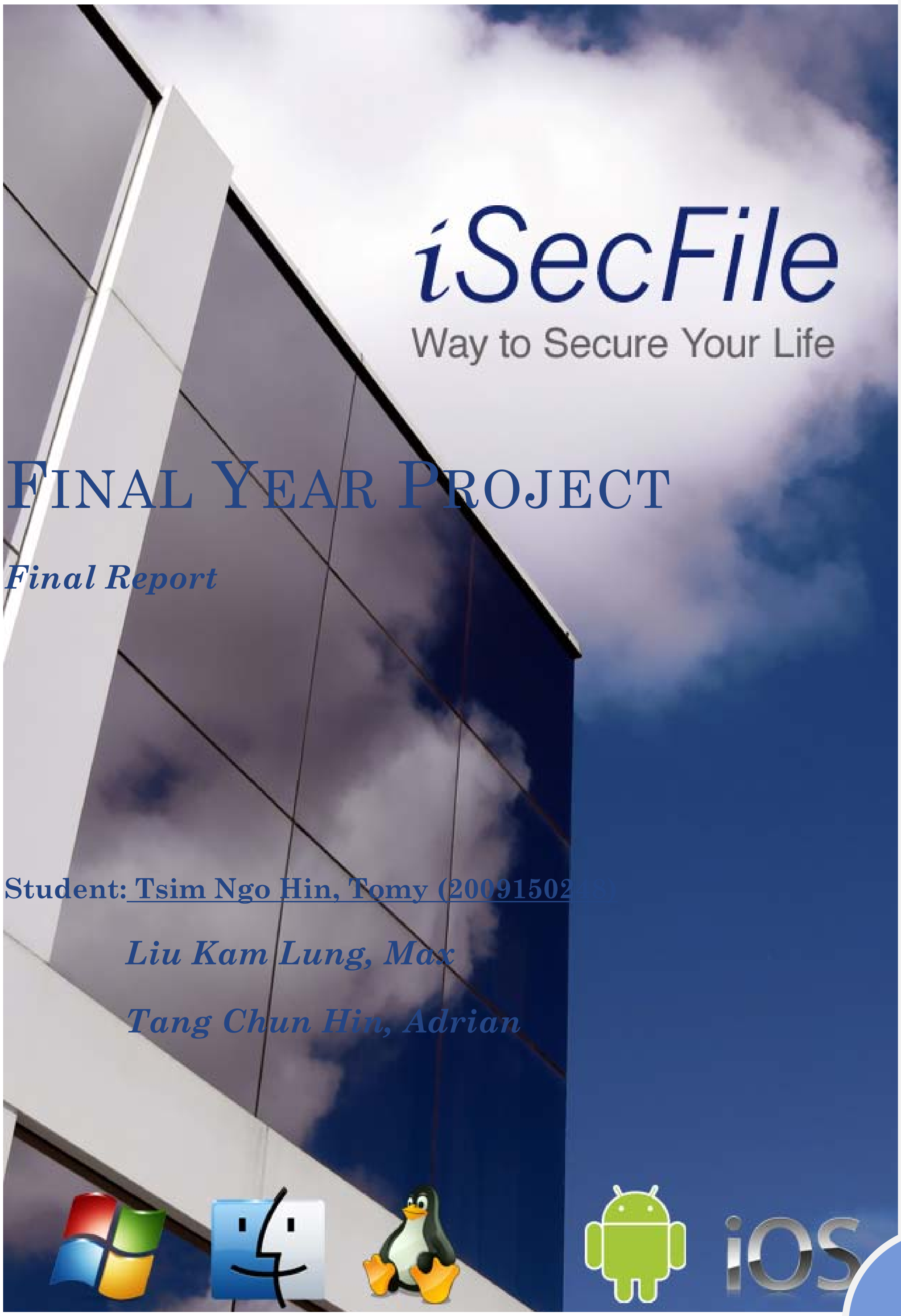


FINAL YEAR PROJECT

Final Report

Student: Tsim Ngo Hin, Tomy (20091502

Liu Kam Lung, Max

Tang Chun Hin, Adrian

Department of Computer Science

CSIS0801 – Final Year Project 2011-12

Supervisor: Dr. L.C.K. Hui

Dr. H.Y. Chung

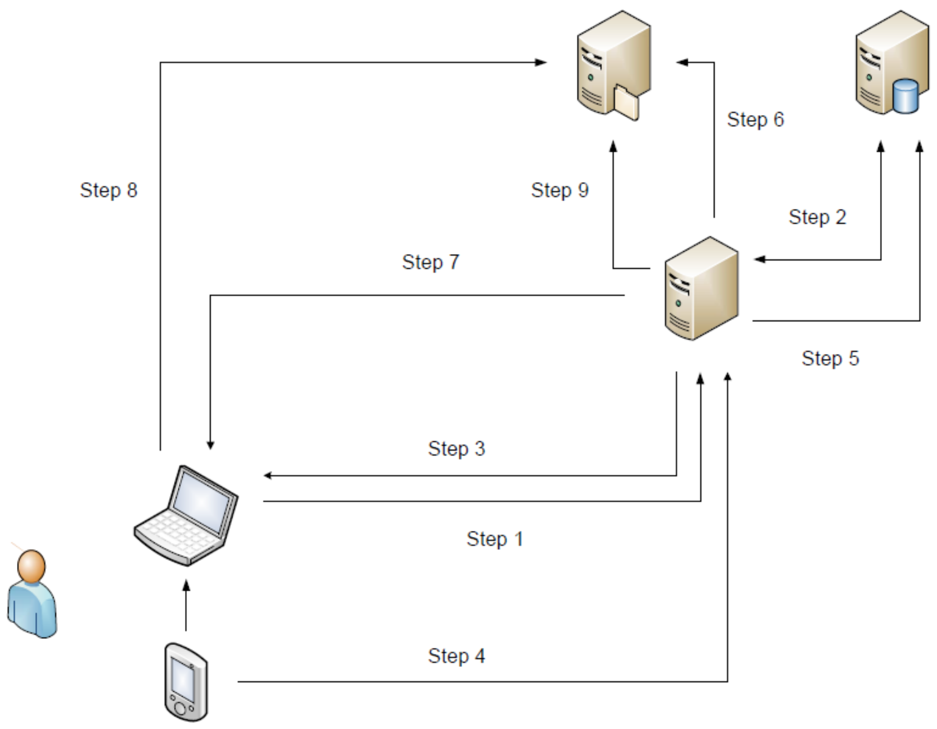
Abstract

With the increase of popularity of cloud service, many files have been stored in the cloud in order to get an easy access of the information anywhere. At the same time, security issues become a hot topic and an important concern among the users. Undoubtedly, a large variety of applications in the market can help users to (1) store their files on cloud system, and (2) access files saved conveniently. Almost most of them simply require username and password to be an authentication way when users are going to access their files. It didn’t give enough confidence to co-operate users to implement such an application or a system in their company.

Our project aims to build a user-friendly secure file system to strengthen the security or protection on the files stored on the server. The system is called iSecFile, an abbreviation of *I secure your file*. It has been developing in the stand of user with high security concerns. To balance between security and convenience, our system combines both our user-friendly mobile applications (in Android and iPhone) and our integrated system (File management system, Database management system, and Application server), users can easily access their data stored on the cloud by their laptop and mobile devices in a secure way. User-friendliness and security are the main aims of the project that we are developing.

Content Table

1. SYSTEM ARCHITECTURE



Step 1 • Authorized User Input username and password in the Client Application on the PC, and send the information to the server.

Step 2 • The Server validates the user by checking the User Database server and get response.

Step 3 • The Server returns a Random 128 characters long token and displays as QR Code on the Client Application.

Step 4 • User Scan the QR Code Token using the Authorized and Registered Mobile Device and send the token to the Server.

Step 5 • The Server confirms the valid token and record in the Database Server.

Step 6 • The Server generates a Random password and sends to the Samba file server as One-time password for mounting purpose.

Step 7 • The Server also sends the password to the Client Application to initiate the mounting process.

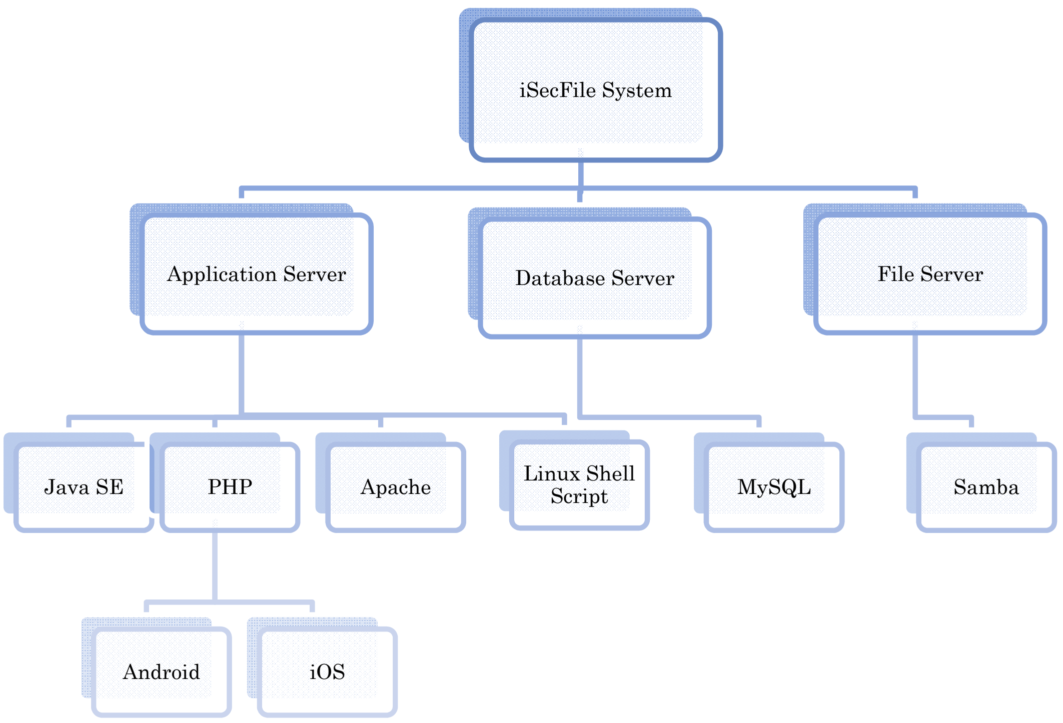
Step 8 • After Mounting Process Completed, Client application sends back a response to the server for confirmation.

Step 9 • The Server receives the response from client, and generates a new random password to Samba Server for guardians.

Step	Parties Involved	Detail
1	Client: Laptop Application server: Java application	Users issue access request to application server, i.e. Java application on server side, by client java application by their username and password.
2	Application server: Java application Database system: MySQL	To check if they are authorized, the application server will co-ordinate with database system to validate.
3	Client: Mobile phone Application server: Java application	A QR code will be automatically displayed on the application when they are authorized. One point needed to be noticed is that the server will ***hold the connection within 60s***. After 60s, the connection will be closed and users need to login again.
4	Client: Mobile phone Application server: Website	Users should use their registered mobile phone together with our application, either iPhone or Android version currently, to scan the QR code on the screen. After scanning, the application will automatically send a https request, with the mac address of their mobile phone and QR code to application server. This time the process will be proceed on website.
5	Application server: Website Database system: MySQL	When the request is authorized and is valid, the application server (website) will immediately update the corresponding status of the token to "detected" in database system.
6	Database system: MySQL Application server: Java application File management system, samba	At the same time, another component of application server (Java) will change the password of the users in samba once the status of the token changed to "detected"
7	Application server: Java application Client: Laptop	Concurrently, the application send the password generated to client java application. Client will notify application server step 8 complete successfully.
8	Client: Laptop File management system, samba	Client will mount their secure storage volume.
9	Application server: Java application File management system, samba	Application will change the password of the user in samba again after receiving a confirmation message from client java application

2. Construction Framework

Elements in the Project	Programming Languages and applications involved
Application Server	Java SE 1.6.0 PHP 5.3.5 Apache
Database Management System	MySQL 5.1.36 Samba v3.5.11 Linux Shell Script
File Management System	iOS 5 SDK Android 2.3.3 SDK



Platforms Supported		
Microsoft Windows	Mac OS X	Linux Ubuntu

3. Second Phase Major Workflow

On the 2nd phase of our project, we focus on the security issues raised from the first version of our system. Targeting the weakness point of the system, we are fixing the risk that might bring to the user and enhancing the user experience at the same time.

i. Token
In the first phase we have implement an extra token of 5 characters long. This is not secure enough as it is too short to be cracked by modern technology within a short period of time. We fix the risk by implementing a much longer random string token with 128 characters long, which is much more difficult to be attacked by brute force method.

ii. 60 Seconds Valid Login Time
After the user information is authenticated with the server, a random token is generated with 128 char long and only valid in a short period of time, for example, 60 seconds. Within the valid period, the server processes all response in real-time, dealing with all the connecting client and multi-thread functions. After the period, the token no longer acts as the key to login the server and labeled as abandoned token in our database.

iii. Token Input on Mobile Device
At the first phase of our system, user needs to input their username and password on the mobile devices to retrieve the valid token, and followed by inputting all three of the important information on the PC side. After the User Acceptance Test carried out in the mid-January, most of the comments received are concerning about the process of the token retrieval. It is quite inconvenient to input all the sensitive data on the mobile device with tiny keyboard, and troublesome to input all the data again on the PC side. As a result, we re-modified the whole system design. User only needs to input their username and password on the PC one time only, and the token input process is replaced by scanning of the QR Code. It makes use of the latest technology which QR Code can store a large amount of data and at the same time improving user experience.

4. FUTURE DEVELOPMENT

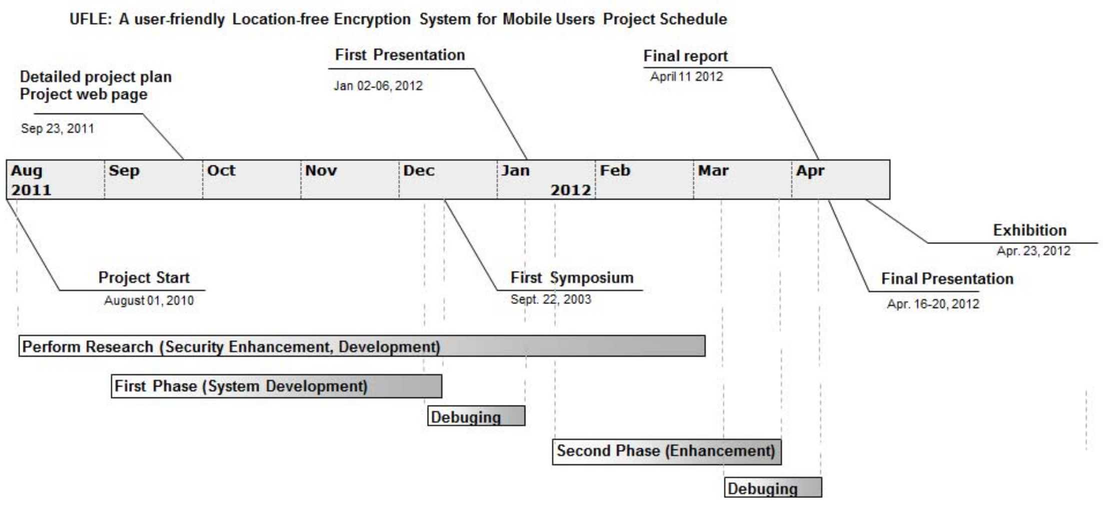
As the system is being developed within one year time, there are still a lot of improvement and functionality which can be implemented in the future.

i. Currently our system on considered and built on a stable network environment. We are planning to discuss the possibility of user the system in an unstable mobile network, and handle the interrupt cases due to the network congestion.

ii. In the current system design, only 1 mobile device is allowed for each user. Concerning the current state that user may own more than one mobile device like a phone and a tablet, we are planning to discuss the implementation of multiple mobile devices and review the security risk that might bring with.

iii. Sharing of files can be done with group folders if the user can in the same group. User can share files between different groups with the use of ACL management.

Whole Work Flow Diagram



5. Security Measures

- SSH, HTTPS ()
  - All the connection between the server and the mobile devices application are communicated with secure and encrypted SSH channel. The iSecFile.com server is also being verified with HTTPS connection.
- Samba
  - Samba username-password structure is modified with one-time password. All the Samba passwords will become invalid once mounting is completed.
  - Only the approved account such as phpadm is allowed to modify the user account details. Other users and parties are restricted.
- Android, iPhone
  - Static token is used to increase security.
  - Hardware token such as MAC address is used to identify the owner of the user.
- Website
  - Session: encrypted
  - Cookie: domain specific
  - Static token
- Client
  - The client will close all the connected drive once the client program is closed.
- Unique Identifier

  Before selecting using MAC address as unique identifier, we have brainstormed and researched a lot of different solutions, such as IMEI code, ANDROID_ID, Serial number, MAC address and Bluetooth ID. We consider among convenience, security, and compatibility problem before the final decision.

  <u>(i) IMEI code</u>
  It is not the prime option because we think that user would become suspicious as permission is needed. Besides, IMEI have only in the phones, not tablets. From IOS perspective, iOS 5 does not allow us to get it.

  <u>(ii) ANDROID_ID</u>
  It is changeable at the factory reset and unpredictable changeable on rooted phones. Besides,
  iPhone does not have this ID number.

  <u>(iii) Serial Number</u>
  From technical perspective, it is easy to get using a function called getDeviceID(). However, it will return the MDN or MEID of the device depending on which radio the phone uses (GSM or CDMA). More importantly, it doesn't work for tablets.

(iv) MAC address
Based on our research, 95% of Android Powered Devices have a wireless card, which is unique among devices theoretically. Though it can be faked easily in "rooted" mobile phone, we assume that users will not display their mobile phone MAC address to people easily.

Limitation

There are a few limitations that our system can work on specific environment:

- Only 1 MAC address per user is allowed
- Only Stable network environment is considered.
- Root permission is needed for mounting drive in Linux
- Several pre-settings and configuration are needed in Windows for proper working. (See Installation Guide).
- In all the platforms, the user account login which login to the computer without administrator right might not be able to mount all the drive space successfully.

6. Responsibilities

Distribution of work can be summarized by the following table

	Server	Client
Samba	Tomy Tsim	
Website	Max Liu	
Java	Max Liu	
		Tomy Tsim, Adrian Tang, Max Liu
Hosting	Adrian Tang , Tomy Tsim	
iPhone		Adrian Tang
Android		Tomy Tsim, Max Liu
Graphic design		Adrian Tang

7. Development Tools

(A) Eclipse [3.6.2]

Eclipse is an open source community about software development. It is a very popular tool in the world. Also, there are many resources and tips on the internet so that it can enhance our work efficiency.

(B) XAMPP [1.7]

XAMPP is a multiplatform web development environment. It allows us to create web application with Apache2, PHP and a MySQL database easily. To create a dynamic web system, we use MySQL, phpMyAdmin containing in XAMPP to build and maintain the database, test the functionalities.

(C) NotePad++

NotePad++ is another useful tool for us to deploy PHP codes

(D) Android SDK [2.3.3]

To develop Android application, Android SDK is the essential tool. This SDK is very comprehensive tool that contains not only the library for development, but also includes the simulator to test the application. We use eclipse together to run and develop all the Android application.

(E) Xcode [4.3.2]

Xcode is another essential tool to develop iOS application. The Xcode interface seamlessly integrates code editing, UI design with Interface Builder, testing, and debugging which help us to develop iOS application in an efficient way.

8. QR Code

8.1 Brief Description

As a string is used as a token to access the file server, the vulnerability of the token becomes another measure for the safety of the whole system. According to research, a password with length of four digits, like the PIN code implement on the iPhone, can be discovered within several seconds only by modern technology. With the use of dictionary attack, the time needed can be even shorter. As a general guideline for password nowadays, a least length of eight characters, with combination of uppercase and lowercase letter, digits and symbols are considered as a standard. Longer length of the password without the use of dictionary word can result a better security. However, these measurement causes inconvenient to users as they are difficult to be memorized. It always leads to exposure of the password as it is written down in paper form which is being lost easily. Also, it is quite difficult to input a long password with symbols on mobile devices.

In choosing between security and user experience, a compromise can be made between two of them by making use of QR Code. It is 2D barcode which can store a lengthy string into a square bitmap. It is encoded with special algorithm which cannot be read by human. By using the QR Code Scanner, the bitmap can then be decoded back to a normal string.

It is suitable to replace the token we mentioned above as it has several characters:

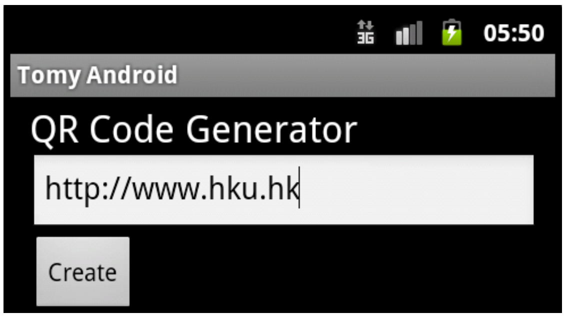
- The QR Code can be generated in run-time, meaning we can generate different QR Code each time.

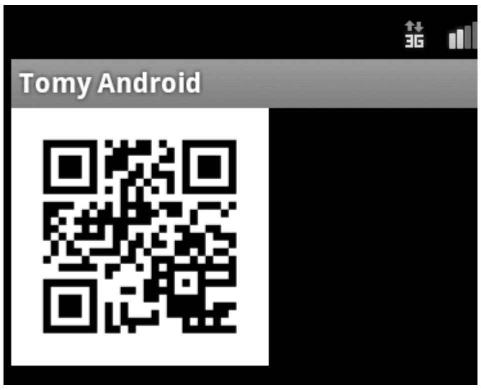
- It supports English characters as well as digits and symbols.
- It can support up to 174 char in version 10, and more in later version.
- It also contains a Read-Solomon Error Correction, which can restore up to 30% error correction depends on different Levels:
  - Level L 7% of codewords can be restored.
  - Level M 15% of codewords can be restored.
  - Level Q 25% of codewords can be restored.
  - Level H 30% of codewords can be restored.
- As it is human unreadable, the token string cannot be easily peeked by the others.
- It is convenient to input the lengthy token by scanning the QR Code than input a long string of token by hand on the mobile devices.

8.2 Implementation

As first of our idea, the token is retrieved on the mobile devices. We have to generate a QR Code on the smartphone, and being scanned by the PC in order to replace the hand-input token.

As QR Code is an opening source project, there is an open source library "ZXing" on the Internet which we can make use of. We still have to implement some code and call some API functions of the library.





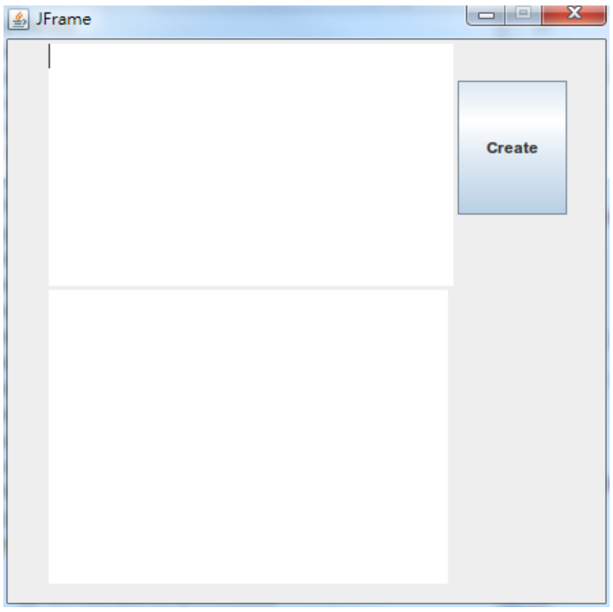
The QR Code can be generated according to the string we input. This is the testing app that we can implement the QR Code generator on the Android App.

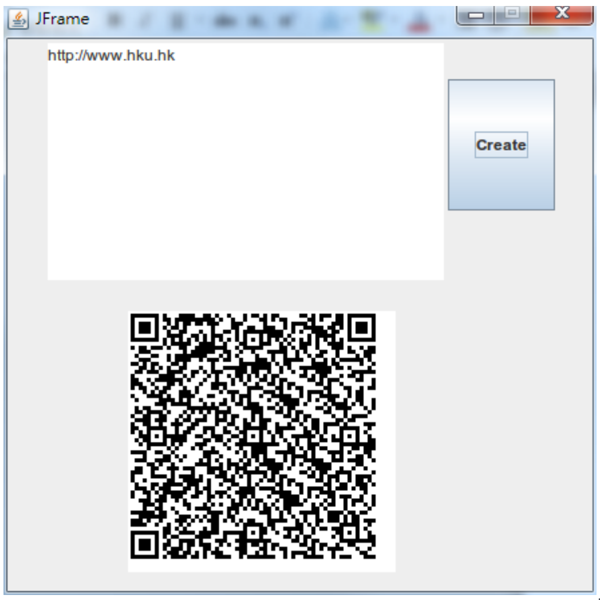
However, after our discuss, it is uneasy for the Laptop to scan the QR Code on a mobile phone, and it is difficult to embed the related Camera driver of different computer to initiate the video capture function. So we would like to reverse the idea.

8.2.1 PC Platform

On the PC Platform, after login using our client Java Program, it will display a lengthy token string, which will be scanned by the registered mobile devices for validation. This process is reversed compared to the idea before, but it is more user-friendly and easy to use.

Instead of generating the QR Code on the smartphone, the Code will be generated on the PC side. A testing program is made to make sure all the bitmap functions work correctly.





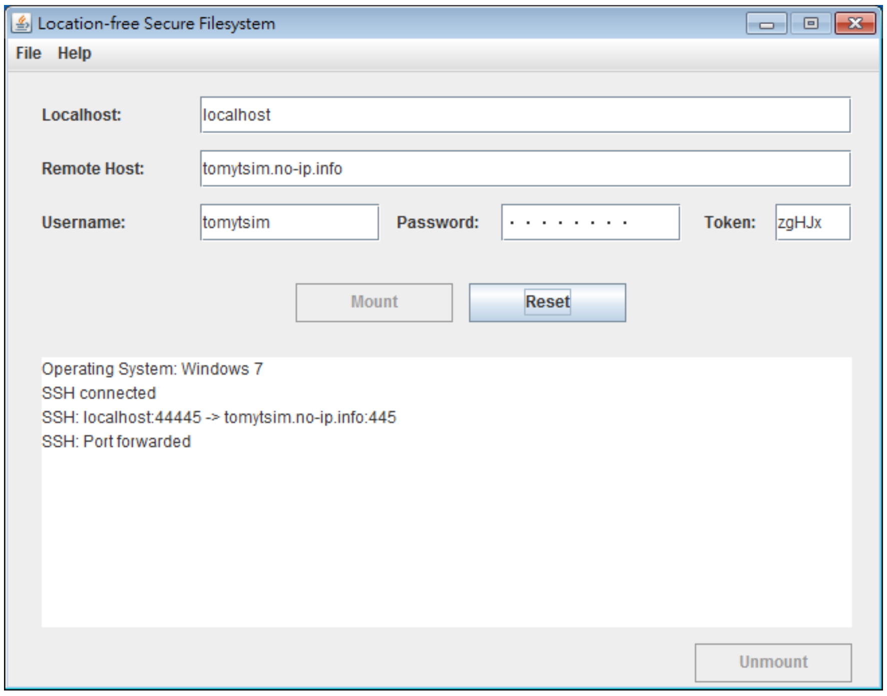
http://www.hku.hk

In the second phase, we can port the QR Code Generator into the client mounting program, though it is not as easy as we thought.

However, some thread handling needed to be taken care as the GUI display thread and the system command thread are required to be run on separated thread in order to work correctly. SwingWorker Class thus is used for display the Code and the status in real-time. The QR Code we generated also has a Level M error correction, meaning 15% of the code can be restored.

```
public class Authentication extends SwingWorker<Strign,String>{
    ……
    @Override
    Protected String doInBackgroud() throws Exception{
            // do connection and UI updates
    }
}
```

Version before QR Code Implementation



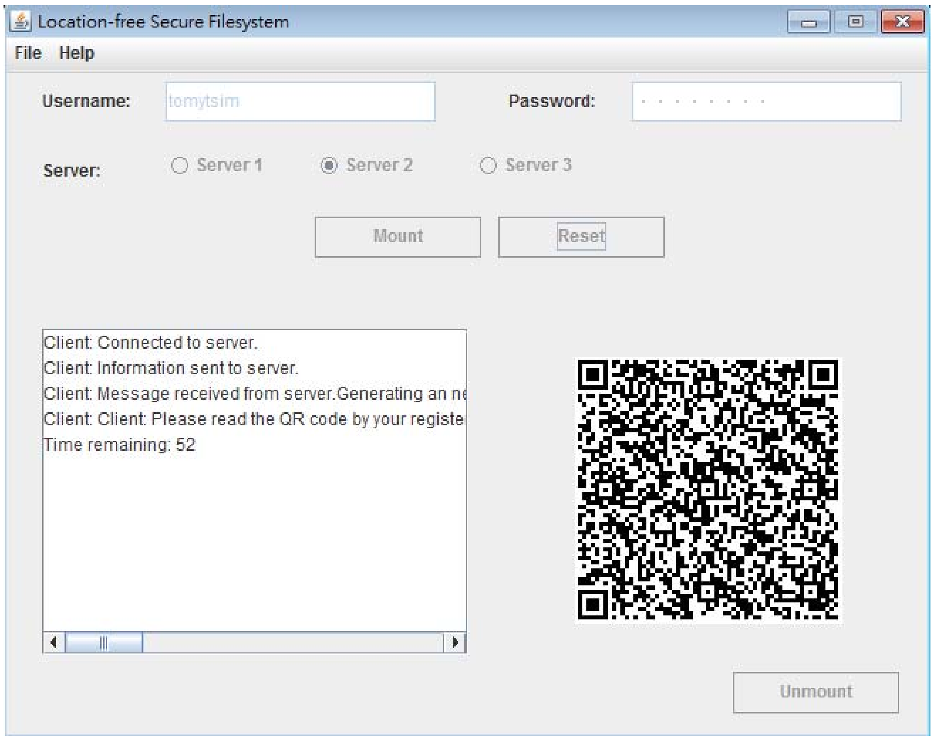
Localhost: localhost

Remote Host: tomytsim.no-ip.info

Username: tomytsim Password: ········ Token: zgHJx

```
Operating System: Windows 7
SSH connected
SSH: localhost:44445 -> tomytsim.no-ip.info:445
SSH: Port forwarded
```

Version after QR Code Implementation

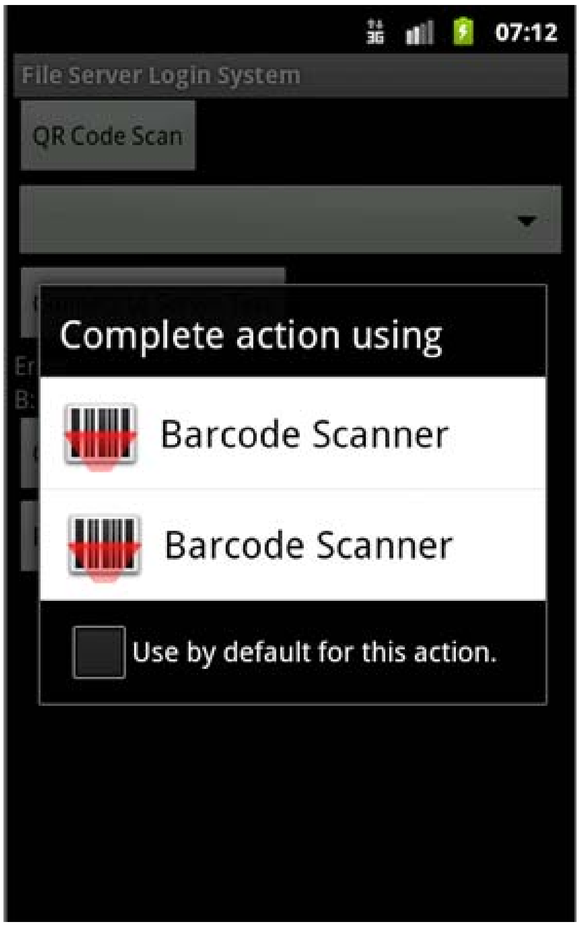


```
Client: Connected to server.
Client: Information sent to server.
Client: Message received from server.Generating an ne
Client: Client: Please read the QR code by your registe
Time remaining: 52
```

8.2.2 Android Platform

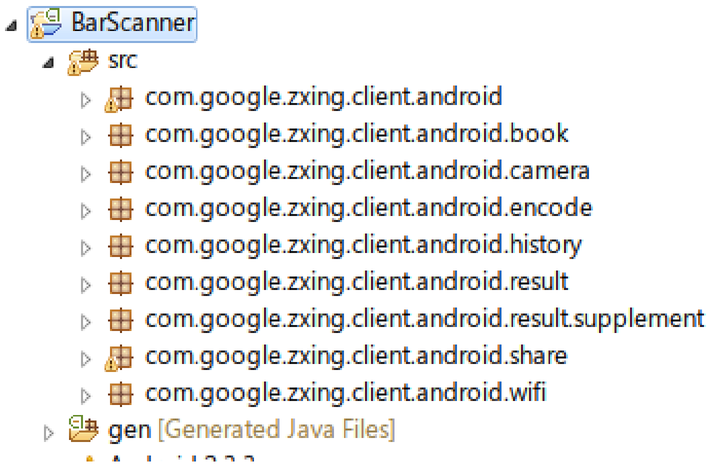
On the Android Phone, we have to implement a scanner instead of a generator to scan the code.

According to the Zxing library and his suggestion, it is convenient and easy to use a function to initiate the scanner by calling "com.google.zxing.client.android". This need to be done by installing a third party application which has the QR Code Scanning function like Barcode Scanner made by Zxing itself. This is for easy calling and error handling. If the users haven't installed the scanner, a dialog will be prompt and take the user to the Android market to install the QR Code Scanner.



After our discussion, although it is easy to implement, it needs user to install another application on their mobile devices. So we decide to embed the whole scanner into our application.

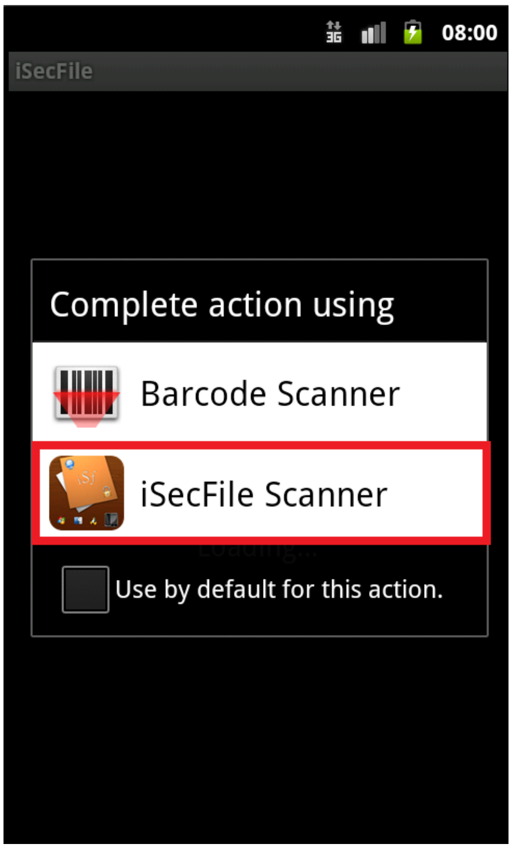
In order to do this, we include the whole related package of the QR Code scanner from the library.



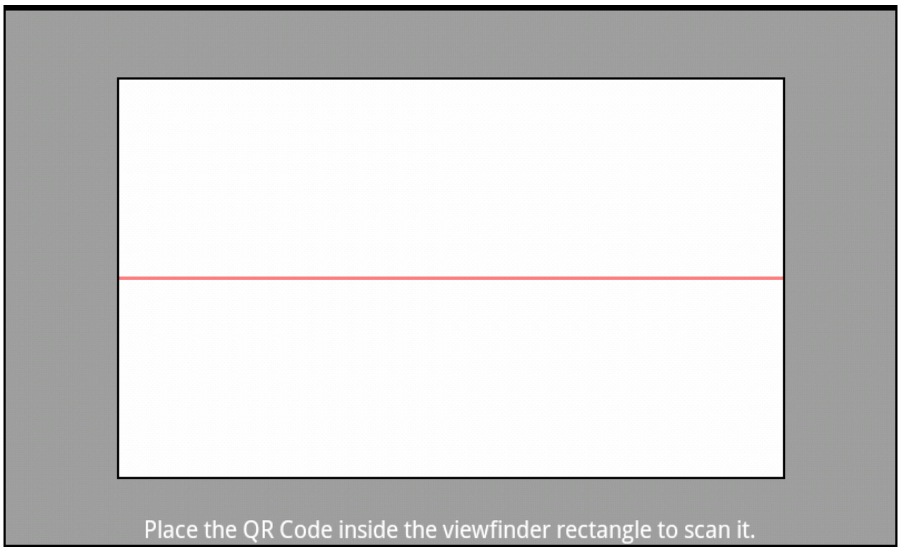
```
BarScanner
  src
    com.google.zxing.client.android
    com.google.zxing.client.android.book
    com.google.zxing.client.android.camera
    com.google.zxing.client.android.encode
    com.google.zxing.client.android.history
    com.google.zxing.client.android.result
    com.google.zxing.client.android.result.supplement
    com.google.zxing.client.android.share
    com.google.zxing.client.android.wifi
  gen [Generated Java Files]
```

And we integrated, modified and optimize the codes to suite our needs.

Now user can use the application without installing 3rd party app. If users still want to use other scanner for better performance, they can still being prompted to choose between different apps. All the scanned result will be returned back to our application for process.



What user has to do is to put the scanner on top of the QR Code displayed on the Client Java Program. It also implemented auto focus function. Depends on the Devices processor ability, the Scanner can quickly capture the code within seconds and decode the bitmap back to normal string. We also optimized it so that it can only scan QR Code, but reject all the other coding such as 1D Barcode, ISBN code, and websites code in error handling and prevent malicious website attack.


Place the QR Code inside the viewfinder rectangle to scan it.

9. Android Application

9.1 Brief Description

Other than Username and password, we need an extra token to secure the account. In most of the cases happened in the past, users would like to write down all the passwords in one note. In case they lose the notes, the account will be exposed no matter how many passwords they have.

In balance between convenient and security, we try to make use of users' personal smartphones like Android and iPhone. First, User login to their own computer, then scan the token generated by the registered mobile devices for validation. Secure File Volume would then be created and mounted for personal workspace.

Different types of Security measures are taken here:

- Unique MAC Address

  As MAC Address is unique among all the Internet accessible devices, it is used as a unique identifier for the mobile device. Unmatched devices with the user account CANNOT login to the system.
- One Time Password

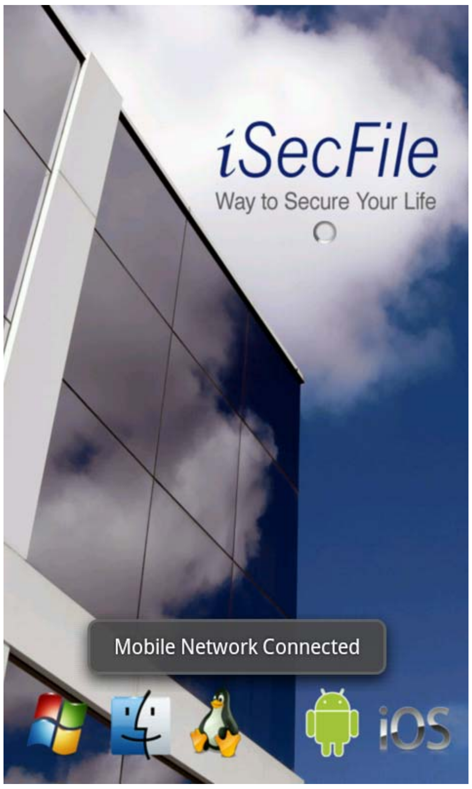
  Each token generated on the server side is one-time password. User has to scan the code within the given valid period (e.g. 60s). The token cannot be re-used or scan after the period as it is considered as invalid. Attacker cannot capture the code and re-use in the next time.

- 128 bit Random long string

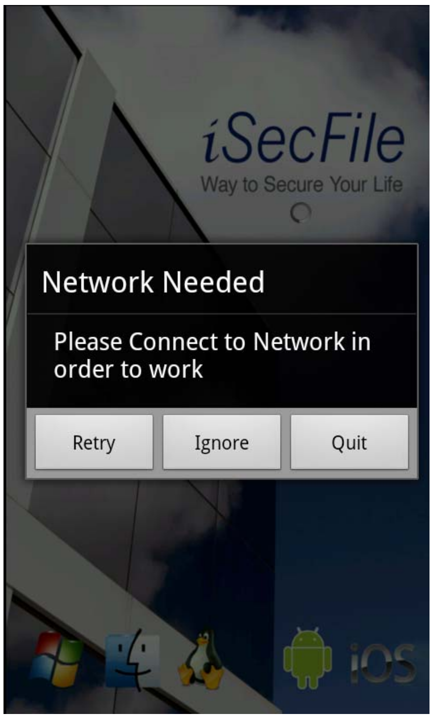
  The token is generated with 128 characters with random upper and lower case letter, numbers and symbols. It is difficult to predict or crack the code by brute force or dictionary attack.

As user does not have to remember an extra password, it is more convenient and secure to use this method to login to the system. Also, user needs both the laptop and their matched mobile device to login, the risk of account exposure is halved and the account is still intact if only one of the equipment like laptop is lost.

9.2 Application Details and User Interface



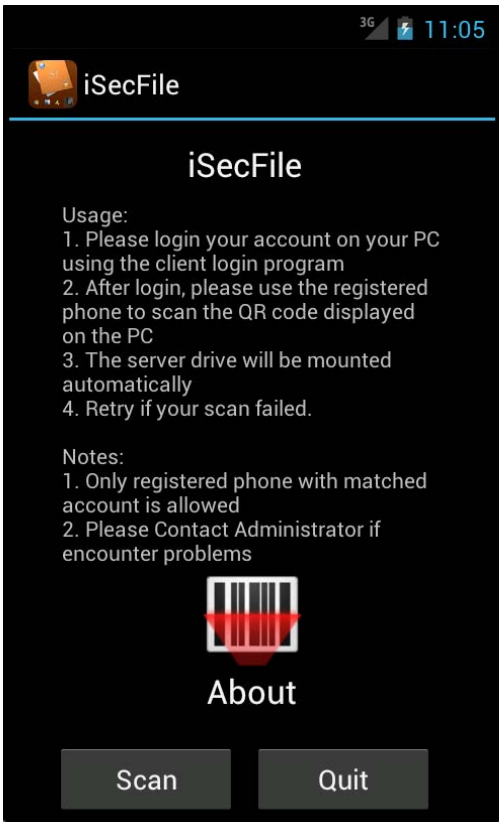
After launching the application, a splash screen will be shown containing our logo, and all the platforms we supported. In the same time, the application will also check if the device has connected to the internet. Wifi or Mobile Network is needed as we have to connect to the server for authentication.



Network Needed

Please Connect to Network in order to work

If there is no network, a dialog will be prompted for user to retry or quit the program. Choose Ignore can still enter the program, but may lead to the function working non-property (Mainly use for Developer Maintainance purpose).



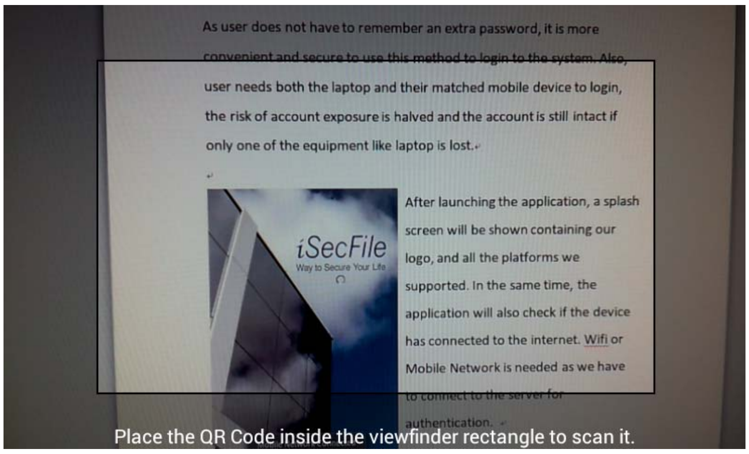
iSecFile

Usage:
1. Please login your account on your PC using the client login program
2. After login, please use the registered phone to scan the QR code displayed on the PC
3. The server drive will be mounted automatically
4. Retry if your scan failed.

Notes:
1. Only registered phone with matched account is allowed
2. Please Contact Administrator if encounter problems

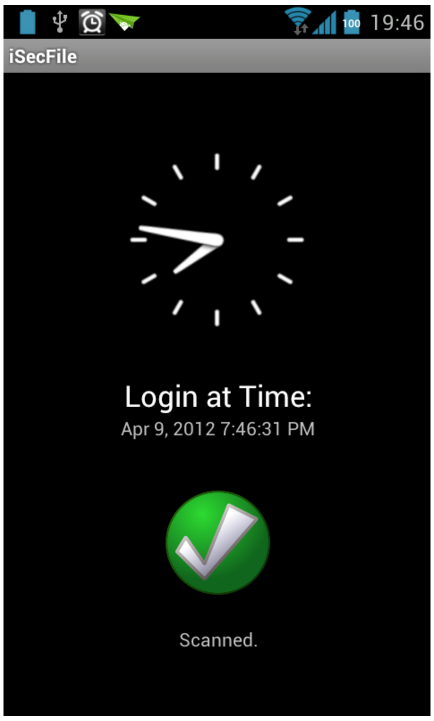
After the Splash Screen and network checking, the main page is shown. It contains mostly the usage of the application and some notes. Press the scan button will initiate the camera directly and quit button to exit the app.

The Camera is initiated and once the QR Code is scanned, it will be captured and decoded. Auto focus is also activated for accurate detection.


As user does not have to remember an extra password, it is more convenient and secure to use this method to login to the system. Also, user needs both the laptop and their matched mobile device to login, the risk of account exposure is halved and the account is still intact if only one of the equipment like laptop is lost.

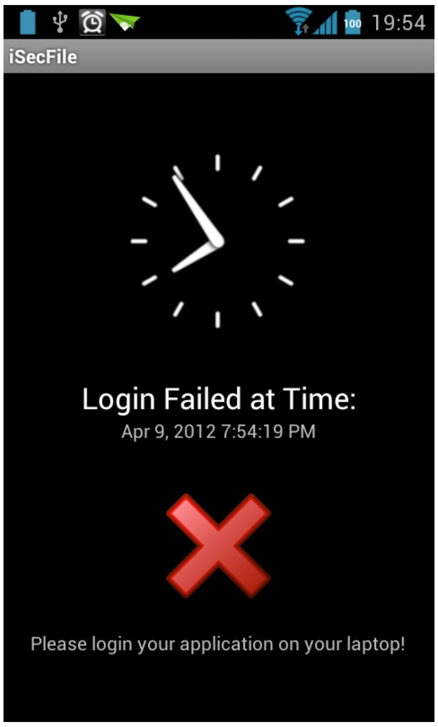
After launching the application, a splash screen will be shown containing our logo, and all the platforms we supported. In the same time, the application will also check if the device has connected to the internet. Wifi or Mobile Network is needed as we have to connect to the server for authentication.

Place the QR Code inside the viewfinder rectangle to scan it.



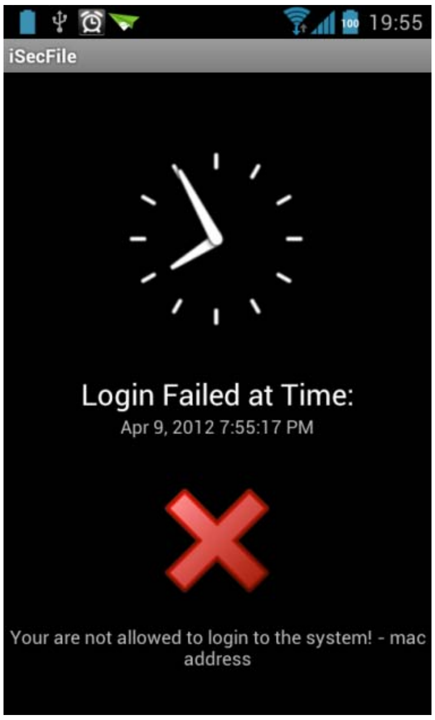
Login at Time:
Apr 9, 2012 7:46:31 PM

Scanned.



Login Failed at Time:
Apr 9, 2012 7:54:19 PM

Please login your application on your laptop!

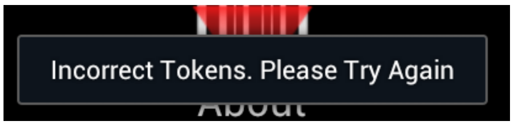


Login Failed at Time:
Apr 9, 2012 7:55:17 PM

Your are not allowed to login to the system! - mac address

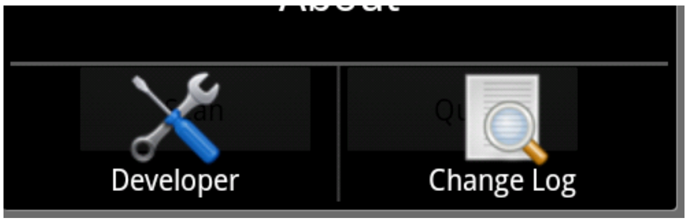
These are the screenshot of different scenario:

1. Successful Scan
2. Scan when PC hasn't login
3. Login with Un-matched devices.
4. For QR code identified not to be a token, a message will be simply shown (also to save bandwidth).

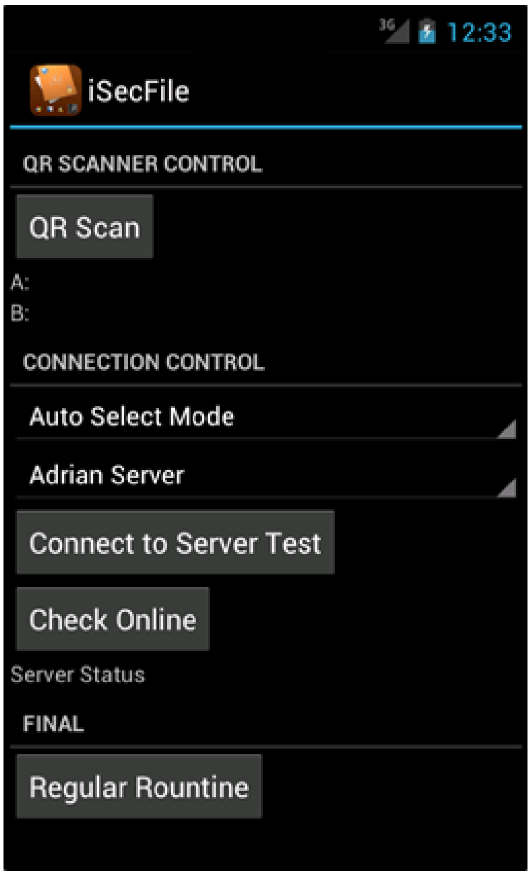

Incorrect Tokens. Please Try Again

Android mainstream Platform 2.3.3 and 4.0 are tested to be run smoothly; version 2.2 and 3.0 should be compatible while we don't have the resource to test yet.

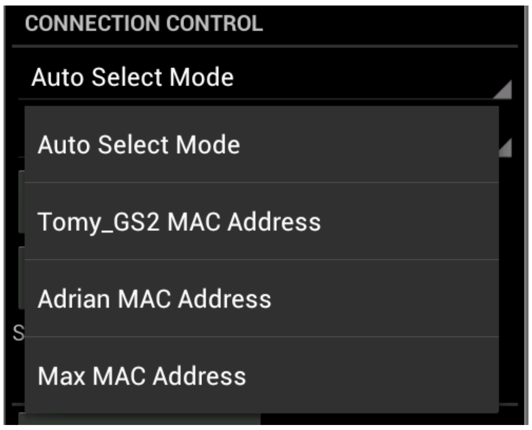
On the other hand, a developer page is added in the main page on the later phase for easy debug.



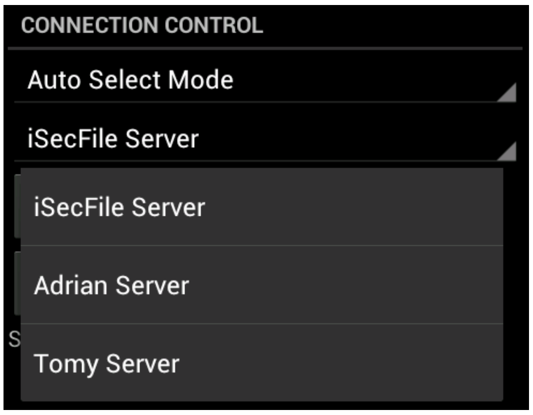
Testing of the QR Code Scanner, Internet Connection Test, Changing of target Server can be made here.



MAC Address Testing:

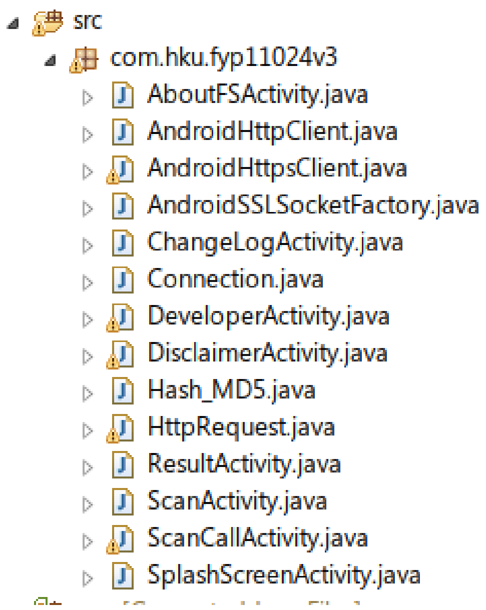


Target Server Choose:



9.3 Implementation

Different classes are implemented for different purpose of the application.


```
src
  com.hku.fyp11024v3
    AboutFSActivity.java
    AndroidHttpClient.java
    AndroidHttpsClient.java
    AndroidSSLSocketFactory.java
    ChangeLogActivity.java
    Connection.java
    DeveloperActivity.java
    DisclaimerActivity.java
    Hash_MD5.java
    HttpRequest.java
    ResultActivity.java
    ScanActivity.java
    ScanCallActivity.java
    SplashScreenActivity.java
```

9.3.1 HTTPs Connection

For the Connection to our servers, we need to initiate a HTTP Post connection and handler. On later phase, we have change to the secure HTTPS server for security. So we need a HTTPS Post handler. At the first hand, we would like to communicate between the server and the app using the secure and encrypted HTTPS 443 channel, but as the server hasn't been issued a certificate and verified by CA. Exception is being throw as error on the Android phone. In order to overcome this, we have to implement a self-created SSL socket to accept the self-signed certificate.

Initiate the HTTP Request using Custom SSL Socket:

```
public static HttpClient getHttpClient() {

    try {

        KeyStore trustStore =
        KeyStore.getInstance(KeyStore.getDefaultType());
```

```
            trustStore.load(null, null);

            SSLSocketFactory sf = new AndroidSSLSocketFactory(trustStore);

sf.setHostnameVerifier(SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER);

            HttpParams params = new BasicHttpParams();

            HttpProtocolParams.setVersion(params, HttpVersion.HTTP_1_1);

            HttpProtocolParams.setContentCharset(params, HTTP.UTF_8);

            SchemeRegistry registry = new SchemeRegistry();

        registry.register(new Scheme("http",
        PlainSocketFactory.getSocketFactory(), 80)); //HTTP Protocol

            registry.register(new Scheme("https", sf, 443)); //HTTPS Protocol

       ClientConnectionManager ccm = new ThreadSafeClientConnManager(params,
       registry);

            return new DefaultHttpClient(ccm, params);

            } catch (Exception e) {

                  return new DefaultHttpClient();

            }

      }
```

Custom SSL Socket Factory (does not throw exception):

```
public class AndroidSSLSocketFactory extends SSLSocketFactory {

      SSLContext sslContext = SSLContext.getInstance("TLS");

      public AndroidSSLSocketFactory(KeyStore truststore)

      throws NoSuchAlgorithmException, KeyManagementException,

      KeyStoreException, UnrecoverableKeyException {

            super(truststore);

            TrustManager tm = new X509TrustManager() {

                  public void checkClientTrusted(X509Certificate[] chain,
                                        String authType) throws CertificateException
{
                        }
                        public void checkServerTrusted(X509Certificate[] chain,
                                        String authType) throws CertificateException
{
                        }
                        public X509Certificate[] getAcceptedIssuers() {
                                return null;
                        }
                };
                sslContext.init(null, new TrustManager[] { tm }, null);
        }
        @Override
        public Socket createSocket(Socket socket, String host, int port,
        boolean autoClose) throws IOException, UnknownHostException {
         return sslContext.getSocketFactory().createSocket(socket, host, port,
         autoClose);
        }
        @Override
        public Socket createSocket() throws IOException {
                return sslContext.getSocketFactory().createSocket();
        }
}
```

At the later stage, when iSecFile.com is registered and got a valid site certificate, normal SSL socket is used back for valid site verified checking.

For the backup server, we use the custom SSL for encrypted data transferred.

9.3.2 AsyncTask for Internet Connection

All the tasks are implemented in the main thread for process such as sending information to data and result retrieving. This is allowed in Android Gingerbread(GB) v2.3.3, while it is depreciated and would throw an exception under new Android Ice-Cream Sandwich(ICS) v4.0. When we added support to the ICS, we spent quite a little time on it. A different thread must be declared for handling non-UI components.

```
class AsyncHttpPost extends AsyncTask<String, String, String>{

             protected void onPreExecute(){

                    //do something before thread start: initialize values

             }


             @Override

             protected String doInBackground(String... params) {

             // Thread for doing work like internet connections

             Httprequest = new Httppost();

             // Periodic return values for UI updates

             publishProgress("Packing Data...");

             //return values for PostExecute

                    return msg;

             }

          protected void onProgressUpdate(String... progress) {

             // Do something to Update the UI, retrieve values form
             publishProgress()

           }


          protected void onPostExecute(String server_response) {

             //Work done after the doInBackground thread finish like return
             and start a new page.
```

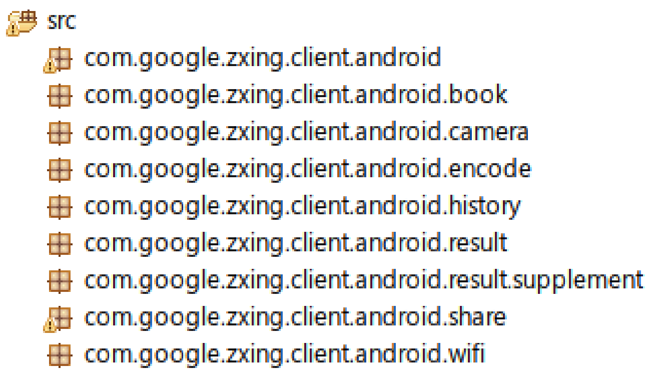
```
                    }
        }
```

This should be declared as subclass under the main UI thread, and it can be called only once per time.

9.3.3 QR Code Reader Integration

We choose to embed the whole scanner to our application for convenient. The Open source ZXing library is used under the Apache License.

We have modified and customized the code in the library. Rename of all the components for duplicated IDs, remove the unused functions such a encoding and sharing of websites. Also we restricted the user to scan only the QR Code but not the other such as 1D barcodes, ISBN codes and websites QR code to prevent malicious attack.



```
src
    com.google.zxing.client.android
    com.google.zxing.client.android.book
    com.google.zxing.client.android.camera
    com.google.zxing.client.android.encode
    com.google.zxing.client.android.history
    com.google.zxing.client.android.result
    com.google.zxing.client.android.result.supplement
    com.google.zxing.client.android.share
    com.google.zxing.client.android.wifi
```

(The whole package of the ZXing Barcode Scanner)

9.3.4 Communication between the Server and Application

As the mobile device is one the element for identification, the MAC address is sent together with the token to the target server for authentication.

```
nameValuePairs.add(new BasicNameValuePair("authentication_mac_address",
Hash_MD5.md5Encryption(macAddress)));

nameValuePairs.add(new BasicNameValuePair("type", "Android"));
nameValuePairs.add(new BasicNameValuePair("static_token",
Hash_MD5.md5Encryption(Connection.static_token)));

nameValuePairs.add(new BasicNameValuePair("token_token", tokens));
```

Four parameters are sent.

1. The device MAC address.

It can be get using WiFi Manager.

```
WifiManager wifiManager =
(WifiManager)getSystemService(Context.WIFI_SERVICE);

macAddress = wifiManager.getConnectionInfo().getMacAddress();
```

This is the best method that we can use for getting the MAC address, although the address may not be correctly retrieved when there is no network connection. So network checking is re-enforced on the first launching of the application.

Also the MAC address is hashed by MD5 to protect privacy of the user.

2. The token

The QR code scanned will be decoded back to normal 128 char long string and sent to the server.

3. The type

"Type is android" is used for the server to identify between different application such as iOS and Android.

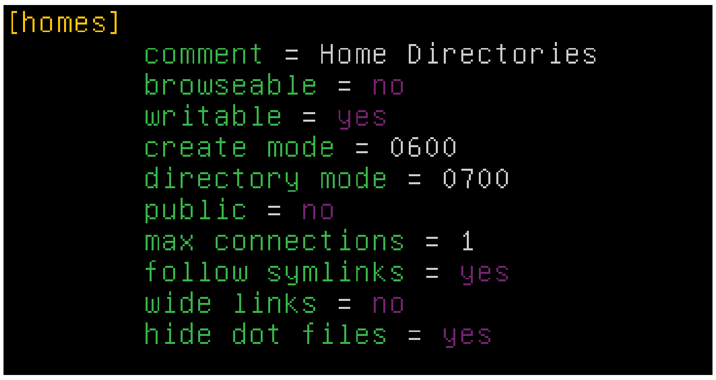
4. The static token

This is a fixed static token which will be sent along. It can be changed along with certain period of time, for example three months.

10. File Server – Samba

Although Windows server would be another nice choice, when we consider the multi-platform OS support, deal with volume creation and file system formats compatibility, we are using Samba for the file server system .As Samba is the main control of the file system, most of the settings are done here.

One of the main points is the Samba Configuration file.



```
[homes]
        comment = Home Directories
        browseable = no
        writable = yes
        create mode = 0600
        directory mode = 0700
        public = no
        max connections = 1
        follow symlinks = yes
        wide links = no
        hide dot files = yes
```

10.1 Maximum login per user

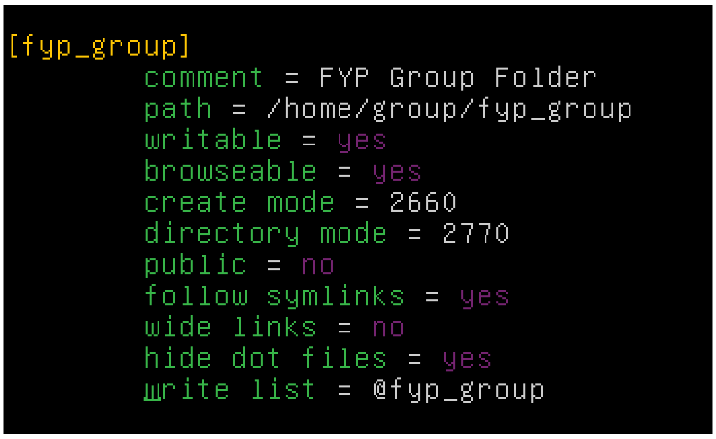
As we want to reduce the chance of repeat login, the file server will automatically reject all the connection from the outside of the same user account to prevent intruder.

10.2 User file creation mask

Files and folders permission are set to be followed the settings of Samba. 0700 for folder and 0600 for file. Only the file owner is able to access the file according to the setting. Permission denied is the result when other user trying to read others files. But there is some exception in Max OS X. The file permission may inherit the original file permission. This is due to the internal setting of Mac OS X. Root access of Mac has to be obtained to alter the OS settings.[2]

10.3 Blockage of Symbolic link

In order to prevent unauthorized access to the file system, each user is restricted to own folder only. User cannot create symbolic link to other parts of the file server.



```
[fyp_group]
        comment = FYP Group Folder
        path = /home/group/fyp_group
        writable = yes
        browseable = yes
        create mode = 2660
        directory mode = 2770
        public = no
        follow symlinks = yes
        wide links = no
        hide dot files = yes
        write list = @fyp_group
```

10.4 Group Folder

For Group Folder, the permission setting is much loses than the personal folder, as it needs to be shared among the group Members.

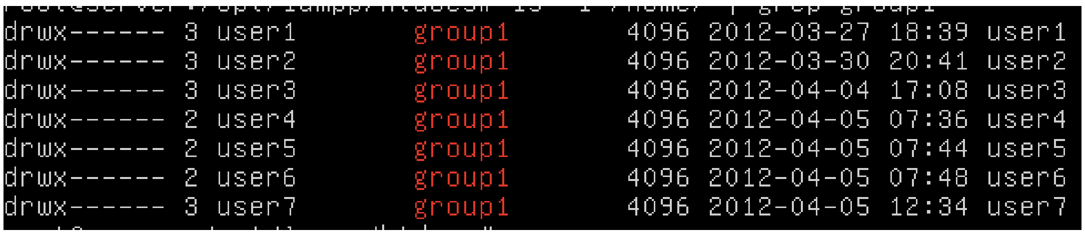
- File Creation Mode is set to 2660, which all the group members can read and write to the file inside the folder.
- Folder Creation mode is 2770, which group members can also gain accessibility
- Write list states that all the members inside the stated group can

gain the right of access to the folder.

FYP_group is one the example setting. Other grouping can also be set under similar rules.

10.5 Home folder permission

As one of the linux function ‘ls’ can list out all the content of a folder, user can go back to the upper folder followed by ‘ls’ to get access of all user name registered on the file server. Hence, the [Home] folder which contains all the user folders is set to be drwx--x--x, and belongs to root. All the other users can then be able to access through the home folder to corresponding user folder, but not able to read the [Home] folder content.

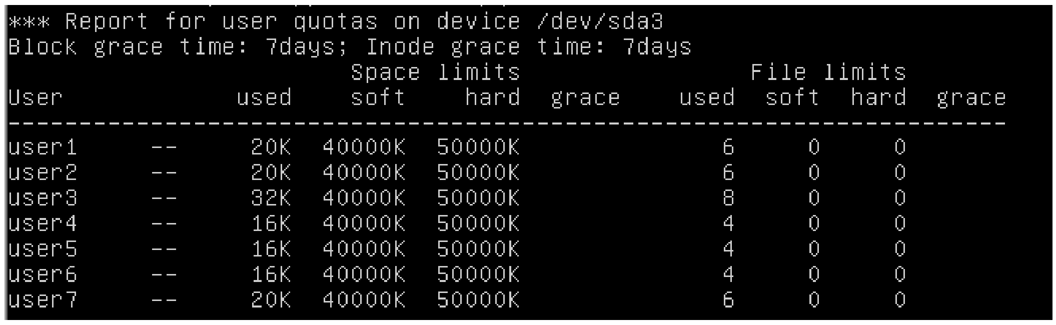


```
drwx------ 3 user1       group1      4096 2012-03-27 18:39 user1
drwx------ 3 user2       group1      4096 2012-03-30 20:41 user2
drwx------ 3 user3       group1      4096 2012-04-04 17:08 user3
drwx------ 2 user4       group1      4096 2012-04-05 07:36 user4
drwx------ 2 user5       group1      4096 2012-04-05 07:44 user5
drwx------ 2 user6       group1      4096 2012-04-05 07:48 user6
drwx------ 3 user7       group1      4096 2012-04-05 12:34 user7
```

10.6 Quota

Personal Folder

Each user has been giving a quota of free space on the server. Each user is allowed to have a soft value of amount of free space. If user has overused the space, a grace period will be given for 7 days and the maximum free space is stated by the hard value. User can no longer access the file system once the grace period has expired and the size of all files on the server still excess the soft value. A sample account with full setup is shown below.

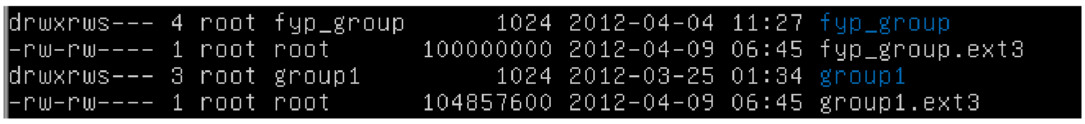


```
*** Report for user quotas on device /dev/sda3
Block grace time: 7days; Inode grace time: 7days
                        Space limits                File limits
User            used    soft     hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
user1     --     20K  40000K  50000K               6     0     0
user2     --     20K  40000K  50000K               6     0     0
user3     --     32K  40000K  50000K               8     0     0
user4     --     16K  40000K  50000K               4     0     0
user5     --     16K  40000K  50000K               4     0     0
user6     --     16K  40000K  50000K               4     0     0
user7     --     20K  40000K  50000K               6     0     0
```

On the above report, it can show all the users activities, containing their current occupied space, the soft limit and hard space limit. Here we set each user a 40MB soft and 50MB hard limit for demonstration.

Group Folder

As we have implement group folder for specific group use on the second phase, the space quota should be more than the personal folder. On linux, each user login as their own account space quota is limited by their personal account. Instead of bounding the group limit, we bound the space by setting space limit to the group folder only. This can be done by creating a virtual space of fixed size, and mounted to the certain point for entrance.



```
drwxrws--- 4 root fyp_group      1024 2012-04-04 11:27 fyp_group
-rw-rw---- 1 root root      100000000 2012-04-09 06:45 fyp_group.ext3
drwxrws--- 3 root group1         1024 2012-03-25 01:34 group1
-rw-rw---- 1 root root      104857600 2012-04-09 06:45 group1.ext3
```

Here we create a large file with desired space, and mounted group folder from these files. The limited folder for group is thus achieved.

- dd if=/dev/zero of=/home/group/$group.ext3 bs=$QUOTA_SIZE count=1

- mkfs.ext3 /home/group/$group.ext3
- mount -o loop,rw,usrquota,grpquota /home/group/$group.ext3 /home/group/$group

10.7 Account Management

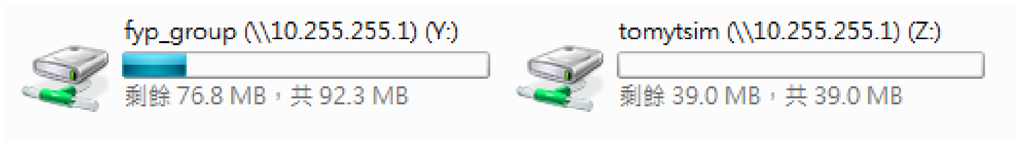
In order to become a user of Samba, it must have an UNIX account as well as Samba account. For the prevention of suspicious attack code on the file server, initially the UNIX account is set to disable so user cannot input any Linux command. Later, we found out that the UNIX account must be activated for SSH, so we changed the approach in a way that user does not have the valid shell, like using the 'nologin' shell.

Also several custom shell scripts are made for easy account management to synchronize the account information and password between the UNIX and Samba account. With the combination of the Java program, the shell scripts are called for creating or deleting user account and synchronize the password in an automated way without inputting a large amount of complex command. For example, part of the shell script like add_user.sh:

- useradd -m -g $group $username //add user in UNIX
- echo "$username:$password" | chpasswd //Change UNIX pw
- chsh -s /usr/sbin/nologin $username //Change to invalid shell
- chmod 700 /home/$username//Permission to home folder
- edquota -p sample_user -u $username //Set Quota
- sh -c "(echo $password; echo $password) | smbpasswd -as $username" //Change SMB pw

Also, password-changing script, user deletion script are also deployed for easy integration with pHp user account management.

Moreover, grouping support is also ready on the server, which different groups can be created for special purpose such as Design team, Financial Team and HR Team etc.


fyp_group (\\10.255.255.1) (Y:)
剩餘 76.8 MB，共 92.3 MB
tomytsim (\\10.255.255.1) (Z:)
剩餘 39.0 MB，共 39.0 MB

This is the picture showing when account is verified, two separated folders are mounted, one for Personal use, with 40MB space limit, and one for Open group folder, with larger space limit of 90MB.

10.8 Blockage of Unauthorized connection

Currently, under our testing, our supported platform cannot directly mount the specific space without the use of our program. However, there still might have chance for intruder trying to access the server access once they discovered the password. In order to prevent this, we have implemented a one-time password mechanism on the samba server to prevent outsider from mounting the file system without our Java Program. The idea is that every-time when user try to login to the system using our program, the server-java program will change the samba password to a one-time password, and send it to the client side for mounting use. After mounting success, the server-side Java will then change the samba password to another random string. The purpose is to prevent other user to login to the file server directly using the given username and password. With the use the pair Java program, checking and validation through our account database is necessary for initiate the server java to send the correct one-time samba password to the client side.

10.9 SSH restriction

SSH is setup on the server for secure file transfer. All the data transmitted through the internet are encrypted. On the other hand, the root login of the SSH is set to disable for security. As we mentioned above, all the account is disable of shell function which user cannot input unnecessary command to the file server system.

11. Web Server HOSTING – pHp, Apache, MySQL

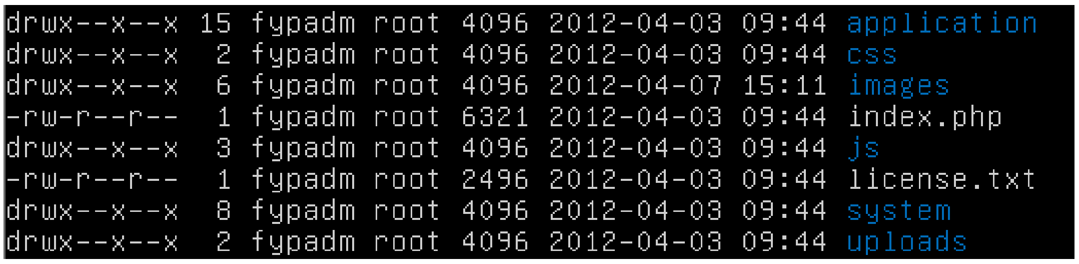
In order to support our online database connection and user account management, a server which support pHp and mobile application queries is setup. Most importantly, the security of the web server and the access right should be set properly for security reason.

11.1 File permission

To make sure the website is protected, only read accessible right is allowed to the public, and all the folders are set to be only executable for reading the files inside only, but not listing out the content of the folders. Also on the pHp side, all the unauthorized access of the pages will be redirected back to the home page.

11.2 FYP ADM

For management of the websites, a FYPADM account is created for managing the websites, pHp, MySQL database and all the related contents. All the websites materials are owned under this account. This is for centralized administration and security purpose.

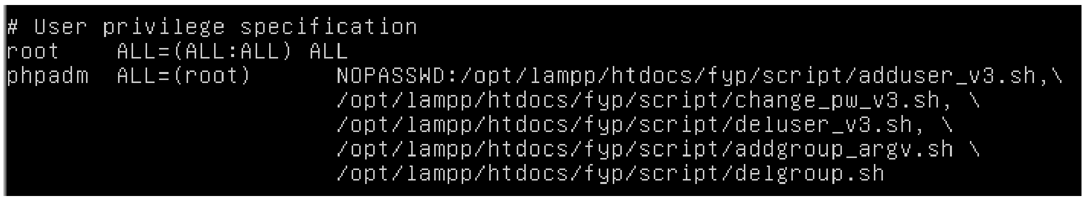


```
drwx--x--x 15 fypadm root 4096 2012-04-03 09:44 application
drwx--x--x  2 fypadm root 4096 2012-04-03 09:44 css
drwx--x--x  6 fypadm root 4096 2012-04-07 15:11 images
-rw-r--r--  1 fypadm root 6321 2012-04-03 09:44 index.php
drwx--x--x  3 fypadm root 4096 2012-04-03 09:44 js
-rw-r--r--  1 fypadm root 2496 2012-04-03 09:44 license.txt
drwx--x--x  8 fypadm root 4096 2012-04-03 09:44 system
drwx--x--x  2 fypadm root 4096 2012-04-03 09:44 uploads
```

11.3 PHP ADM

As we run pHp on the websites, there is a default user running the command behind the pHp – nobody. As in the concept of our system, all the user account management is being done on the website, with the administration account right. Only accounts with assigned Administration role will be allowed to deal with user account management. As we have to create user account and modified their password in the UNIX system, root permission is necessary for doing the jobs above. In balancing the potential security thread arouse by pHp, and the necessity of account management, "sudoer" is used.

In the sudoer, we give permission to PHPADM account to run specific script under the root right. On the other hand, we restrict all other commands using sudo rights. This is to ensure phpadm account has the lease right to give impact to the server while it is able to modify the user account. Also, as our testing, phpadm account is also being disable on remote ssh command. All the valid shell for running command is being disabled on the system.

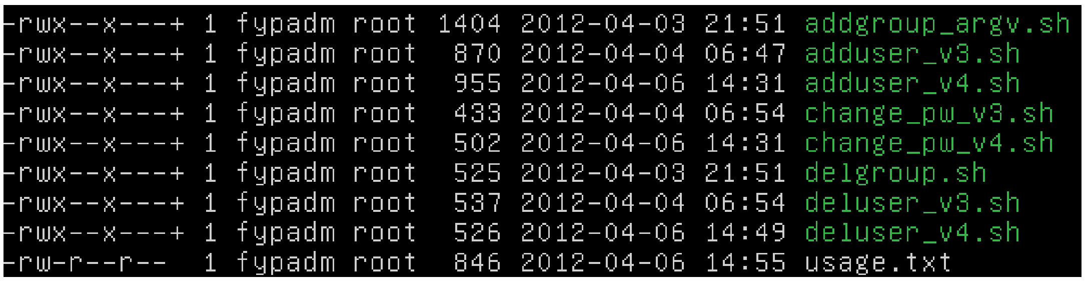


```
# User privilege specification
root    ALL=(ALL:ALL) ALL
phpadm  ALL=(root)      NOPASSWD:/opt/lampp/htdocs/fyp/script/adduser_v3.sh,\
                        /opt/lampp/htdocs/fyp/script/change_pw_v3.sh, \
                        /opt/lampp/htdocs/fyp/script/deluser_v3.sh, \
                        /opt/lampp/htdocs/fyp/script/addgroup_argv.sh \
                        /opt/lampp/htdocs/fyp/script/delgroup.sh
```

This is some part of the suoder. Inside the file, we only declare that phpadm has the right to run our specific scripts. System will reject all the other commands outside our scope. It is to ensure phpadm has the least right to run any other command on the system.

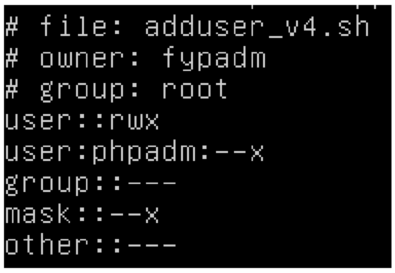
11.4 Access Control List (ACL)

For some part of the system, like the websites part, some of the files are mainly owned by the PHPADM. On the other hand, it needed to be read or executed by other user. For example, the script dealing with user account like adding user and modify user account passwords, they need to be run by the phpadm, but on the same time, they should be owned and managed under FYPADM account. PHPADM only needs the right to execute it but not to read or modify them. ACL is used for these specific cases.



```
-rwx--x---+ 1 fypadm root 1404 2012-04-03 21:51 addgroup_argv.sh
-rwx--x---+ 1 fypadm root  870 2012-04-04 06:47 adduser_v3.sh
-rwx--x---+ 1 fypadm root  955 2012-04-06 14:31 adduser_v4.sh
-rwx--x---+ 1 fypadm root  433 2012-04-04 06:54 change_pw_v3.sh
-rwx--x---+ 1 fypadm root  502 2012-04-06 14:31 change_pw_v4.sh
-rwx--x---+ 1 fypadm root  525 2012-04-03 21:51 delgroup.sh
-rwx--x---+ 1 fypadm root  537 2012-04-04 06:54 deluser_v3.sh
-rwx--x---+ 1 fypadm root  526 2012-04-06 14:49 deluser_v4.sh
-rw-r--r--  1 fypadm root  846 2012-04-06 14:55 usage.txt
```

All of the above scripts are set with ACL setting.



```
# file: adduser_v4.sh
# owner: fypadm
# group: root
user::rwx
user:phpadm:--x
group::---
mask::--x
other::---
```

Under the settings, the fypadm can read and modify the content of the script, while phpadm only have the right to executes it by passing parameters to it, such as:

./adduser_v4.sh username password group

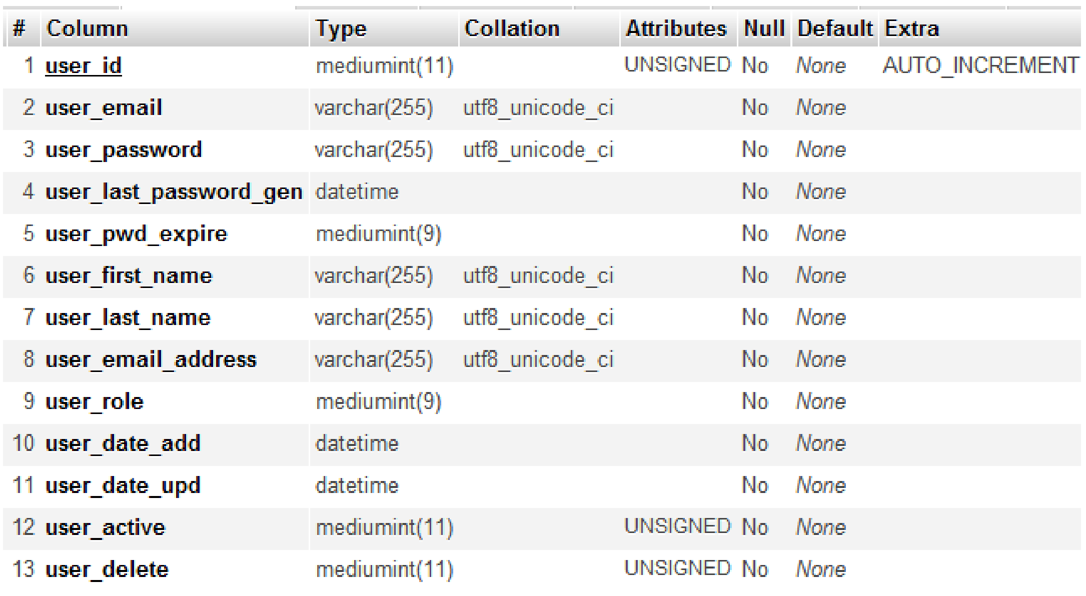
This is useful for account creation in smooth automation style as well as security measure. Outsider cannot even read the content of the script for the internal command used for the account manipulation.

11.5 MySQL

We make use of the MySQL to store our users information, such as username, passwords, matched mobile devices, account creation date, account last access time etc. It is used to keep track of all the account activity.

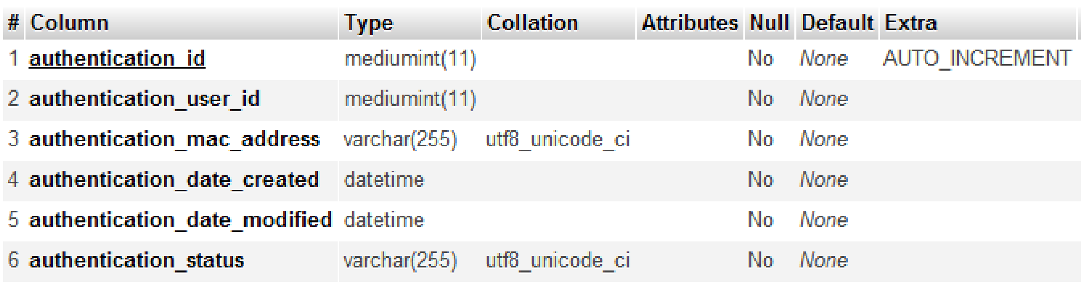
All the passwords and MAC address are encrypted with SHA1 and MD5 to protect user privacy.

User Info Table:



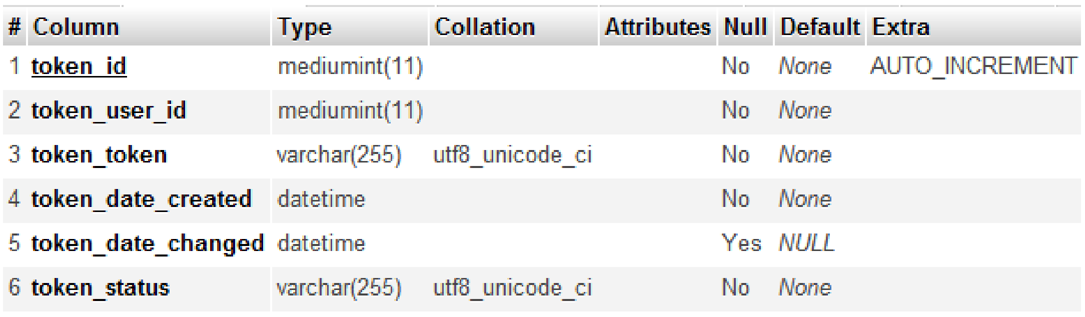
#	Column	Type	Collation	Attributes	Null	Default	Extra
1	user_id	mediumint(11)		UNSIGNED	No	*None*	AUTO_INCREMENT
2	user_email	varchar(255)	utf8_unicode_ci		No	*None*	
3	user_password	varchar(255)	utf8_unicode_ci		No	*None*	
4	user_last_password_gen	datetime			No	*None*	
5	user_pwd_expire	mediumint(9)			No	*None*	
6	user_first_name	varchar(255)	utf8_unicode_ci		No	*None*	
7	user_last_name	varchar(255)	utf8_unicode_ci		No	*None*	
8	user_email_address	varchar(255)	utf8_unicode_ci		No	*None*	
9	user_role	mediumint(9)			No	*None*	
10	user_date_add	datetime			No	*None*	
11	user_date_upd	datetime			No	*None*	
12	user_active	mediumint(11)		UNSIGNED	No	*None*	
13	user_delete	mediumint(11)		UNSIGNED	No	*None*	

User mobile Device Table:



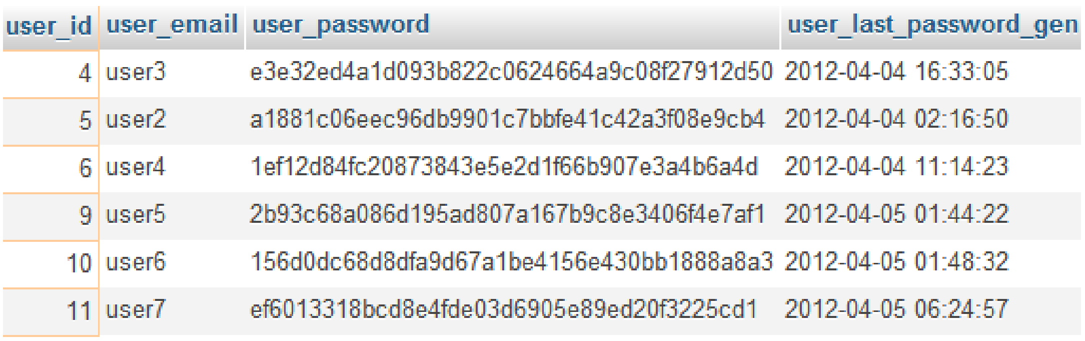
#	Column	Type	Collation	Attributes	Null	Default	Extra
1	authentication_id	mediumint(11)			No	*None*	AUTO_INCREMENT
2	authentication_user_id	mediumint(11)			No	*None*	
3	authentication_mac_address	varchar(255)	utf8_unicode_ci		No	*None*	
4	authentication_date_created	datetime			No	*None*	
5	authentication_date_modified	datetime			No	*None*	
6	authentication_status	varchar(255)	utf8_unicode_ci		No	*None*	

Token Tables:

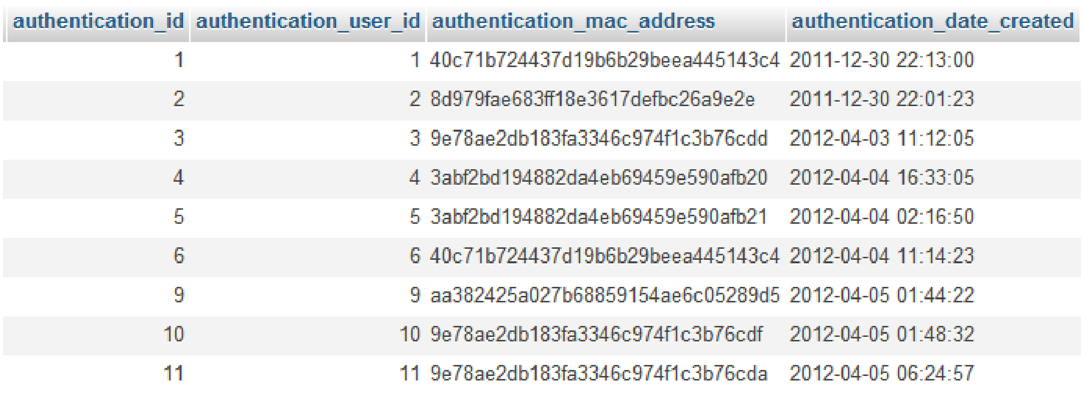


#	Column	Type	Collation	Attributes	Null	Default	Extra
1	token_id	mediumint(11)			No	*None*	AUTO_INCREMENT
2	token_user_id	mediumint(11)			No	*None*	
3	token_token	varchar(255)	utf8_unicode_ci		No	*None*	
4	token_date_created	datetime			No	*None*	
5	token_date_changed	datetime			Yes	*NULL*	
6	token_status	varchar(255)	utf8_unicode_ci		No	*None*	

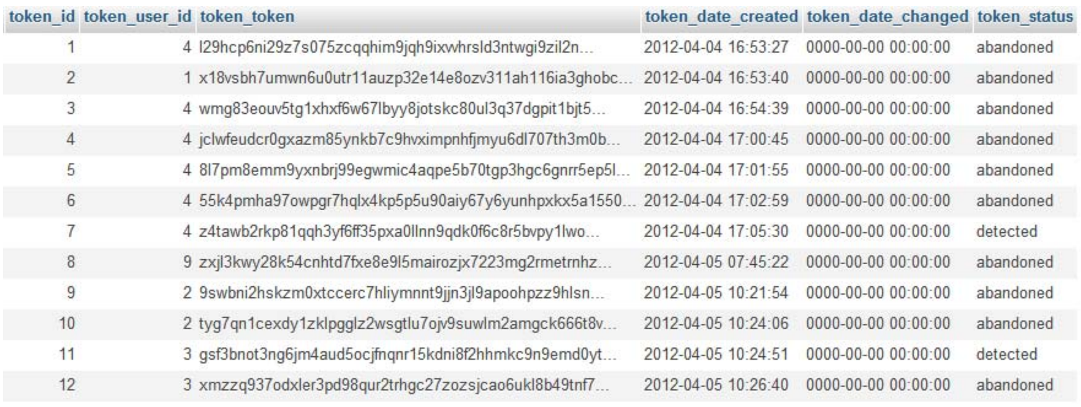
Sample:



user_id	user_email	user_password	user_last_password_gen
4	user3	e3e32ed4a1d093b822c0624664a9c08f27912d50	2012-04-04 16:33:05
5	user2	a1881c06eec96db9901c7bbfe41c42a3f08e9cb4	2012-04-04 02:16:50
6	user4	1ef12d84fc20873843e5e2d1f66b907e3a4b6a4d	2012-04-04 11:14:23
9	user5	2b93c68a086d195ad807a167b9c8e3406f4e7af1	2012-04-05 01:44:22
10	user6	156d0dc68d8dfa9d67a1be4156e430bb1888a8a3	2012-04-05 01:48:32
11	user7	ef6013318bcd8e4fde03d6905e89ed20f3225cd1	2012-04-05 06:24:57



authentication_id	authentication_user_id	authentication_mac_address	authentication_date_created
1	1	40c71b724437d19b6b29beea445143c4	2011-12-30 22:13:00
2	2	8d979fae683ff18e3617defbc26a9e2e	2011-12-30 22:01:23
3	3	9e78ae2db183fa3346c974f1c3b76cdd	2012-04-03 11:12:05
4	4	3abf2bd194882da4eb69459e590afb20	2012-04-04 16:33:05
5	5	3abf2bd194882da4eb69459e590afb21	2012-04-04 02:16:50
6	6	40c71b724437d19b6b29beea445143c4	2012-04-04 11:14:23
9	9	aa382425a027b68859154ae6c05289d5	2012-04-05 01:44:22
10	10	9e78ae2db183fa3346c974f1c3b76cdf	2012-04-05 01:48:32
11	11	9e78ae2db183fa3346c974f1c3b76cda	2012-04-05 06:24:57



token_id	token_user_id	token_token	token_date_created	token_date_changed	token_status
1	4	l29hcp6ni29z7s075zcqqhim9jqh9ixwhrsld3ntwgi9zil2n...	2012-04-04 16:53:27	0000-00-00 00:00:00	abandoned
2	1	x18vsbh7umwn6u0utr11auzp32e14e8ozv311ah116ia3ghobc...	2012-04-04 16:53:40	0000-00-00 00:00:00	abandoned
3	4	wmg83eouv5tg1xhxf6w67lbyy8jotskc80ul3q37dgpit1bjt5...	2012-04-04 16:54:39	0000-00-00 00:00:00	abandoned
4	4	jclwfeudcr0gxazm85ynkb7c9hvximpnhfjmyu6dl707th3m0b...	2012-04-04 17:00:45	0000-00-00 00:00:00	abandoned
5	4	8l7pm8emm9yxnbrj99egwmic4aqpe5b70tgp3hgc6gnrr5ep5l...	2012-04-04 17:01:55	0000-00-00 00:00:00	abandoned
6	4	55k4pmha97owpgr7hqlx4kp5p5u90aiy67y6yunhpxkx5a1550...	2012-04-04 17:02:59	0000-00-00 00:00:00	abandoned
7	4	z4tawb2rkp81qqh3yf6ff35pxa0llnn9qdk0f6c8r5bvpy1lwo...	2012-04-04 17:05:30	0000-00-00 00:00:00	detected
8	9	zxjl3kwy28k54cnhtd7fxe8e9l5mairozjx7223mg2rmetrnhz...	2012-04-05 07:45:22	0000-00-00 00:00:00	abandoned
9	2	9swbni2hskzm0xtccerc7hliymnnt9jjn3jl9apoohpzz9hlsn...	2012-04-05 10:21:54	0000-00-00 00:00:00	abandoned
10	2	tyg7qn1cexdy1zklpgglz2wsgtlu7ojv9suwlm2amgck666t8v...	2012-04-05 10:24:06	0000-00-00 00:00:00	abandoned
11	3	gsf3bnot3ng6jm4aud5ocjfnqnr15kdni8f2hhmkc9n9emd0yt...	2012-04-05 10:24:51	0000-00-00 00:00:00	detected
12	3	xmzzq937odxler3pd98qur2trhgc27zozsjcao6ukl8b49tnf7...	2012-04-05 10:26:40	0000-00-00 00:00:00	abandoned

12. ACKNOWLEDGEMENT:

1. W3Schools. (2012). OS Platform Statistics. Retrieved from http://www.w3schools.com/browsers/browsers_os.asp
2. Wikipedia. (2012). Tunneling protocol. Retrieved from http://en.wikipedia.org/wiki/Tunneling_protocol
3. JCraft, Inc. (2011). JSch – Java Secure Channel. Retrieved from http://www.jcraft.com/jsch/
4. ZXing. (n.d.). ZXing ("Zebra Crossing"). Retrieved from http://code.google.com/p/zxing/
5. Wikipedia. (2012). QR code. Retrieved from http://en.wikipedia.org/wiki/QR_code
6. Chris Chavez. (2012). XRY Software Allows Law Enforcement To Crack Your Smartphone PIN In Minutes [Video]. Retrieved from http://phandroid.com/2012/03/28/swiss-xry-software-allows-law-enforcement-to-crack-your-smartphone-pin-in-minutes-video/
7. Debian Bug tracking system. (2012). Debian Bug report logs - #532856 umask settings overridden by Mac OS X 10.5 (Leopard) clients. Retrieved from http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=532856
8. peasap. (2007). Mirror In The Sky II. Retrieved from http://www.flickr.com/photos/peasap/1680885692/
9. Dr. L.C.K. Hui, Dr. H.Y. Chung, Dr. T.W. Chim