Secured Access Control System on Cloud Disk Based on Cryptography

FINAL REPORT

Student	:	Chow Chi Ling (3035072832)
Supervisor	:	Dr. Lucas Hui
Course	:	COMP4801 Final Year Project
Project Code	:	FYP16007
Date	:	April 16, 2017

ACKNOWLEDGEMENT

I would like to express a special thanks to the project supervisor, Dr. Lucas Hui, who has provided me guidance and feedback throughout the project.

Second, I would like to extend my thanks to Mr. Rocky Zhang, who has provided many assistance to me.

CONTENTS

1 INTRODUCTION

Since the introduction of cloud platform, popularity of cloud usage has been rising among enterprises. As corporate data will be stored outside company enclosed environment, security, confidentiality and privacy become the major concern. It is often the situation that complex access requirements are demanded from the corporation. Hence, traditional public key cryptography may not be a suitable solution. Nevertheless, due to the flexible nature of attribute based encryption (ABE), this relatively younger cryptography provides a better alternative.

In this project, we explored the opportunity of applying ABE in the context of cloud based electronic health records (EHR) system with the following three objectives: (1) to protect EHR system by ciphertext-policy ABE (CP-ABE); (2) to extend the operations to mobile devices; and (3) to develop convenient access policy management for CP-ABE.

The project was carried out based on CaboLabs EHRServer, which is an open source EHR system complying with an international standard called openEHR. We modified the original EHRServer and database structure for incorporation of CP-ABE. Second, we also developed an Android Application to interact with the users. Third, we implemented CP-ABE related functions in two python web application with the help of Django framework and the open source Charm-Crypto Library. One application is responsible for key generation while the other is responsible for encryption and decryption. Together, all these components collaborate to provide an ABE Enabled EHR System.

All the objectives are achieved on schedule, yet the system still contains some limitations. Therefore, we plan to further improve in three major areas. Firstly, we would like to extend the support of document type in the Android Application. Secondly, we would like to enhance security of the system. Thirdly, we would like to adopt CryptDB's idea to provide encrypted EHR query.

2 BACKGROUND

2.1 CURRENT SITUATION

Usage of cloud computing has been growing rapidly recently. Statistics have shown that there is a steady increase in adoption of cloud solutions for enterprises, from 57% respondents in 2012 to 72% of respondents in 2015. Despite the rising popularity, security remains one of the top challenges, such as tackling unauthorized access and data protection issues. [2]

In the enterprise situation, files and data are usually encrypted and are shared and accessible by certain groups of staffs only. This could be securely encrypted by conventional cryptography, such as the public key infrastructure. However, this could require much computation power. If there are large number of staffs sharing the same piece of information, the number of encryption computation could be very huge. As cloud is also often used together with mobile devices, with limited processing power and storage space in mobile devices, cryptographic protection is even more challenging.

Under this situation, attribute based encryption (ABE), a relatively younger cryptography, could possibly be a better alternative. Encryption and decryption are calculated based on attributes of individuals and a specified access control policy. Therefore, people with different secret keys would be able to decrypt the same ciphertext if the supplied attributes match the specified access control policy. This could allow more flexibility in information sharing while protecting from unauthorized access. Moreover, more recent researches have proposed outsourcing ABE which breaks the decryption process into stages, thereby reducing the computation power requirements.

2.2 THEORETICAL BACKGROUND

2.2.1 Types of Attribute Based Encryption

Attribute based encryption (ABE) was introduced by Sahai and Waters in 2005. The two major components in ABE are attributes and access control policy. Attribute is the characteristic of a piece of information or a user. A piece of information or a user may be identified by a set of attributes. On the other hand, access control policy is the specification of the set of attributes that users must satisfy in order to decrypt the ciphertext. It is organized in tree based data structure. [1]

ABE can be classified into two types, namely ciphertext-policy attribute based encryption (CP-ABE) and key-policy attribute based encryption (KP-ABE).

In CP-ABE, each user is associated with a set of attributes and is represented in their secret key. During the encryption, ciphertext is associated with the access control policy specified by the sender. All users possessing a secret key with attributes satisfying the access control policy specified in the ciphertext would be able to decrypt and get the message.

On the contrary, in KP-ABE, user's secret key is associated with the access control policy while the ciphertext is associated with a set of attributes specified by the sender. Users would be able to decrypt a ciphertext if the ciphertext contains attributes that satisfy the access control policy specified in his secret key.

2.2.2 Implementation

Since 2005, there are various related works in ABE. Implementation methods are slightly different, but they comprise of the four basic steps described below. [3],[4],[5]

Setup

In this step, both CP-ABE and KP-ABE generate a public key pk and a master secret key mk. The function is denoted by $Setup \rightarrow (pk, mk)$.

Key Generation

In CP-ABE, the function is denoted by $KeyGen\ (w, mk) \rightarrow sk_w$. A set of attributes w and the master secret key mk generated in setup are taken as input. Based on the inputs, an individual secret key sk_w is generated.

In KP-ABE, the function is denoted by $KeyGen\ (\tau, mk) \rightarrow sk_\tau$. An access policy τ and the master secret key mk generated in setup are taken as input. Based on the inputs, an individual secret key sk_τ is generated.

Encryption

In CP-ABE, the function is denoted by $Encrypt\ (pk, m, \tau) \rightarrow C_\tau$. The public key pk generated in setup, a message m and an access policy τ are taken as input. Message is encrypted by the sender from the public key and an access control policy to give the ciphertext C_τ.

In KP-ABE, the function is denoted by $Encrypt\ (pk, m, w) \rightarrow C_w$. The public key pk generated in setup, a message m and a set of attributes w are taken as input. Message is encrypted by the sender from the public key and a set of attributes to give the ciphertext C_w.

Decryption

In CP-ABE, the function is denoted by $Decrypt\ (C_\tau, sk_w) \rightarrow m$. It takes a ciphertext C_τ and a secret key sk_w as input. Ciphertext is decrypted by the user who possesses a valid secret key to give the original message m.

In KP-ABE, the function is denoted by $Decrypt\ (C_w, sk_\tau) \rightarrow m$. It takes a ciphertext C_w and a secret key sk_τ as input. Ciphertext is decrypted by the user who possesses a valid secret key to give the original message m.

2.3 LITERATURE REVIEW

Since the introduction of ABE by Sahai and Waters in 2005, researchers gained interest in this cryptographic scheme and further research on it. Due to its security and flexibility, many researchers have proposed the use of ABE under cloud environment. Furthermore, the application of ABE in electronic health records (EHR) system is especially popular.

Below are some examples. Suhair et al. [6] utilized CP-ABE in their proposed cloud-based EHR system. In Shaikh et al.'s system [10], they also utilized ABE in their system. Different with Suhair et al.'s system, they classified data into several sensitivity levels, in which data belonging to different levels are encrypted by different cryptographic methods. It is also observed that some researchers adopted ABE together with other symmetric key encryptions. Narayan et al. [7] and Banescu et al. [3] proposed using ABE with Advanced Encryption Standard (AES). In both system, data are encrypted by AES first and the symmetric key and other meta data are further encrypted by ABE.

There are also systems with two separate domains to provide better security. One for public and the other for personal. Li et al. [9] proposed system with KP-ABE and Multi-Authority ABE (MA-ABE). Their system not only protects data, but also support user revocation in efficient manner. In Rezaeibagha et al.'s approach [8], data are protected by CP-ABE with different access policy in the two domains. Proxy re-encryption are adopted when data are transferred from one domain to the other.

Nevertheless, it is observed that the above-mentioned works focus only on the application of encryption technique to secure the system and hence their EHR data format is in rather elementary format.

3 OBJECTIVE

The project would focus on the implementation of ciphertext-policy attribute based encryption (CP-ABE) in the context of electronic health record system. The project aims at achieving the following objectives:

1. To protect electronic health records by CP-ABE
2. To support operations on mobile devices
3. To develop convenient management of access control policy and attributes for CP-ABE

4 PROJECT SCHEDULE

The project is organized into three phases with different objectives. In Phase 1, we focus on project analysis. In Phase 2, we focus on objective 2 and 3 by making necessary changes on existing system to prepare for incorporation of CP-ABE and developing Android Application. In Phase 3, we focus on objective 1, which is the incorporation of CP-ABE into the system. The timeline is illustrated in the following table.

Date	**Milestones**	**Deliverables**
2 October 2016	Completion of Phase 1	Detailed Project Plan Project Web Page
9-13 January 2017	First Presentation	
22 January 2017	Completion of Phase 2	Preliminary Implementation Detailed Interim Report
16 April 2017	Completion of Phase 3	Finalized Tested Implementation Final Report
18-21 April 2017	Final Presentation	

5 METHODOLOGY

The project is carried out based on an open source electronic health record system CaboLabs EHRServer version 0.7. It is a web based service-oriented system complying with the internationally accepted openEHR standard. Our work would modify CaboLabs EHRServer while following openEHR standard at the same time. Besides, an Android Application would be developed as a client application interface. To incorporate CP-ABE into the system, the open source Charm-Crypto Library would be used together with Django Web Framework.

5.1 OPENEHR FRAMEWORK

The openEHR framework is an international open standard, governing systems related to electronic health records. The standard is developed and continuously maintained by a large group of professionals. Many organizations, including commercial and research projects, are conforming to the standard and hence it is internationally accepted. The aims of the standard are as follows. [11]

1. To design better software architecture for maintainability
2. To unify data format for system scalability and interoperability

Two-level modelling is suggested in openEHR framework. In this modelling method, the structure is governed by reference model while the content is governed by archetype model. In terms of access control, there are no concrete requirements, but some guidelines summarized as below.

1. Patient consent is needed whenever his/her information is shared.
2. Different parts of EHR should be allowed to access with different rights.
3. Learning curve must not be steep for users.
4. Access control is needed for modification of access control policy of EHR.
5. Access must be declined except it is defined explicitly.

5.2 CABOLABS EHRSERVER

CaboLabs EHRServer is an open source project conforming to the openEHR standard. It is developed using Grails framework, a web framework for Groovy. Data are stored in XML files in the project directory as well as in MySQL database. [12]

There are two parts in version 0.7, namely an Administrative Web Platform for direct graphical access from users and REST API for other application interfaces to access the data. The following major functions are provided:

1. Register and login as users
2. Create and modify other users
3. Create and modify patients and their electronic health records
4. Create and modify documents of electronic health records
5. Query data in electronic health records

However, it is observed that the implementation of EHRServer imposed certain restrictions on EHR sharing. Access control of EHR depends on the organizations the user belongs to. Hence, users are only able to access EHRs under the same organization. On the contrary, access control cannot be defined in more fine-grained ways so as to allow part of the people in the same organization to access EHR. In reality, there may be situations that EHR has to be shared to other medical organization for follow up, or different departments of the same organization have different access rights to an EHR. More importantly, data are not encrypted at all. Hence, any server compromises would cause data leakage and privacy issues. There are also risks rising from curious server administrator or database administrator. In summary, EHRServer provides a suitable base for our project to focus on enhancing security of such system without devoting excessive efforts in the semantic context.

5.3 CHARM-CRYPTO LIBRARY

Charm-Crypto Library is an open source python library with implementation of several encryption schemes including both CP-ABE and KP-ABE. Three versions of ABE schemes were implemented in the library. The implementation following scheme proposed by Bethencourt and Waters in 2007 is adopted in this project. [13] In addition to CP-ABE, the AES implementation in this library is also adopted in the project.

5.4 DJANGO WEB FRAMEWORK

Django Web Framework is a web application framework based on python. It provides easy and convenient development process for a web application. [15] The framework is adopted in the project as a façade to wrap the Charm-Crypto Library, thereby allowing connection to be made from CaboLabs EHRServer.

6 IMPLEMENTATION DETAILS

6.1 SYSTEM DESIGN

6.1.1 System Security

The system makes use of both AES and CP-ABE together to protect the EHR documents and achieve access control. EHR documents are protected by AES with a symmetric key. The symmetric key is further encrypted by CP-ABE to achieve access control. For attributes used in CP-ABE, we identify user by the unique identifier of the organizations and departments they belong to. For access policy of CP-ABE, the system accepts operators “and” and “or”. For the access control of access policy modification, it is assumed that only the user who create the access policy has the rights to update it. For demonstration simplicity, keys are currently stored within the EHRServer.

6.1.2 Component Diagram

The system consists of six interacting components. Among all the components, EHRServer is the major component connecting the other five components. It provides the administrative web interface for users to directly interact with, as well as a REST API for the Android Application to interact with. Secondly, there are XML documents for storage of EHR documents. Thirdly, there is a MySQL database storing other information and data. Fourthly, there is an Android Application for users to upload and retrieve EHR documents through mobile devices. Finally, there are two components for security purpose, namely the Trusted Authority and the ABE Server. The Trusted Authority is responsible for key generation, while the ABE Server is responsible for encryption and decryption using CP-ABE. Detailed development of each components will be further explained in later sections.

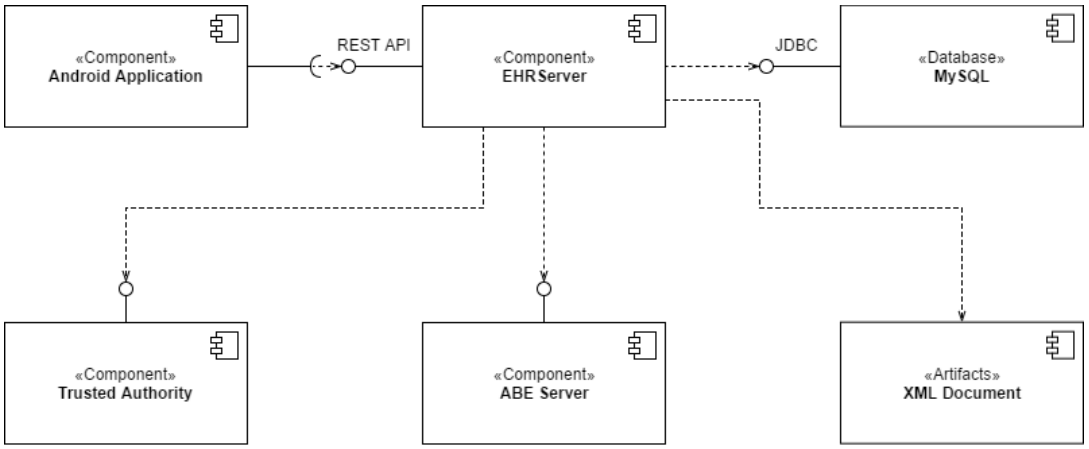

Figure 1 Component Diagram

6.1.3 Sequence Diagram

This section describes the interaction between different system components. Since not all functions from the original EHRServer are directly related to the project, only the eight major flows are explained below. They are (1) system set up; (2) user create; (3) user update; (4) EHR create; (5) document create; (6) document retrieval; (7) document update; and (8) EHR sharing.

System Set Up

The function is performed when EHRServer is initially set up on server. EHRServer requests a public key from the Trusted Authority for usage in other functions including key generation and ABE encryption.

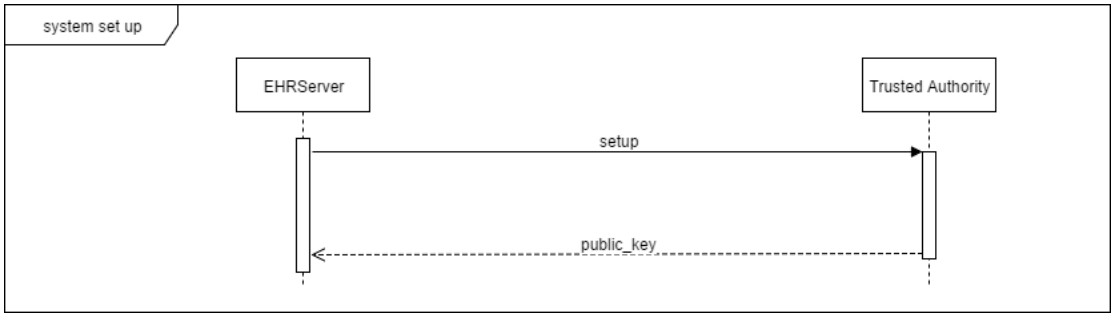

Figure 2 Sequence Diagram (System Set Up)

Create User

The function is triggered when user registers for an account in EHRServer. Upon successful registration, EHRServer computes the new user's attributes and then requests for a secret key from the Trusted Authority for that user. The returned secret key will be stored within EHRServer.

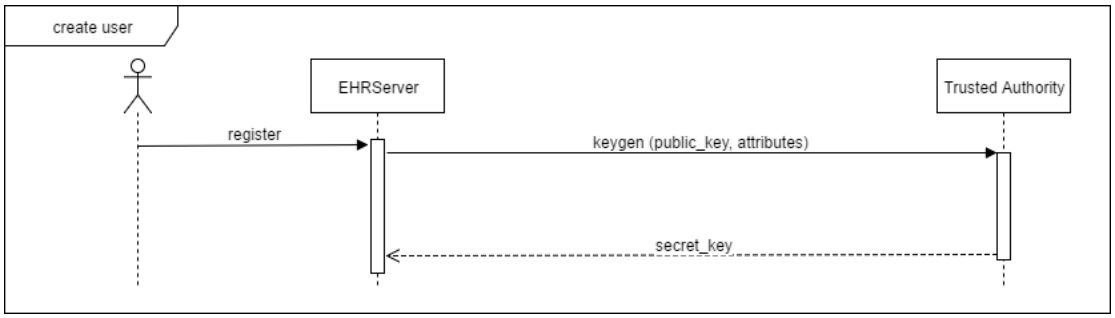

Figure 3 Sequence Diagram (Create User)

Update User

The function is triggered when user updates his information in EHRServer. Upon successful update, EHRServer computes the user's attributes again and then requests for a new secret key from the Trusted Authority for that user. The old secret key will be replaced by the newly returned secret key.

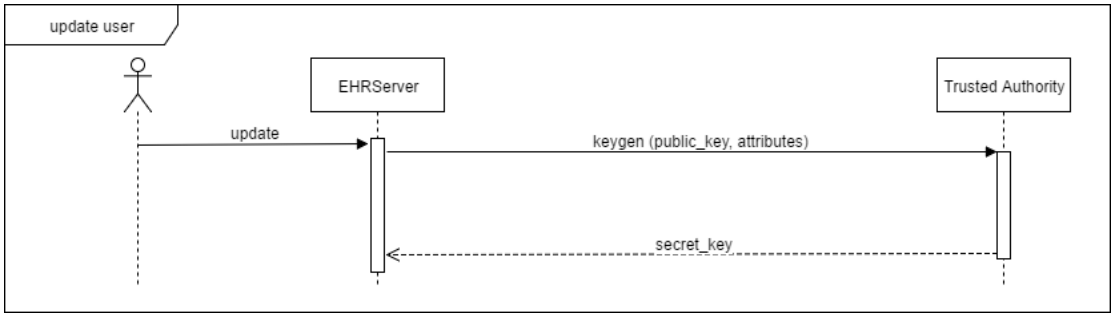

Figure 4 Sequence Diagram (Update User)

Create EHR

The function is triggered when user creates an EHR for a patient. The EHRServer first creates an EHR for the patient and then generates a default access policy for it, which requires other users to be in the same organization and same department with the patient.

Figure 5 Sequence Diagram (Create EHR)

Create Document

The function is triggered when user creates a medical document through the Android Application. Upon successful committing, EHRServer triggers ABE Server to encrypt the document. The returned ciphertext and symmetric key are stored in EHRServer.

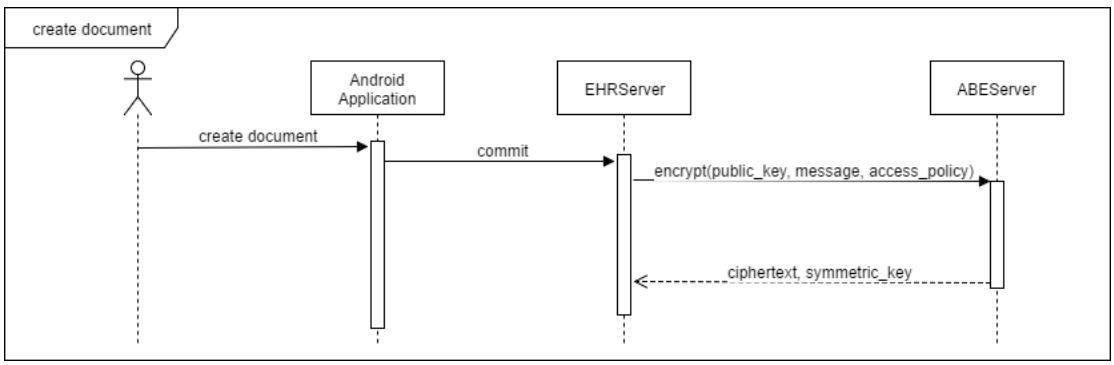

Figure 6 Sequence Diagram (Create Document)

Retrieve Document

The function is triggered when user retrieves a medical document through the Android Application or directly through the Administrative Web Platform of the EHRServer. After receiving the retrieval request, the EHRServer triggers ABE Server to decrypt the document. The user will be able to read the decrypted document if decryption is performed successfully.

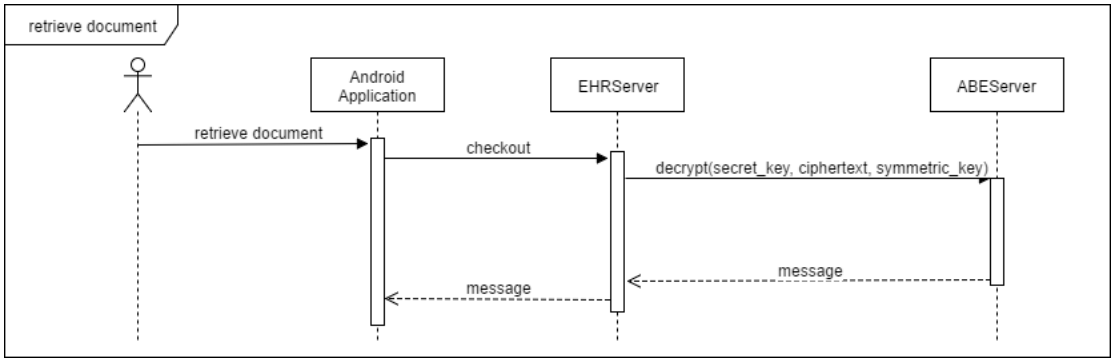

Figure 7 Sequence Diagram (Retrieve Document through Android Application)

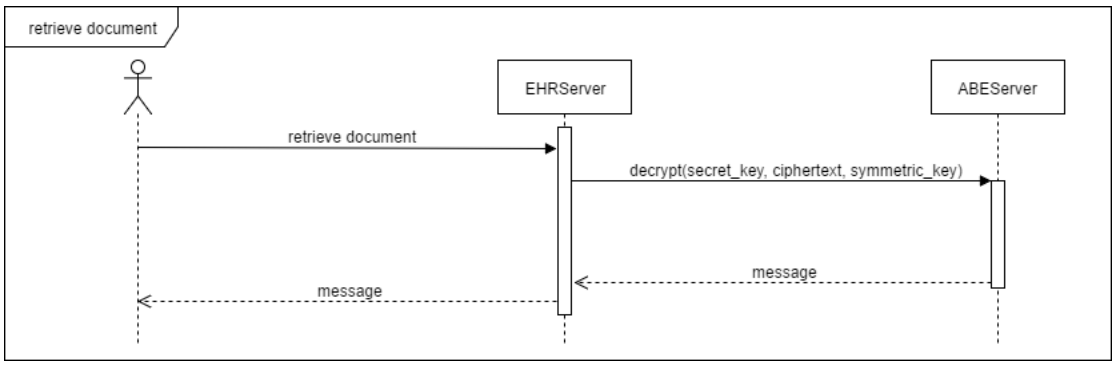

Figure 8 Sequence Diagram (Retrieve Document through Administrative Web Portal)

Update Document

The function is triggered when user updates a medical document through the Android Application. Upon successful committing, EHRServer triggers ABE Server to encrypt the document. The returned ciphertext and symmetric key are stored in EHRServer.

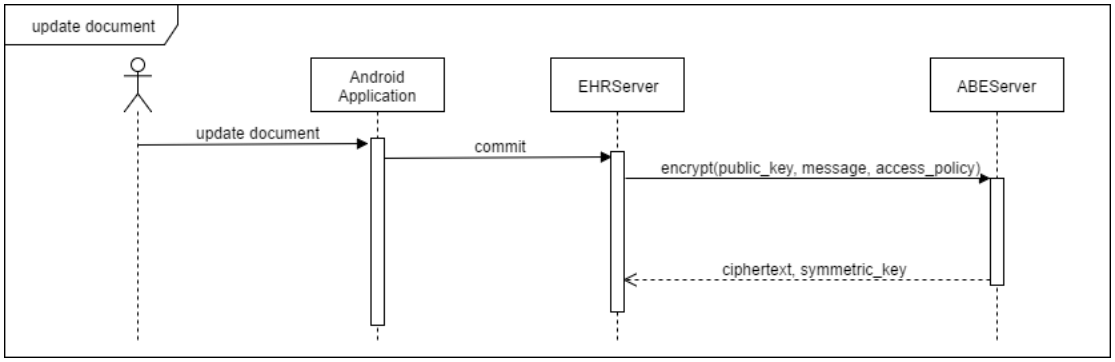

Figure 9 Sequence Diagram (Update Document)

Share EHR

The function is triggered when user shares the EHR to other users. The user first updates the access policy of the EHR. Upon successful update, EHRServer triggers ABE Server decrypt all related documents and re-encrypt them with the new access policy.

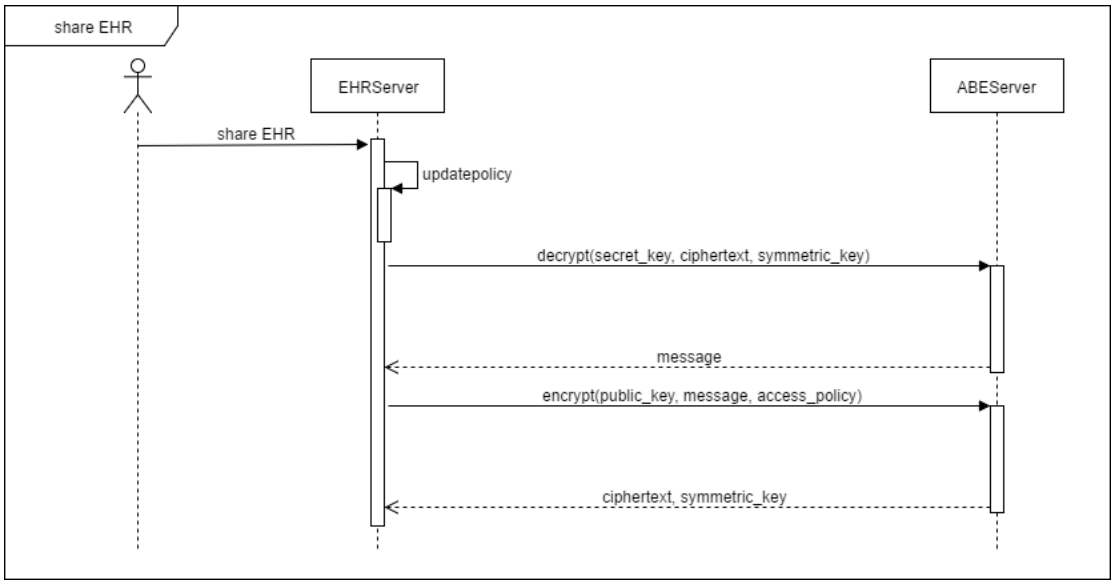

Figure 10 Sequence Diagram (Share EHR)

6.2 MODIFICATION TO CABOLABS EHRSERVER

6.2.1 Administrative Web Platform

Modification was carried out in two stages. In the first stage, two modules, including department module and access policy module, were added to prepare for the incorporation of CP-ABE. In the second stage, some existing functions were modified to connect with ABE Server and Trusted Authority.

New Department Module

Firstly, functions related to departments were added. In the original implementation, users belong to different organizations. In the actual situation, however, users are also divided into different departments. More importantly, different departments may have different access rights to a patient's EHR depending on situations. Therefore, the department module was implemented with creation, retrieval, and update functions.

After implementation, each user can be associated with the departments they are working for. Each patient should also belong to one of the department. Figure 11 – 13 shows the user interfaces relating to these functions.

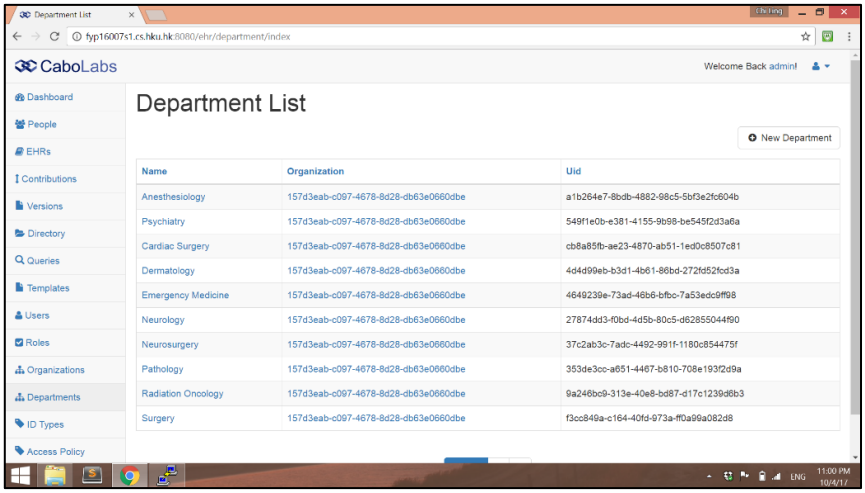

Figure 11 Administrative Web Platform User Interface (List Departments)

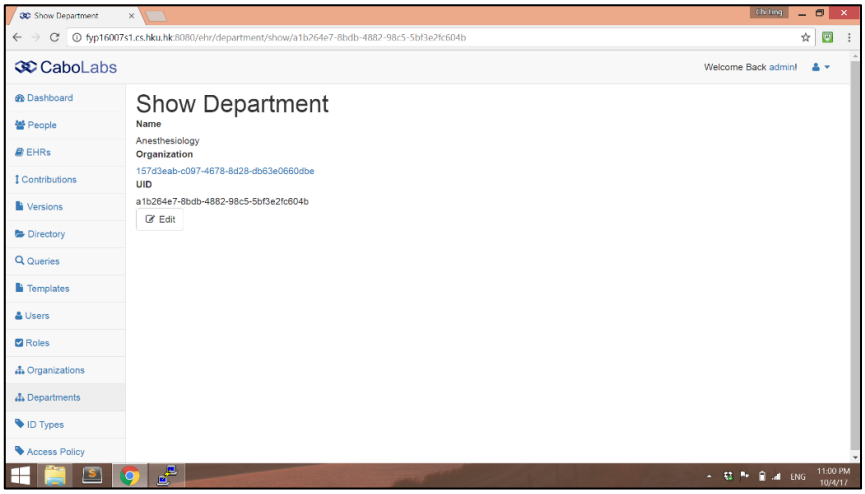

Figure 12 Administrative Web Platform User Interface (Show Department)

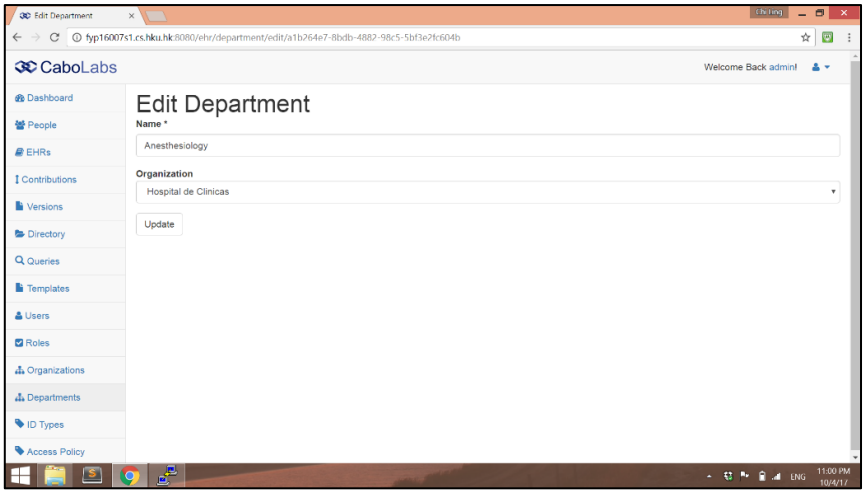

Figure 13 Administrative Web Platform User Interface (Create/Edit Department)

New Access Policy Module

Secondly, functions related to access policies were added. To allow convenient management of access control policy, creation, retrieval, and update functions of access policies were implemented. Create and edit pages were implemented to be flexible such that a more complicated access policy can also be supported. For example, the policy "(EASTERN_HOSPITAL OR WESTERN_HOSPITAL) AND (DEPARTMENT_OF_PHYSIOTHERAPY OR DEPARTMENT_OF_CARDIOLOGY)" involving multiple levels in tree structure is also supported.

Besides, retrieve and modification of access policy are guarded by access control. It is assumed that only the user who create the access policy is eligible to read and make amendments. The eligible user can modify the access policy of a specific EHR whenever it is needed. Figure 14 – 16 shows the user interfaces relating to these functions.

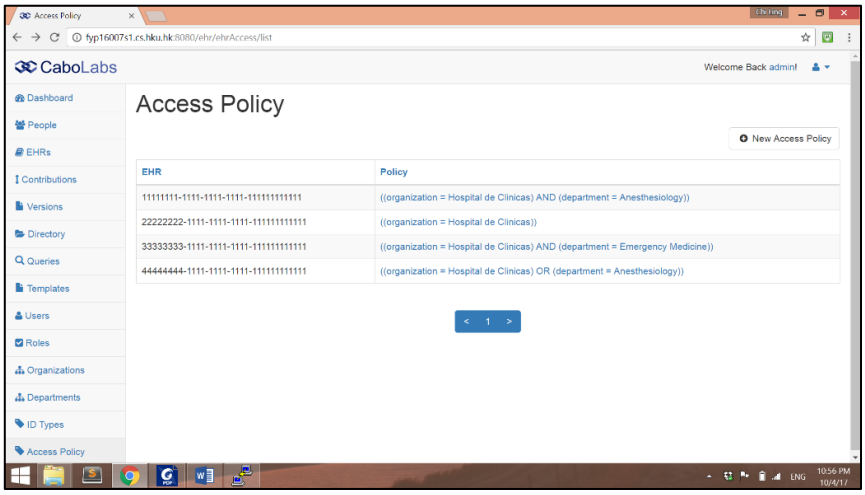

Figure 14 Administrative Web Platform User Interface (List Access Policy)

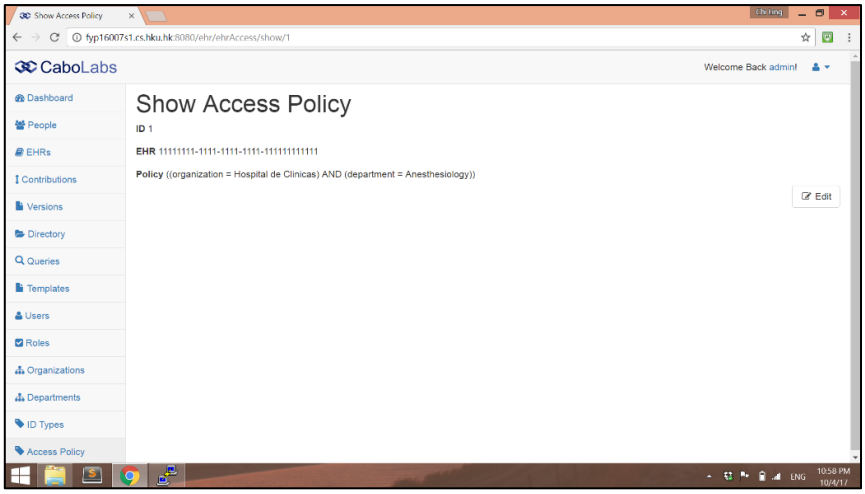

Figure 15 Administrative Web Platform User Interface (Show Access Policy)

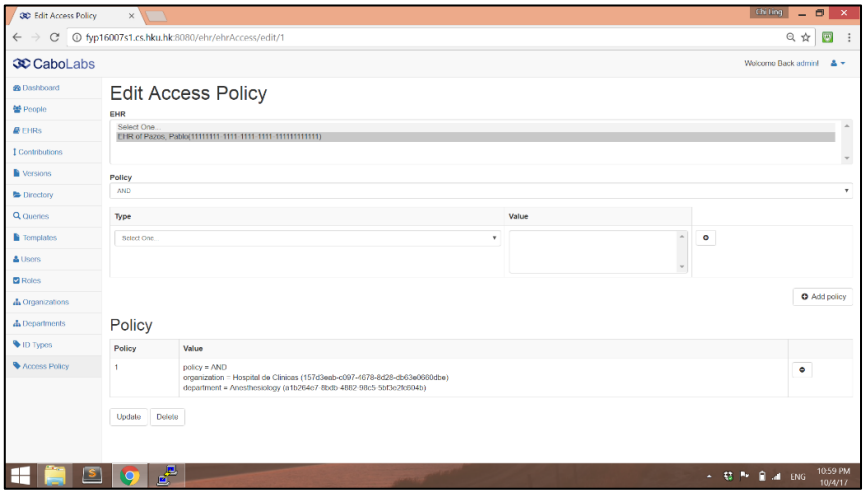

Figure 16 Administrative Web Platform User Interface (Create/Edit Access Policy)

Modification to Existing Functions

In the second stage, some existing functions were modified to connect with ABE Server and Trusted Authority. Firstly, functions such as creating user and updating user attributes were modified to trigger Trusted Authority to generate secret key for the user and store the secret key. Secondly, functions relating to EHR documents were modified to trigger ABE Server's encryption and decryption. In the process, there are no temporary file generated. As a result, the server only stores encrypted EHR documents. Thirdly, access control of EHR documents were extended to work according to the access policy instead of the organization the user belongs to.

6.2.2 REST API

REST API were modified accordingly with the incorporation of CP-ABE and the development of Android Application.

Firstly, functions related to creating, retrieving or updating EHR documents were modified to trigger ABE Server's encryption and decryption, thereby allowing users with Android Application to read and write EHR documents as usual.

Secondly, although most of the functions in REST API has been implemented with both XML and JSON response, it is found that a few can only support XML response. Hence, two functions were modified to support JSON response as well.

Thirdly, a new function was created to retrieve EHR access policy by the unique identifier of the EHR. The function can be triggered by a HTTP GET request.

GET /ehr/$ehrUid/ehrAccess

Parameters:

"format": specify the output format, valid values are "xml" or "json"

Result sample: (Content-Type: application/json)

```
{
    "policy" : "((department-a1b264e7-8bdb-4882-98c5-5bf3e2fc604b) or (department-
                549f1e0b-e381-4155-9b98-be545f2d3a6a))"
}
```

6.3 MODIFICATION TO DATABASE

In total, four new tables were created to support the modification mentioned in the above section.

A table “department” was created to store the information of the departments. One user can work for more than one departments. Hence, a table “user_department” is created to store the 1:M relationship of users and departments.

department

“uid”	:	is the unique identifier of the department.
“version”	:	records the number of edition to the record.
“name”	:	is the name of the department.
“organization_uid”	:	is the foreign key to identify the organization the department belongs to.

user_department

“departments_idx”	:	is the unique identifier of a record.
“user_departments_id”	:	is the foreign key to identify the user.
“department_id”	:	is the foreign key to identify the department.

Two new tables were created to store the access policy of EHR. Since a CP-ABE access policy is in tree structure, the table “access_policy” was created to store this information. The whole access policy is stored separately in more than one records. The table “ehr_access” was created to point an EHR to the root of its access policy in the table “access_policy”.

ehr_access

“uid”	:	is the unique identifier of an EHR access.
“version”	:	records the number of edition to the record.
“scheme”	:	is the security scheme used.
“ehr_uid”	:	is the foreign key to identify the corresponding EHR.
“settings_id”	:	is the foreign key to identify the access policy.

access_policy

“uid”	:	is the unique identifier of an access policy.
“version”	:	records the number of edition to the record.
“type”	:	defines the meaning of the value column. It can be organization, department or criteria.
“value”	:	is the value corresponding to the type defined. When type is organization or department, the value is the unique identifier of that organization or department. When type is criteria, the value is either “and” or “or”.
“parent_id”	:	is the foreign key to identify its parent access policy. A null value means it is the root of the whole access policy tree.

6.4 Development of Android Application

The Android Application acts as a client application interface for users. It consists of three major functions.

Firstly, create a document. There are many cases that new clinical documents are created to record patients' situations or medical treatments they have received. Since the focus of the project is CP-ABE, the type of clinical documents supported is limited to nursing observations only for simplicity. After users complete all required inputs and click submit, the application will send a HTTP POST request to the REST API. The remaining process would be handled by REST API. Upon successful submission, the user will be redirected back to the previous page.

Secondly, retrieve a document. In diagnosis, it is essential to retrieve previously created clinical documents to learn about the patients or monitor their situations. When users click on a patient's EHR, a HTTP GET request will be sent to the REST API to obtain the full list of clinical documents. If the user does not have enough access rights to access the patient's EHR, an error message will be displayed and he will be redirected back to the previous page. If the user has enough access rights, he can read the full list of clinical documents. By clicking on one of the record, the application will send a HTTP GET request to the REST API to get the specific document and display on screen.

Thirdly, modify a document. There could be situations that some previously inputted data are found to be incorrect and thus correction is needed. After user retrieves a specific document, he can click modify button and correct any mistakes. After submission, the application will send a HTTP POST request to the REST API. The remaining process would be handled by REST API. Upon successful submission, the user will be redirected back to the previous page.

Wireframe

The following is the wireframe of the Android Application.

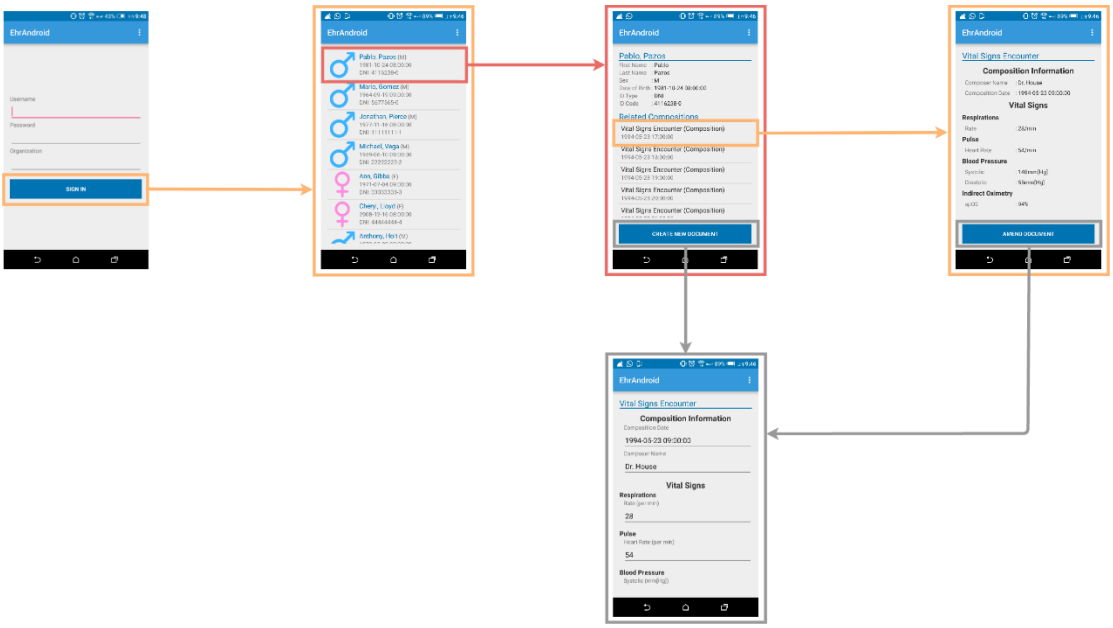

Figure 17 Wireframe of Android Application

User Interface

The following are the user interfaces of Android Application.

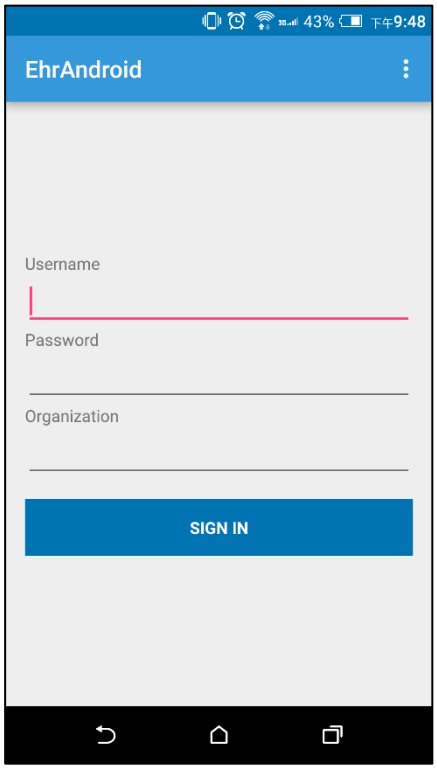

Figure 18 Android Application User Interface (Login)

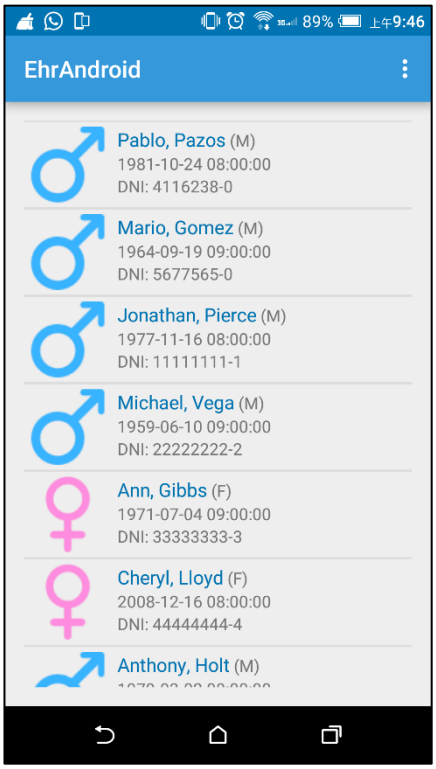

Figure 19 Android Application User Interface (List Patients)

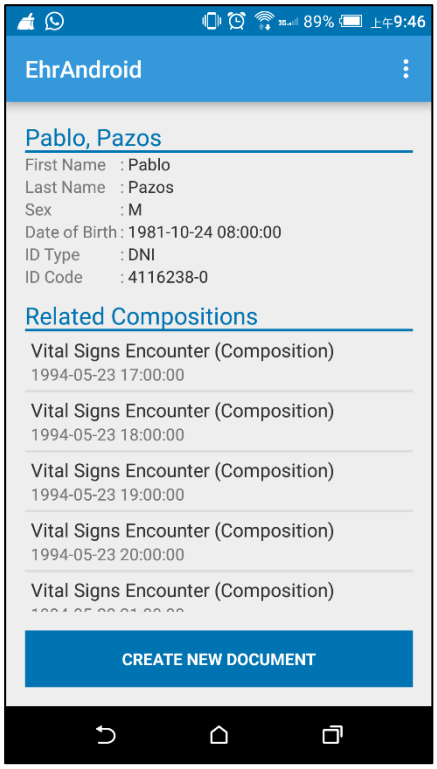

Figure 20 Android Application User Interface (View EHR)

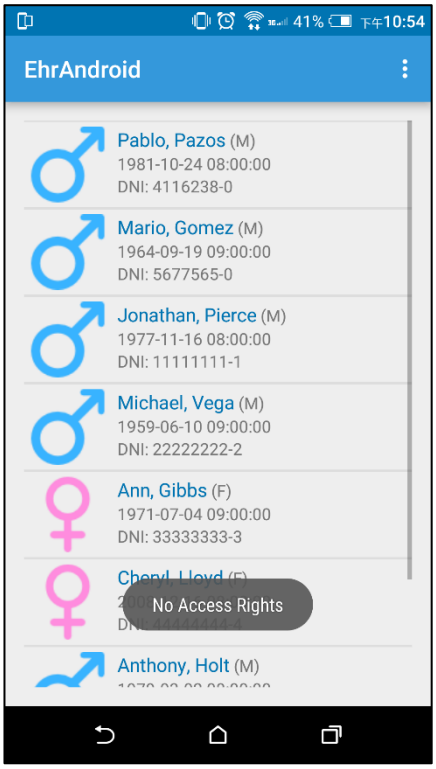

Figure 21 Android Application User Interface (View EHR Failure)

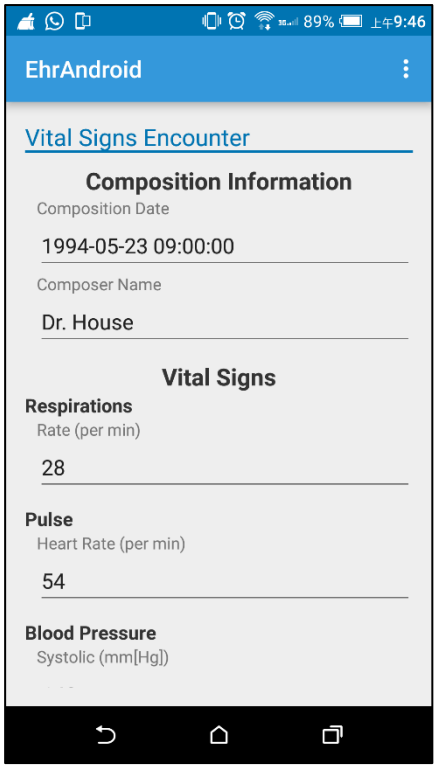

Figure 22 Android Application User Interface (Create/Amend Document)

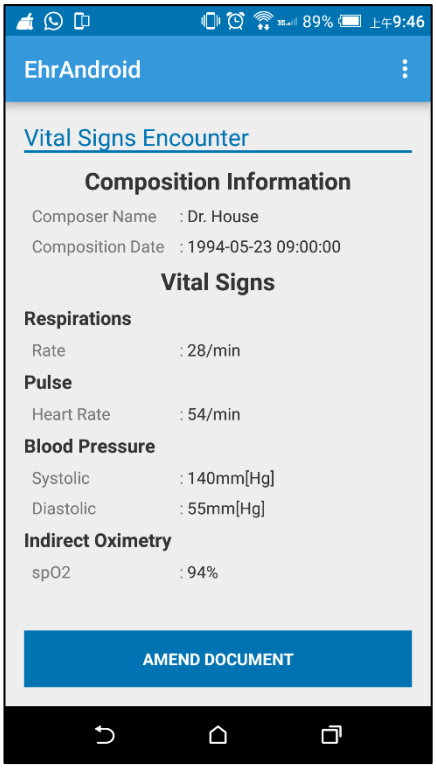

Figure 23 Android Application User Interface (View Document)

6.5 DEVELOPMENT OF TRUSTED AUTHORITY

The Trusted Authority functions as a standalone component for key generation. It consists of two major functions, namely setup and key generation. The Trusted Authority is developed using Django framework. The CP-ABE functions involved are provided by the Charm-Crypto Library. Detailed usage and communication format of the two functions are descripted as follows.

The setup function runs CP-ABE setup algorithm to obtain a master key and a public key. The master key is kept within the Trusted Authority for future key generation use. On the other hand, the public key is returned to the client. The function can be triggered by a HTTP POST request.

POST /setup

Parameters:

None.

Result sample: (Content-Type: application/json)

```
{
    “result” : “1”,
    “public_key” : “PUBLICKEYINBYTES”
}
```

The key generation function takes the public key of the client and a set of attributes as input. It runs CP-ABE key generation algorithm to produce a secret key for the user based on the attributes supplied. The function can be triggered by a HTTP POST request

POST /keygen

Parameters:

“public_key” : public key held by EHRServer
“attributes” : attributes of the user

Result sample: (Content-Type: application/json)

```
{
    “result” : “1”,
    “secret_key” : “SECRETKEYINBYTES”
}
```

6.6 DEVELOPMENT OF ABE SERVER

The ABE Server is responsible for providing ABE and AES encryption and decryption. It consists of two major functions, namely encrypt and decrypt. Same as the Trusted Authority, the ABE Server is developed using Django framework. The ABE and AES functions involved are provided by the Charm-Crypto Library. Detailed communication format of the two functions are descripted as follows.

The encrypt function runs CP-ABE encrypt algorithm. It takes the public key of the client, the message in plaintext and the access policy as input. Then, the function generates a random symmetric key for AES. The symmetric key is used to encrypt the original message. After that, the symmetric key is further encrypted by CP-ABE. Finally, the function returns the AES encrypted message and the CP-ABE encrypted symmetric key to the client. The function can be triggered by a HTTP POST request.

POST /encrypt

Parameters:

"public_key" : public key held by the EHRServer
"message" : message to be encrypted
"access_policy" : access policy for encryption

Result sample: (Content-Type: application/json)

```
{
    "result" :  "1",
    "ciphertext" : "CIPHERTEXT",
    "symmetric_key" : "SYMMETRICKEYINBYTES"
}
```

The decrypt function takes a secret key, a CP-ABE encrypted symmetric key and an AES encrypted message as input. It runs CP-ABE decrypt algorithm to obtain the original symmetric key. Upon successfully decryption, the encrypted message is decrypted by AES using the symmetric key and returns to the client. If decryption is unsuccessful, the function will return error. The function can be triggered by a HTTP POST request.

POST /decrypt

Parameters:

"ciphertext" : ciphertext to be decrypted
"secret_key" : secret key used to decrypt the symmetric key
"symmetric_key" : CP-ABE symmetric key used to decrypt the ciphertext

Result sample: (Content-Type: application/json)

```
{
    "result" :  "1",
    "message" : "MESSAGE"
}
```

7 RESULTS

The project was carried out successfully with the achievement of all three objectives. Firstly, EHRs are protected with the incorporation of CP-ABE. This is achieved with the development of Trusted Authority and the ABE Server. Together, they provide the functionality for key generation, ABE encryption and ABE decryption. Secondly, operations on mobile devices are supported with the development of an Android Application. Users can interact with the system conveniently through their mobile devices. Finally, convenient access policy management for CP-ABE is also supported. This is achieved through the development of a new function on the EHRServer Administrative Web Platform.

8 FUTURE DEVELOPMENT

There are three major areas we would like to improve regarding both functionality and security.

8.1 DOCUMENT TYPE EXTENSION

Currently in the Android Application, only one type of document is supported, which is vital signs encounter. This restrictes the usage of the Android Application. To increase the usability of the system, more document types are to be supported in the Android Application.

8.2 SECURITY ENHANCEMENT

Although the EHR documents in the system are protected by CP-ABE, the system is not yet fully protected. Firstly, other sensitive information that store in the database are still stored in plaintext. These data should be encrypted as well. Moreover, for ease of demonstration, currently the secret keys of every users are stored in the same directory of the server and all the system components are implemented in the same server. The key management should be enhanced and the Trusted Authority should also be separated out so as to reduce the risk of key compromises. Besides, currently only creation and modification of EHR documents are logged down. Other actions should also be logged for audit purpose.

8.3 ENCRYPTED QUERY

After the incorporation of CP-ABE, the original EHR documents query functionality is no longer supported. Hence, we propose to adopt the idea of CryptDB. [14] This will be achieved by encrypting various data depending on search requirements with onions encryption layers. Based on the requirements, deterministic encryption, order preserving encryption, and homomorphic encryption will be adopted so that queries can be performed on ciphertext. After that, the database field will be further encrypted by ABE to achieve the security requirements. Besides, a database proxy will be developed. The database proxy is responsible for connecting EHRServer with MySQL database. When a normal SQL query is issued from the EHRServer, the database proxy will transform the values into required encrypted form and then perform normal SQL on the database. After the database has executed the query, the database proxy will decrypt the result and return to EHRServer.

9 CONCLUSION

To conclude, the project was carried out successfully and on schedule. We have demonstrated the application of Ciphertext-Policy Attribute Based Encryption (CP-ABE) on a cloud-based electronic health record (EHR) system.

We have separated the system into several components: (1) EHRServer Administrative Web Platform is responsible for interacting with users directly; (2) EHRServer REST API is responsible for connecting the Android Application with other parts of the system; (3) Android Application is responsible for providing convenient usage on mobile devices; (4) Trusted Authority is responsible for generating key for CP-ABE usage; and (5) ABEServer is responsible for encrypting and decryption EHR documents by CP-ABE and AES.

It is observed that the system has the following limitations: (1) limited support of document type on Android Application; (2) unprotected sensitive information; and (3) loss of EHR querying functions. As a result, we plan to further work on them to extend the coverage of document type, enhance security and adopt CryptDB's idea to provide EHR querying functions.

10 BIBLIOGRAPHY

[1] "Introduction to Attribute Based Encryption (ABE)," [Online]. Available: http://gleamly.com/article/introduction-attribute-based-encryption-abe.

[2] IDG Enterprise Marketing, "Clound Computing Survey 2015," 17 11 2015. [Online]. Available: http://www.idgenterprise.com/resource/research/2015-cloud-computing-study/.

[3] S. Banescu, H. Ideler and S. Posea, "Personal Health Record System Protected using CP-ABE".

[4] J. Li, X. Huang, J. Li, X. Chen, Y. Xiang and IEEE, "Securely Outsourcing Attribute-Based Encryption with Checkability," *IEEE Transactions on Parallel and Distributed Systems,* vol. 25, no. 8, pp. 2201-2210, 2014.

[5] F. Xhafa, J. Li, G. Zhao, J. Li, X. Chen and D. S. Wong, "Designing Cloud-Based Electronic Health Record System with Attribute-Based Encryption," Springer Science+Business Media New York, New York, 2014.

[6] A. Suhair, R. Stanisaw and K. R. Rajendra, "Designing a secure cloud-based ehr system using ciphertext-policy attribute-based encryption," *Proc. DMC,* pp. 21-25, 2012.

[7] Narayan, Shivaramakrishnan, M. Gagn and P. Bandi, "Securing E-healthcare records on Cloud Using Relevant data classification and Encryption".

[8] Rezaeibagha, Fatemeh and Y. Mu, "Distributed clinical data sharing via dynamic access-control policy transformation," *International journal of medical informatics,* vol. 89, pp. 25-31, 2016.

[9] Li, Ming et al., "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE transactions on parallel and distributed systems,* vol. 24, no. 1, pp. 131-143, 2013.

[10] R. Shaikh, J. Banda and P. Bandi, "Securing E-healthcare records on Cloud Using Relevant data classification and Encryption".

[11] OpenEHR Foundation, "Welcome to openEHR," [Online]. Available: http://www.openehr.org/.

[12] CaboLabs, "CaboLabs Health Informatics, Standards and Interoperability," [Online]. Available: http://cabolabs.com/en/projects.

[13] Johns Hopkins University ISI, "Charm-Crypto Project," [Online]. Available: http://charm-crypto.com/.

[14] R. A. Popa, C. M. S. Redfield, N. Zeldovich and H. Balakrishnan, “CryptDB: processing queries on an encrypted database,” *Communications of the ACM,* vol. 55, no. 9, pp. 103-111, 2012.

[15] Django Software Foundation, “The Web framework for perfectionishs with deadlines | Django,” [Online]. Available: https://www.djangoproject.com/.