success		0.327
GroupRequestTest	testGetGroups	Success		0.264
MovieRequestTest	testDownloadMovieImages	Success		0.705
MovieRequestTest	testLikeMovie	Success		0.373
MovieRequestTest	testMovieGetLikes	Success		0.206
MovieRequestTest	testDownloadMovieChallege	Success		0.427
ShareSecretTest	testExtractSecret	Success		0.001
UserRequestTest	testUserRegister	Success		0.732
UserRequestTest	testUserLogin	Success		0.248
UserRequestTest	testLogOut	Success		0.225
UserRequestTest	testDownloadUserImages	Success		0.210

Appendix E

User Questionnaire



Online Cinema User Experience Survey

This questionnaire is to gather feedbacks from the users of Online Cinema application.

Thank you very much for your time!

*Obligatoire

Gender

1. Have you ever watched movies online with your friends that live far away from you (eg. in another city) ? *

2. Which method(s) did you use to watch movies online with your friends remotely before? How was the experience? (Multiple choices)

- Used an online software, and it worked well.
- Used an online software, but it didn't work well.
- Synchronised the playback manually (eg. while chatting over Skype or telephone.)
- Shared the screen remotely.
- Autre :

3. If you haven't watched movies online with your friends remotely before, what is the main reason?

- That is a great idea, but I didn't know how to do it.
- That is a great idea, but I haven't tried it.
- That is a boring idea.
- Autre :

4. For this software, how was the overall user experience? *

	1	2	3	4	5	
Poor						Nice

5. Did the software work properly for the video playback synchronisation function? *

	1	2	3	4	5	
Not properly						Properly

6. How do you feel about the user interface design of this software? *

	1	2	3	4	5	
Ugly						Beautiful

7. Was it easy to understand and use this software? *

	1	2	3	4	5	
Difficult						Easy

8. How likely would you use this software with your friends in the future? *

	1	2	3	4	5	
Unlikely						Very likely

9. What do you think could be improved? (Optional)

Envoyer

N'envoyez jamais de mots de passe via Google Forms.

100 % : vous avez réussi.