A SECURE MOBILE SYSTEM TO SUPPORT CITIZEN JOURNALISM

CSIS0801 Final Year Project
FYP11015

Final Report

Supervisor: Dr. S.M. Yiu
Student: Lam Harry To Chun (2009177272)
# Table of Contents

Introduction .................................................................................................................. 2

1. Our Scheme ............................................................................................................ 3  
   A. Design Flow ........................................................................................................ 3  
   B. Comparison to existing scheme ........................................................................ 7  
   C. Other functions .................................................................................................. 8  

2. Assumptions ......................................................................................................... 8  

Timeline of my part .................................................................................................... 9  

Progress ...................................................................................................................... 10  

1. Setting up Azure ................................................................................................. 10  
2. Building WCF services ....................................................................................... 11  
3. Linking up phone application to WCF services ................................................. 14  
4. Building the crypto class .................................................................................... 15  
5. Designing the Ring Signature Scheme ............................................................... 18  
6. Implementing the photo selling scheme ............................................................. 21  

Testing and Performance Evaluation ......................................................................... 25  

1. Photo Encryption (AES) ...................................................................................... 26  
2. Photo Decryption (AES) ...................................................................................... 27  
3. Ring Signing Time ............................................................................................... 28  
4. Ring Verification Time ......................................................................................... 29  

Project Management .................................................................................................. 30  

Challenges .................................................................................................................. 32  

1. Problems on WCF services ................................................................................ 32  
2. Encryption on phone ........................................................................................... 32  
3. Immature SDK .................................................................................................... 33  

Further Development .................................................................................................. 34  

1. Anonymous Photo Selling .................................................................................. 34  
2. News Agencies Subscription .............................................................................. 35  
3. Map Navigation ................................................................................................... 35  
4. Blacklisting and Whitelisting ............................................................................. 35  
5. Anonymous Polling using Ring Signature .......................................................... 36  

Conclusion .................................................................................................................. 37
Introduction

Nowadays, many people have smartphones in their pocket. They use the phone camera to capture bits and pieces of everyday life. However, few people know that their photos are of much news value. In a vibrant and dynamic city like Hong Kong, news is always happening around us. News agencies want to get the first hand news and pictures of the news, but there is always a time lag between the happening of the event and the reporters arrive. In this situation, the smartphones of people near the site of event can help to capture the news in a much faster way. After that, they can sell the photos to news agencies. This is a win-win transaction as news agencies can get the latest news from the citizen photographers, and the photographers can have reward selling the photos.

Although photo sharing is easy on smartphones, it is often hard to have a secure channel to sell photos. Current implementation of photo sharing software doesn’t support encryption and digital signature. Photos may be leaked and tampered during the upload process. In sending photos to news agencies, one may even need to send photos through email. The one way process means people uploading the photo may not be guaranteed to have a reward if the photos are used in publication.

Furthermore, sometimes people may encounter dangerous situations that uploading photos may put the photographer at risk. To solve this problem, we need a tool to upload photos anonymously so that the photos are published but the identity of the photographer is not leaked.

Our project is to implement a smartphone photo sharing platform that supports transaction of photos. There are 4 main functions of the platform:

1. A two-way process selling photos with encryption and digital signature
2. Ring signature for uploading photos anonymously
3. A map to let people know the news around them
4. Categorization of photos to improve organization

In the project, we aim to solve the problem of complicated photo selling process, as well as promoting citizen journalism to report breaking news in a faster way.
1. Our Scheme

In our project, we aim to speed up the process of photo transaction by using smartphone application and cloud technology. We also try to protect the interest of the seller and buyer by encryption and digital signature. Our project consists of three main pieces: the Windows Azure cloud server, the Windows Phone 7 application and the web portal hosted on Windows Azure.

A. Design Flow

First of all, when a user encountered an event that he thinks is of news value, he can immediately take a photo using the smartphone application. Our application design ensures that the user can finish the photo capture as quickly as possible.

![Capture photo of the event](image-url)
After taking the photo, the user can categorize the photo by adding a tag so that people who are interested in such an event can find the photo quickly. The user can also add comments into the photo as a description to help the buyers to choose. When the tags and comments are set, the user can then upload the photo. The user can choose to encrypt the photo in here. If the user chooses not to encrypt it, everyone can download the photo. If he does, other users have to buy the photo in order to download it.

Figure 2 - Adding tags and comments to photo

After clicking the “upload” button, the photo will be uploaded to Windows Azure Blob storage. If the user intended to sell the photo, the photo will be encrypted with 1024bit AES encryption, while the key is stored in the phone waiting for other users to purchase. A thumbnail with watermark will also be uploaded to serve as a preview for customers to choose. The location of the photo taken is uploaded along with the photo.
Next, other users can view the thumbnail of photo along with its comments and tags in the application or web portal. If a user is interested in buying the photo, the user can click the "buy" button, along with the price he wants to give. A push notification will be sent to the seller of the photo for him to choose whether to sell the photo with the price or not.

If the seller agrees the sell the photo, the system will send a push notification back to the buyer, to notify him that the photo is ready to download.

Figure 3 - Upload encrypted photo and publish watermarked thumbnail

Figure 4 - Selling the photo
When the buyer click the download button, the aforementioned amount of e-cash$^{1}$ will be sent from the buyer to the seller, encrypted by the public key of the seller using 1024bit RSA encryption. At the same time, the AES key of the photo will be sent to the buyer, encrypted by the public key of the buyer, using 1024bit RSA encryption.

$^{1}$ The e-cash platform is considered as an external system and is not being implemented in the project.

Finally, after the buyer received the AES key, he can decrypt the original photo and save it into the photo library. There will be a check in hash value of the photo so as to ensure the photo is the correct photo the buyer is supposed to buy.

![Figure 5 - Decryption of photo and transfer of E-cash](image_url)
B. Comparison to existing scheme

The table below shows a sketch of flow in selling photo using our platform, compared to the traditional way of selling photos to news agencies:

Table 1 - Comparison between traditional approach and proposed scheme

<table>
<thead>
<tr>
<th>Traditional approach</th>
<th>Our scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Steps</strong></td>
<td><strong>Action</strong></td>
</tr>
<tr>
<td>1</td>
<td>Witness the event</td>
</tr>
<tr>
<td>2</td>
<td>Take a photo</td>
</tr>
<tr>
<td>3</td>
<td>Attach the photo to email</td>
</tr>
<tr>
<td>4</td>
<td>Send the photo to news agencies and wait for response</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>News agencies checked the photo and decided to use in publication</td>
</tr>
<tr>
<td>6</td>
<td>News agencies contact the photo seller by email/phone</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Photo seller receives an award</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As compared to the original photo selling mechanism, the photo selling process using our scheme is speeded up by a wide margin and is much more secure. The seller does not need to wait for the news agencies for response (step 4 of traditional approach) and can guarantee to receive an award after buyer downloaded the photo (Step 6 of our scheme). Also, through the control in the application, we can limit the photos sold so that it cannot be sold twice. The notification and transaction is fully automatic so that the news agencies do not need to care about contacting the photo seller.
C. Other functions

In uploading the photo, the user can choose to do a ring signature. The system will not upload the identification of the user who uploaded the photo, so as to protect him from being found by malicious people. The ring signature allows us to ensure the person who uploaded the photo is a valid user of our platform.

In addition to the photo selling mechanism, our platform allows users to find out the news happening around them. In uploading photos, no matter they are encrypted or not, the location of the photo will also be uploaded onto the server. Users can search on the map in the phone application or web portal for news nearby. For news agencies, this would be a handy function as they can find out the accurate location of the news event from the map.

The server application and database are built on Windows Azure cloud platform. This requires us to build services in order to allow communication between the smartphone application and the server. We are also building the application on Windows Phone 7 platform in order to take advantage of the sophisticated access control on Microsoft platforms.

2. Assumptions

During our project, we utilize 3G network and GPS technology. We have made the following assumptions in order to make our project possible:

i) The city has 3G network coverage with fast internet speed
ii) The firewall in the internet connection can allow communication between phone and cloud server
iii) Users has obtained digital signature from trusted authority
iv) The Windows Phone installed with the application is with the following minimum requirement:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>1 GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>512 MB</td>
</tr>
<tr>
<td>Camera</td>
<td>5MP</td>
</tr>
<tr>
<td>OS</td>
<td>Windows Phone 7.0 or above</td>
</tr>
</tbody>
</table>
Timeline of my part

I am responsible for the secured linkage between smartphone application and server. I have created a crypto library that does the encryption of photos with AES and encryption of AES keys with RSA. The crypto library has the function of doing a ring signature and verifying one. I am also responsible in building the WCF service to link up the SQL Azure database and setting up the storage on the cloud server. Finally I helped creating the part of the smartphone application that allows users to view the photos taken by other users around them, and implement the photo selling scheme on the phone application.

Table 2 - Timeline of my responsible part

<table>
<thead>
<tr>
<th>Task</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Studying and setting up Azure</td>
<td>Early September</td>
<td>Late September</td>
</tr>
<tr>
<td>Building WCF service</td>
<td>Late September</td>
<td>Late October</td>
</tr>
<tr>
<td>Linking up phone application and service</td>
<td>Late October</td>
<td>Mid November</td>
</tr>
<tr>
<td>Building the crypto class (AES/RSA)</td>
<td>Mid November</td>
<td>Mid December</td>
</tr>
<tr>
<td>Testing on encryption</td>
<td>Mid December</td>
<td>Early January</td>
</tr>
<tr>
<td>Research on ring signature</td>
<td>Early January</td>
<td>Mid February</td>
</tr>
<tr>
<td>Incorporate ring signature scheme into crypto class</td>
<td>Mid February</td>
<td>Early March</td>
</tr>
<tr>
<td>Implement photo selling scheme</td>
<td>Early March</td>
<td>Late March</td>
</tr>
<tr>
<td>Performance test</td>
<td>Late March</td>
<td>Early April</td>
</tr>
</tbody>
</table>
Progress

As aforementioned, I am responsible in 6 parts of the project, including setting up Azure, building the WCF services, linking up phone application to WCF service, building the crypto class, designing ring signature scheme and implementing the photo selling mechanism. I will focus on these 6 parts of the project, while my group mates will describe the others.

1. Setting up Azure

Since cloud server is a relatively new idea to us, we have to gather more information in order to understand how the server works. We have been using a month in testing the configurations of the server in order to meet our requirement.

![Windows Azure Platform Control Panel](image)

Figure 6 - Windows Azure Platform Control Panel
After several trials, we decided to deploy 3 services on Azure, for handling the web portal, the image upload service and the SQL Azure database respectively.

![Cloud services deployment status](image)

**Figure 7 - Cloud services deployment status**

2. Building WCF services

In order to allow the smartphone application to communicate with Azure (the cloud server), we have to build WCF services to act as a middle man. In our project, we need two services to handle database operation and image storage respectively. I am responsible for the database connection part of the project, while my groupmate, May, is responsible for the image upload service.

*Cloud database connection, SQLservice.svc*

SQLservice.svc receives the database queries from the phone and then executes them on SQL Azure (Cloud database). Originally there’s a service called Odata, which is provided by Microsoft to allow easy access to the SQL Azure. Odata would allow Windows Phone applications to send SQL queries directly to SQL Azure server and receive the returned result. However, the registration for the service is temporarily down, and yet to be resumed on late March. Since our project is ended in early April, we decided to create a workaround. We have implemented the interface ourselves. We decided to let the WCF service to handle all the database queries.

There are two kinds of methods in SQLservice.svc. One kind of them is of general use that receives queries from the phone application and execute on database. The other kind of functions performs specific functions related to the database to suit the needs of different functions on the phone.
i. **Methods that receives queries**

In creating the functions that receive queries from the phone, we need to preload the database structure into the service so that it can handle different data type. In order to make the data serializable, we have to fit the table returned into a list so that it can be transferred through the network. Also due to this problem, we have to separately create functions for executing queries for different database tables. Very unfortunately, sub-queries are not allowed using this scheme. Here is an example of the method:

```csharp
public List<Photo> selectPhotoQuery(string query)
{
    using (var context = new XamarinEntities())
    {
        ObjectQuery<Photo> reQuery = context.CreateQuery<Photo>(query);
        List<Photo> result = new List<Photo>();
        foreach (Photo temp in reQuery)
        {
            if (temp != null)
            {
                result.Add(temp);
            }
        }
        return result;
    }
}
```

**Figure 8 - Method to retrieve data from "photo" table**

In this function, we can execute a query on the table “tags”, and the function will return the result in a list. The method call from the phone would be as follow:

```
SchedulerHelper.selectPhotoQueryAsync("SELECT value photo FROM XamarinEntities.photo AS photo ORDER BY photo.id DESC");
SchedulerHelper.selectPhotoQueryCompleted += new EventHandler<SQLService.selectPhotoQueryCompletedEventArgs>(selectPhotoQueryCompleted);
```

**Figure 9 - Method call for selecting data from "photo" table**

For queries that have no returning object, we can generalize it to be run on the same method. It is made available by using a different way to connect to the database and execute the queries. Implementation is shown below:

```csharp
public int executeQueryNoReturn(string query)
{
    string conn_str = ConfigurationManager.ConnectionString[]"MyConnectionString".ConnectionString;
    SqlConnection conn = new SqlConnection(conn_str);
    conn.Open();
    SqlCommand sql = new SqlCommand(query, conn);
    int row = sql.ExecuteNonQuery();
    conn.Close();
    return row;
}
```

**Figure 10 - Method for executing query with no return value**
ii. Methods of specific use

In creating methods that perform specific functions, we accept parameters from the method call and return specific value that the caller wants to get. Here is an example of this kind of functions that returns the cipher of the photo from photo ID provided: The calling of the method is as follow:

![Figure 11 - Calling of the method to retrieve cipher of photo of corresponding ID](image)

```csharp
public string getkeyCipherFromID(int photoID)
{
    string conn_str = ConfigurationManager.ConnectionStrings["NHREConnectionString"].ConnectionString;
    SqlConnection conn = new SqlConnection(conn_str);
    conn.Open();
    string sql = "SELECT keycipher FROM buy WHERE photoid = " + photoID;
    SqlCommand cmd = new SqlCommand(sql, conn);
    SqlDataReader reader = cmd.ExecuteReader();
    reader.Read();
    return (string)reader["keycipher"];}
```

 ![Figure 12 - Method to retrieve cipher of photo with corresponding ID](image)

There are both advantages and disadvantages in using these two kinds of methods. The former one is more like traditional SQL queries but cannot handle sub-queries. Complex queries cannot be run. Also, there is always more overhead in data transfer as the table is returned from the cloud server to the phone application and let the phone process on the returned table.

For the later kind of methods, they lack flexibility as each method can only serve one purpose, but it can handle more complex queries. The cloud server handles the returned tables so that we can save more resources on the phone. There are fewer overheads in data transfer, as only a value is returned.

We think that the later kind of methods are more suitable to be used in companion with a phone application, as data transfer is expensive on mobile phones, and phones often have limited resources. In later development stage, most of the functions in the later development stage are designed in the second way.
3. Linking up phone application to WCF services

To allow the smartphone application to call the WCF services, we have built a class, `UploadHelper.cs`, to handle the upload of photo. The class handles the creation of blob on Azure Blob Storage, initializes the storage using the WCF service, and uses the returned URI to upload files.

```csharp
public void uploadObject()
{
    var client = new UploadServiceClient();
    client.UploadUrlWithSharedAccessSignatureCompleted += UploadUrlWithSharedAccessSignatureCompleted;
    client.UploadUrlWithSharedAccessSignatureAsync(Build.NewBuild().ToString());
}

private void UploadUrlWithSharedAccessSignatureCompleted(object sender, UploadUrlWithSharedAccessSignatureCompletedEventArgs e)
{
    if (e.Error != null)
    MessageBox.Show(e.Error.Message.ToString());
    if (e.Error == null)
    {
        // Determine upload path - Add filename to container path
        var builder = new UriBuilder(e.Result);
        builder.Path = builder.Path + '/' + Filename;
        var blobUrl = builder.Uri;
        // Open the image file from isolates storage to read from
        IsolatedStorageFileStream file = IsolatedStorageFile.GetUserStoreForApplication().OpenFile(Filename, FileMode.Open, FileAccess.Read);
        // Create the uploader and kick off the uploader
        var uploader = new CloudBlobUploader(file, blobUrl.AbsoluteUri);
        var pos = blobUrl.AbsoluteUri.ToString().IndexOf('?');
        SegUrl = blobUrl.AbsoluteUri.ToString().Substring(0, pos);
        uploader.UploadFinished += (s, args) =>
        {
            IsolatedStorageFile.GetUserStoreForApplication().DeleteFile(Filename);
            uploadCompleted(this, e);
        }
        uploader.StartUpload();
    }
}
```

Figure 3 – Methods for uploading photos

The complication of such a method is that it is not like uploading files to ordinary file servers. We have to first create a blob on the storage, generate an URI, request for permission to upload, and then actually uploading to the storage.

We can also see in the code that the read and write of file in Windows Phone applications are not the same as other phones. We have to use an Isolated Storage to store the photo taken, and then call the method to upload the photo to the storage. We are taking this approach as this makes the method and WCF service highly reusable. Next time when we need to upload files to another Azure Blob Storage, we just need to change the connection string. This will be beneficial to the further development of our projects as well as others trying to use Azure cloud servers.
4. Building the crypto class

In order to support the encryption and decryption of photos using AES, as well as the encryption and decryption of AES keys using RSA, we have created a crypto helper class to serve the purpose. The crypto class also provides ring signature and verification services. In here, we break down the class into 3 parts:

i. RSA security service

At first we need to generate RSA key pairs. We cannot find keys issued by trusted authorities so the generated keys are only serving for testing purposes. The keys are generated with 1024 bit strength. The key pairs generated are compatible to be used in .NET 4.0 Framework RSACryptoServiceProvider. In fact, real RSA keys can be easily imported to the library as it follows the same set of standards.

![Figure 14 - Generate RSA key pairs](image)

The keys are generated at the first launch of the application. As long as the application is not uninstalled, the key will be stored in isolated storage inside the application. The key cannot be regenerated so each installation is stick to one user account.
For RSA encryption and decryption, we use the following methods:

```java
public byte[] RSACrypt(string obj)
{
    // Convert your text to a byte array
    byte[]/rawBytes = System.Text.Encoding.UTF8.GetBytes(obj);
    // Encrypt your raw bytes and return the encrypted bytes
    byte[]/encryptedBytes = RSA.Encrypt(rawBytes);
    return encryptedBytes;
}

public string RSAEncrypt(byte[]/encryptedBytes)
{
    // Now decrypt your encrypted bytes
    byte[]/decryptedBytes = RSA.Decrypt(encryptedBytes);
    // Convert your decrypted bytes back to a string
    return System.Text.Encoding.UTF8.GetString(decryptedBytes, 0, decryptedBytes.Length);
}
```

Figure 15 - Methods for RSA encryption and decryption

The RSA encrypt method receives a Base64 string and encrypts it into a byte array with an imported public key, and the RSA decrypt method decrypts the byte array into a Base64 string with the imported private key.

### ii. AES security service

After initializing the crypto class, we can encrypt the photo using AES. The key is generated with HMACSHA1 pseudo-random number generator and the iteration count is 5000. The method accepts the byte array containing the data to be encrypted and returns the string that is the cipher of the AES encryption process.

```java
public string AEDecrypt(byte[]/dataToEncrypt)
{
    Asymmetric aes = null;
    MemoryStream memoryStream = null;
    Cryptostream cryptostream = null;
    try
    {
        // Generate a key based on a Password and HMACSHA1 pseudo-random number generator
        Rfc2898DeriveBytes rfc2898 = new Rfc2898DeriveBytes(AESPassword, Encoding.UTF8.GetBytes(AESPassword), 5000);
        // Create AES algorithm
        aes = new Asymmetric();
        // Key derived from byte array with 32 pseudo-random key bytes
        aes.Key = rfc2898.GetBytes(16);
        // IV derived from byte array with 16 pseudo-random key bytes
        aes.IV = rfc2898.GetBytes(16);
        // Create Memory and Crypto Streams
        memoryStream = new MemoryStream();
        cryptostream = new Cryptostream(memoryStream, aes.CreateEncryptor(), CryptostreamMode.Write);
        // Encrypt Data
        byte[]/data = dataToEncrypt;
        cryptostream.Write(data, 0, data.Length);
        cryptostream.FlushFinalBlock();
        // Return Base 64 string
        return Convert.ToBase64String(memoryStream.ToArray());
    }
    finally
    {
        if (cryptostream != null) cryptostream.Close();
        if (memoryStream != null) memoryStream.Close();
        if (aes != null) aes.Clear();
    }
}
```

Figure 16 - Method for encryption using RSA
The method for decrypting data using AES is similar to the AES encryption method, just that the method is doing the reverse, receiving the cipher as a Base64 string and return a byte array as decrypted data.

```csharp
public byte[] AESdecrypt(string dataToDecrypt)
{
    AesManaged aes = null;
    MemoryStream memoryStream = null;

    try {
        //Generate a Key based on a Password and HMACSHA1 pseudo-random number generator
        //Must be at least 8 bytes long
        //Use an iteration count of at least 10000
        Rfc2898DeriveBytes rfc2898 = new Rfc2898DeriveBytes(AESpassword, Encoding.UTF8.GetBytes(AESpassword), 5000);

        //Create AES algorithm
        aes = new AesManaged();
        //Key derived from byte array with 32 pseudo-random key bytes
        aes.Key = rfc2898.GetBytes(32);
        //IV derived from byte array with 16 pseudo-random key bytes
        aes.IV = rfc2898.GetBytes(16);

        //Create Memory and Crypto Streams
        memoryStream = new MemoryStream();
       CryptoStream cryptoStream = new CryptoStream(memoryStream, aes.CreateDecryptor(), CryptoStreamMode.Write);

        //Decrypt Data
        byte[] data = Convert.FromBase64String(dataToDecrypt);
        cryptoStream.Write(data, 0, data.Length);
        cryptoStream.FlushFinalBlock();

        //Return Decrypted String
        byte[] decryptedBytes = memoryStream.ToArray();
        //Dispose
        if (cryptoStream != null)
            cryptoStream.Dispose();
        //Return the decrypted bytes;
    }
    return decryptedBytes;
}
```

**Figure 17 - Method for decryption using AES**
5. Designing the Ring Signature Scheme

In order to support anonymous upload, we need to apply a ring signature scheme. There are currently no libraries that support ring signature in C# or Silverlight. Therefore, we have to implement the ring signature schemes from mathematical formulas.

In the project, we are implementing a ring signature scheme based on Schnorr’s signature. The mathematical formulas are as follow:

For signing message $M$,

i) Choose random number $r$
ii) Let $R = g^k$
iii) Let $h = H(M \parallel r)$
iv) Let $z = (k - xe)$

The signature pair is $(z, h)$, while $x$ is the private key of the user, $\parallel$ denotes concatenation and $H()$ denotes hash function.

For verifying message $M$ with signature pair $(z, h)$,

i) $z_v = g^{se}$
ii) $h_v = H(M \parallel r_v)$

If $h_v = h$, the signature is verified.

To extend the Schnorr’s signature to a ring signature scheme, we need to perform significant modification.

Suppose we are signing a message $M$ with user key pair $(Y_1, x_1)$,

i) Choose two public key $Y_2$ and $Y_3$ from a pool of users
ii) Generate fake $(z_2, h_2)$ and $(z_3, h_3)$ signature pairs
iii) $R_2 = g^{z_2 Y_2 h_2}$, $R_3 = g^{z_3 Y_3 h_3}$
iv) Generate random number $r$
v) $R_1 = g^r$
vi) $h = H(M \parallel R_1 \parallel R_2 \parallel R_3 \parallel Y_1 \parallel Y_2 \parallel Y_3)$
vii) $h_1 = (h + h_1 + h_2)$
viii) $z_1 = r - h_1 x_1$
The ring signature would then be \((z_1, h_1, z_2, h_2, z_3, h_3)\). The ring-signing scheme is implemented as follow:

```java
public BigInteger[] ringsign(String msg, BigInteger Z2, BigInteger Y3)
{
    // fake signature (Z2, H2)
    BigInteger Z2 = new BigInteger();
    Z2.gennamemult(1024, R);
    BigInteger H2 = new BigInteger();
    H2.gennamemult(1024, R);
    BigInteger R2 = (g.modPow(Z2, p) * Y2.modPow(H2, p)) % p;

    // fake signature (Z3, h3)
    BigInteger Z3 = new BigInteger();
    Z3.gennamemult(1024, R);
    BigInteger H3 = new BigInteger();
    H3.gennamemult(1024, R);
    BigInteger R3 = (g.modPow(Z3, p) * Y3.modPow(H3, p)) % p;

    // generate real signature
    BigInteger r = new BigInteger();
    r.gennamemult(1024, R);
    BigInteger R1 = g.modPow(r, p);
    byte[] mm = encoding.getBytes(mm);
    byte[] kbb = H1.getBytes();
    byte[] kbb = H2.getBytes();
    byte[] kbb = H3.getBytes();
    byte[] xbb = Y1.getBytes();
    byte[] ybb = Y2.getBytes();
    byte[] ybb = Y3.getBytes();
    byte[] concat = new byte[mm.length + R2.length + R3.length + R1.length + R2.length + R3.length + R1.length];
    System.arraycopy(mm, 0, concat, 0, mm.length);
    System.arraycopy(kbb, 0, concat, mm.length, kbb.length);
    System.arraycopy(kbb, 0, concat, mm.length + kbb.length, kbb.length);
    System.arraycopy(kbb, 0, concat, mm.length + kbb.length + kbb.length, kbb.length);
    System.arraycopy(xbb, 0, concat, mm.length + kbb.length + kbb.length + kbb.length, xbb.length);
    System.arraycopy(ybb, 0, concat, mm.length + kbb.length + kbb.length + kbb.length + xbb.length, ybb.length);
    System.arraycopy(ybb, 0, concat, mm.length + kbb.length + kbb.length + kbb.length + xbb.length + ybb.length, ybb.length);
    byte[] hash = HighCore.gethash(concat);
    BigInteger H = new BigInteger(hash);
    BigInteger H1 = (h + h2 + h3) % h;
    BigInteger t1 = r * H1 % x1;
    BigInteger[] signature = new BigInteger[3, 2];
    for (int i = 0; i < 3; i++)
    {   signature[i, ] = new BigInteger();
    }
    signature[0, 0] = x1;
    signature[0, 1] = H1;
    signature[1, 0] = x2;
    signature[1, 1] = H2;
    signature[2, 0] = x3;
    signature[2, 1] = H3;
    return signature;
}
```

There are minor modifications to meet computing requirements, but the implementation is closely following the mathematical formulas.
To verify message $M$ with signature $(z_1, h_1, z_2, h_2, z_3, h_3)$,

i) $R'_1 = g^{z_1}Y_1^{h_1}$

ii) $R'_2 = g^{z_2}Y_2^{h_2}$

iii) $R'_3 = g^{z_3}Y_3^{h_3}$

iv) $h' = H(M || R'_1 || R'_2 || R'_3 || Y_1 || Y_2 || Y_3)$

If $(h' + h_2 + h_3 = h_1)$ then the signature is verified. The ring verification is implemented as follow:

```java
public bool ringVerify(string m, BigInteger[] signature, BigInteger Y1, BigInteger Y2, BigInteger Y3) {
    System.Text.UTF8Encoding encoding = new System.Text.UTF8Encoding();
    BigInteger z1 = signature[0];
    BigInteger h1 = signature[1];
    BigInteger z2 = signature[2];
    BigInteger h2 = signature[3];
    BigInteger z3 = signature[4];
    BigInteger h3 = signature[5];

    BigInteger R'_1 = (g.modPow(z1, p) * Y1.modPow(h1, p)) % p;
    BigInteger R'_2 = (g.modPow(z2, p) * Y2.modPow(h2, p)) % p;
    BigInteger R'_3 = (g.modPow(z3, p) * Y3.modPow(h3, p)) % p;

    byte[] m = encoding.GetBytes(m);
    byte[] R'_1 = R'_1.getBytes();
    byte[] R'_2 = R'_2.getBytes();
    byte[] R'_3 = R'_3.getBytes();
    byte[] Y1 = Y1.getBytes();
    byte[] Y2 = Y2.getBytes();
    byte[] Y3 = Y3.getBytes();

    byte[] concat2 = new byte[m.Length + R'_1.Length + R'_2.Length + Y1.Length + Y2.Length + Y3.Length];
    System.Buffer.BlockCopy(m, 0, concat2, 0, m.Length);
    System.Buffer.BlockCopy(R'_1, 0, concat2, m.Length, R'_1.Length);
    System.Buffer.BlockCopy(R'_2, 0, concat2, m.Length + R'_1.Length, R'_2.Length);
    System.Buffer.BlockCopy(Y1, 0, concat2, m.Length + R'_2.Length, Y1.Length);
    System.Buffer.BlockCopy(Y2, 0, concat2, m.Length + Y1.Length + R'_2.Length, Y2.Length);
    System.Buffer.BlockCopy(Y3, 0, concat2, m.Length + Y2.Length + R'_2.Length + Y1.Length, Y3.Length);

    byte[] h_m = MD5Core.GetMD5(concat2);
    BigInteger h_m = new BigInteger(h_m);
    if ((h_m + h_2 + h_3) % h_1 == 0) {
        return true;
    } else {
        return false;
    }
}
```

Figure 19 - Method for ring signature verification
6. Implementing the photo selling scheme

In the design of photo selling scheme, we integrate bits and pieces developed in our project to create a scheme to sell photos securely and efficiently. The steps of the photo selling process are described below:

Step 1: When the user encounters an event, he can take a photo of it. (Camera capture is handled by May) At the same time, a thumbnail of the photo with a watermark is created. (Eric handles watermark and thumbnail creation, mentioned earlier in the report)

Step 2: The user can add comments on the photo as well as categorize them with tags. (Ray handles comments and tags, mentioned earlier in the report) The tags will enable photo buyers to find the photos they are interested quickly.

Step 3: User clicks the “upload“ button. The photo is encrypted by AES and uploaded to Azure Blob Storage. The thumbnail is also uploaded but without encryption, for other users to preview. The original photo is not stored in the phone so that the user cannot sell twice.

Figure 20 - AES encryption of photo
Step 4: When another user finds the photo and wants to buy it, he can enter the price he wants to pay in photo page and click the “buy” button. A push notification with price information will be sent to the seller of the photo.

Figure 21 - Methods for photo purchase
Step 5: The seller accepts the deal. The system will obtain the public key of the buyer from the database. The AES key is encrypted by the public key of buyer using RSA and sent to the database. A push notification will be sent to the photo buyer to notify the accepted offer.

Step 6: The buyer encrypts e-cash and sent to the seller. (E-cash is not implemented in the project) At the same time, the seller takes the encrypted AES key from database and decrypts it using his private key.
Step 7: The buyer obtained the AES key and decrypts the photo. The buyer will verify the photo with the hash value of the original photo. The photo will be saved to the picture library of the phone.

The scheme will ensure the sellers can sell the photos quickly and guarantee them to receive an award after the photo is sold. In the buyers’ point of view, they can buy photos without caring about contacting the sellers, and can easily find the photos they need.
Testing and Performance Evaluation

In the performance evaluation plan, we mainly focus on mainly 4 areas:

- Photo Encryption (AES)
- Photo Decryption (AES)
- Photo Signing with Ring Signature
- Ring Signature Verification

In the usage of the application, we found that the above areas are the most time consuming and resource hungry areas. Therefore, through analyzing the performance figures in the above areas, we can find out the bottleneck of the application.

The performance evaluation was carried out on the following 2 phones. Their specifications as shown:

<table>
<thead>
<tr>
<th></th>
<th>HTC 7 Mozart</th>
<th>HTC Radar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chipset</td>
<td>Qualcomm QSD8250 Snapdragon</td>
<td>Qualcomm QSD8255 Snapdragon</td>
</tr>
<tr>
<td>CPU</td>
<td>1 GHz Scorpion</td>
<td>1 GHz Scorpion</td>
</tr>
<tr>
<td>RAM</td>
<td>576 MB</td>
<td>512 MB</td>
</tr>
<tr>
<td>OS</td>
<td>Windows Phone 7 Build 7740</td>
<td>Windows Phone 7.5 Build 8107</td>
</tr>
</tbody>
</table>

We have tested the phone in the above 4 areas under 4 different resolutions, which are 640x480 (0.3 MP), 1280x960 (1.23 MP), 1600x1200 (1.92 MP) and 2048x1536 (3.14 MP). Each resolution was tested 10 times on each phone, and the average time taken was used to plot the graph to see the performance.
1. Photo Encryption (AES)

![AES Encryption Time Against Resolution](image)

**Findings:**

From the graph, we can see that the photo encryption time increases linearly with the number of pixels. This is a good sign as we can see that there is no bottleneck in encryption process under the tested resolutions. From the figures, we can also see that the newer CPU in Radar has better performance. We can foresee that the time needed for encryption decrease as technology advance.
2. Photo Decryption (AES)

**Findings:**

The AES decryption time increases linearly with the increase in resolution of photo, which is the same as in the findings of AES encryption time.
3. Ring Signing Time

![Ring Signature Time Against Resolution](image.png)

**Findings:**

From the graph, we can see that the ring signature time mostly remain constant in different resolution. This is expected because the ring signature signs on the 128 bit MD5 hash value of the photo. The figures obtained from performance evaluation confirm the prediction. Moreover, we can see that ring signature is a very slow process on the phone. We believe that the slow performance is due to unoptimized code in the ring signature method. The ring signature scheme is coded directly from mathematical formulas, so hardware accelerations are not supported. We believe that in optimizing the code, the performance of ring signature can be greatly increased.
4. Ring Verification Time

Findings:

Similar to signing, the time taken to verify a ring signature is independent to photo resolution. This is because the verification is done on the 128 bit MD5 hash value of photo. In addition, the verification is done in the cloud server, so that it would be less resource-hungry for the phone.
# Project Management

## System Infrastructure and Database Connection

<table>
<thead>
<tr>
<th>Task</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting up Windows Azure</td>
<td>May</td>
</tr>
<tr>
<td>Setting up SQL Azure</td>
<td>May</td>
</tr>
<tr>
<td>Setting up Access Control Service</td>
<td>May</td>
</tr>
<tr>
<td>Connection between Windows Phone to SQL Azure</td>
<td>Harry</td>
</tr>
<tr>
<td>Setting up Image Service</td>
<td>May</td>
</tr>
<tr>
<td>Push Notification</td>
<td>Ray</td>
</tr>
<tr>
<td>Connection between Web Portal to SQL Azure Database</td>
<td>May</td>
</tr>
</tbody>
</table>

## Security

<table>
<thead>
<tr>
<th>Task</th>
<th>Contributor(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building the crypto class</td>
<td>Harry</td>
</tr>
<tr>
<td>Ring Signature Scheme</td>
<td>Harry</td>
</tr>
<tr>
<td>Session Key</td>
<td>May, Ray</td>
</tr>
<tr>
<td>Water mark</td>
<td>Eric</td>
</tr>
</tbody>
</table>

## Apps

<table>
<thead>
<tr>
<th>Task</th>
<th>Contributor(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface</td>
<td>Ray</td>
</tr>
<tr>
<td>Application login and Access Control</td>
<td>May</td>
</tr>
<tr>
<td>Photo Upload Service</td>
<td>Harry</td>
</tr>
<tr>
<td>Transaction</td>
<td>Harry</td>
</tr>
<tr>
<td>Map Page</td>
<td>Eric</td>
</tr>
<tr>
<td>Album Page</td>
<td>Eric</td>
</tr>
<tr>
<td>Photos Storing</td>
<td>Harry</td>
</tr>
</tbody>
</table>

## Web Function

<table>
<thead>
<tr>
<th>Task</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>May</td>
</tr>
<tr>
<td>Access Control</td>
<td>May</td>
</tr>
<tr>
<td>Album Page</td>
<td>May</td>
</tr>
<tr>
<td>Transaction</td>
<td>May</td>
</tr>
<tr>
<td>Map Page</td>
<td>May</td>
</tr>
</tbody>
</table>
# Project Timeline

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Plan</td>
<td>All</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web Page</td>
<td>All</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Studies on each topic</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Study on Security Linkage</td>
<td>Harry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Study on Windows Phone 7 apps</td>
<td>Eric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Study on Web Portal</td>
<td>Ray</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Study on Windows Azure</td>
<td>May</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Implementation on App</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encryption of the Photo</td>
<td>Harry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Map Function</td>
<td>Eric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other functions</td>
<td>Ray</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame and the linkage to Azure</td>
<td>May</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Implementation on Windows Azure</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Features</td>
<td>Harry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Azure with Bing Map</td>
<td>Eric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other Function</td>
<td>Ray</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access Control</td>
<td>May</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Implementation on App and Web Portal</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Features on App</td>
<td>Harry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Photo Page on App</td>
<td>Eric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Interface on App</td>
<td>Ray</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web Portal</td>
<td>May</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mid-term Review and interim Report</td>
<td>All</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Implementation on App and Web Portal</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transaction</td>
<td>Harry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Water Mark</td>
<td>Eric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Push Notification</td>
<td>Ray</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web Portal And Access Control</td>
<td>May</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>Eric and Harry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final Report</td>
<td>All</td>
<td>All</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presentation and Exhibition</td>
<td>All</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Challenges

We encountered several problems during the course of our project. Problems mainly lies on the WCF services, as the system architecture is relatively new to other platforms, there are not much references that we can make use of. Another problem is about the encryption and decryption process. As we have to take care of the performance of the process on slow phones, the encryption and decryption process on phone are a bit troublesome.

1. Problems on WCF services

At the very beginning, we tried to use the Windows Phone Azure toolkit to handle the upload of photos. However, the complicated toolkit drives us to use many custom built libraries and we have to amend the flow of our application in order to fit the service. Moreover, the usage of toolkit is overly restricted, so we decided to go for a more time consuming approach. After studying multiple references, we created photo-uploading services that have high reusability. This is one of the most challenging parts in our project.

2. Encryption on phone

Modern CPUs on phones provide hardware acceleration on encryption and decryption. However, slow phones still give very poor performance on the process. The initial implementation of the crypto class generates RSA keys in 30 seconds. The application will hang there for a very long time and being extremely unresponsive. The AES encryption will also make random crashes on our older phone. After tweaking the encryption process for several times, we are able to achieve acceptable performance on the slowest phone without compromising strength of security.

We are also encountering bugs in encryption and decryption process on the crypto library we used. The library randomly crashes while decrypting photos sent from another phone. It is hard to find out the problem until we solved it in a random occasion. This is one of the unfavorable results of not using more reliable and proved library such as Bouncy Castle. However, if the crypto library we see more and more bugs in the crypto library we are using, we may consider changing it.
3. Immature SDK

We encountered problems when using the Windows Phone SDK to develop applications. Our applications are developed using Silverlight but there are many references cannot be used in developing .Net application due to immature SDK. For example there no library supports the conversion between the images and bytes, which is used to be found in .NET applications, so we need to implement this function by our own. The implementation really wastes our time. Microsoft should add back some commonly used methods to the library to improve the experience in developing Windows Phone application.

Furthermore, there are issues in debugging and using emulators. The windows phone emulator does not reflect the true performance of the phone. It is running according to the configuration of your computer, which is not reasonable at all. The emulator also does not support multitouch, which is a popular feature on new applications. Debugging the application on Windows Phone is also troublesome, as we cannot open the camera while it is connected to the computer. That means we either take pictures without debugging, or debugging without taking pictures. This is not desirable for applications that have to make use of the camera. We have devoted more time on debugging because of the flaw of the SDK.
Further Development

The system developed in our project has lots of spaces for improvement. First of all, anonymous photo selling scheme can allow users to upload photos anonymously but also preserve the opportunity to make money out of it. Next, map navigation function can help users to navigate to the site of event in our application. Furthermore, an introduction of blacklisting system can block malicious users uploading fake photos or using photos inappropriately. Last but not least, as a spinoff of the project, the ring signature scheme can be applied on the application polling systems so that we can verify a valid user but also protect the identity of the user.

1. Anonymous Photo Selling

The current ring signature scheme supports anonymous upload, but only in an unencrypted way. With little modification in the ring signature scheme, a user should be able to sell a ring signed photo on our system.

Here we provide a sketch of flow in the modified scheme. First of all, when the photo is encrypted, an AES key is generated. At the same time, a new RSA key pair is generated and the public key is sent to the transaction record. We do ring signature on the AES key, as opposed to the hash value of the photo in the original scheme. When a buyer buys the photo, the AES key is sent to the buyer using RSA technology. The buyer can verify the ring signature on the AES key obtained. Consequently, the buyer upload the e-cash encrypted with the new public key on the transaction record. Only the photo seller with the corresponding private key can decrypt the e-cash. Since it is hard to trace the e-cash, anonymity is preserved.

There are few issues remain unsolved in the proposed scheme. First of all, the newly generated RSA key pair is not from trusted authority. There might be security issues aroused by the unsecure RSA key pair. Furthermore, there might be an ethical issue aroused in the scheme, as users can upload forbidden photos in order to make money.
2. News Agencies Subscription

We believe that it is the news agencies that are most interested in citizen journalism platforms. Therefore, one possible improvement in our system is to allow news agencies to subscribe to our system.

The subscribed users can have a dedicated button in the page when photo sellers are choosing the tags and comments. The sellers can choose to notify the subscribed users about the photo uploaded. The subscribed users are then immediately notified of the event and decide to buy the photo or not. This is a win-win situation for the photo sellers and subscribed users as the sellers can increase the probability of selling the photo, and the subscribers can get the information of the event as soon as possible.

3. Map Navigation

The system we have implemented can show the users events happening around them. Some of the users interested in the event may like to go directly to the location to witness the event. To improve user experience, the system should guide the user from the current location to the site of event.

The map navigation can be made possible by integrating navigation services from Google Map or Bing Map.

4. Blacklisting and Whitelisting

We have implemented the “like“ and “dislike“ function for photos in the system and counting the number of views of the photo. However, there should be a further development in the system in order to rate a photo according to popularity of photos. If a photo is of high popularity, the user should be awarded, and if a user is uploading too many inappropriate photos, we should punish the user.

We can implement a whitelisting scheme, which allows users of high rating to have priority in photo ranking, so that the photos taken by them are more easily seen by others.
We can also implement a blacklisting scheme, which bans those users uploading many inappropriate photos. Since we registers users with the Device ID of the phone and the Live ID account, banning the user means the user have to buy a new phone and use a new Live ID in order to join the platform again.

5. Anonymous Polling using Ring Signature

Recently, the Public Opinion Programme of the University of Hong Kong has hosted an online poll on Chief Executive Election. However, to poll using the system, the user has to provide their Hong Kong ID number in order to identify themselves as a valid voter. Some people are concerned that the polling system may link their poll with their Hong Kong ID number and discloses their vote and identity.

As a spinoff of our project, we suggest that the ring signature scheme can solve the aforementioned problem. The system can use ring signature to sign on the vote, so that we can make sure the voter is valid without disclosing their identity.

There are still problems like double voting if we implement ring signature on the voting system, but this is outside the scope of this project.
Conclusion

Recalling the 4 main goals of our project, which are: (1) A two-way process selling photos with encryption and digital signature; (2) ring signature for uploading photos anonymously; (3) a map to let people know the news around them; (4) categorization of photos to improve organization; we have created a photo sharing platform that have met the goals.

The photo selling process has been speeded up so as to encourage people to use our application in everyday life. We help to protect the interests of both the photo buyers and sellers by photo encryption and digital signatures so that the users can trust our platform to do photo transaction.

The ring signature scheme in our project can help to protect the identity of the photo sellers, so that they can share photos freely on our platform.

We have incorporated GPS technologies and map services so that people can quickly find news happening around them.

Last but not least, through categorization of photos, we can allow users to quickly choose the photos according to their own preferences.

Although there still many bugs and performance issues in the photo sharing platform, especially in the newly introduced ring signature scheme, we can foresee that there are huge potentials in further development in the platform.

All in all, we are confident that our application can help solving the existing problem photo sharing mechanism, as well as promoting citizen journalism.