Assignment 1
Secure Data Management

Rick van Galen - s0167746 (UT)
Erwin Middelesch - s0197106 (UT)
Jeroen Senden - s0191213 (UT)
Bas Stottelaar - s0199141 (UT)

November 25, 2013

1 Introduction

In this report, we describe our implementation for a Public Health Record (PHR) system. This system is implemented in Python and consists of a client and server. The role of the server is to announce the public parameters and store encrypted data while the client takes care of the encryption and decryption of data.

Our solution is based on a Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme and uses no classic access control to authenticate users. Different categories of data can be read and written by different parties or sub parties.

First, we introduce the data model, including the categories of data and different parties. Then, we introduce the access policies. The third section will discuss the actual implementation and verifies the the requirements. The last section will discuss our implementation.

2 Data model

We split the health record globally in three different segments. These are:

2.1 Data categories

The system defines the following categories of data

Patient data In this segment of the health record, we store personal information of the user.

Health data This is the data inserted by a hospital where the user has been treated.

Training data This section contains training related data stored by the health club.

2.2 Parties

The following parties are defined and assumed to work with the system.

Patient The subject of the electronic health system. Each health record is associated with one patient and should be under full control of the patient. The patient has full read and write access to every category of data.

Doctor A patient's doctor treats the patient with health issues. The doctor may require read access to any part of a patient's health record. Therefore, the patient may choose to give the doctor read access to his personal information, health data and training data.

Insurance Health insurance companies may want to read a patient's records for fee calculations or health verification. A patient may give insurance companies read access to all of the segments of the his/her health record.

Employer Like the insurance company, an employer may need access to an employee's health information and may be granted read access.

Hospital Hospitals are given read and write access to the health data of the health record. An additional restriction is that a hospital may only write to the health record when a patient has been treated at that hospital.

Health club The health club provides the user with additional health services and training programs. Because this information is relevant to the health record, the health club gets read and write access to the training related segment of a patients health record.

3 Access model

Setup: person generates 3 sets of public and master keys.
personal data public key(pk_{pd}) and master key(mk_{pd})
health data public key(pk_{hd}) and master key(mk_{hd})
training data public key(pk_{td}) and master key(mk_{td})

Keygen: For every party the person generates 3 secret keys based on their identity attribute. Thus the doctor, insurance and employer get a secret key based on their attribute for personal data (sk_{pd}), health data(sk_{hd}) and training data(sk_{td}). The hospital receives a secret key based on their attribute for health data (sk_{hd}) and the health club receives a secret key based on their attribute for training-related data (sk_{td})
The person then uploads the existing data encrypted with the access policies he wishes the data to have, together with the corresponding public keys and attribute lists to the server.

3.1 Encrypt

The only parties that are allowed to encrypt data (write-access) are the person to whom the health record belongs, the hospital that the person has been treated at and the healthclub the person is currently a member of.

Person A person can change the data of every field of his health record.

Hospital Reads the encrypted pk_{hd} and access policy from the database, decrypts them using his sk_{hd}. and encrypts the message using the pk_{hd} and access policy.

Health Club Reads the encrypted pk_{td} and access policy from the database, decrypts them using his sk_{td}. and encrypts the message using the pk_{td} and access policy.

3.2 Decrypt

To decrypt, the decrypting party simply uses his secret key. He either gets data that makes sense, or will decrypt the data into gibberish.

4 Implementation

4.1 Database model

4.2 Verification of requirements

The assignment includes the following requirements:

1. A patient can insert personal health data into his *own* record.
2. A patient can provide his doctor, his insurance, and his employer with read access to (parts of) his health record.
3. A hospital can insert patient health data for any patient that has been treated by *that* hospital.
4. A health club can insert training-related health data to any patient record that is a member of *that* club.

To show that our system fulfills the requirements, we show a sample session via the command line interface. One can play this session back and should see similar results.

4.2.1 Creating a PHR

Since the patient is in control of his own PHR, he is responsible for creating one.

```
>>> python manage.py phr_create patient.json http://127.0.0.1:8000 "Patient record"
BEGIN SECRET READ FOR KEYS DOCTOR
(...truncated for clearance...)
END SECRET READ KEYS FOR DOCTOR

BEGIN SECRET READ FOR KEYS HOSPITAL-A
(...truncated for clearance...)
END SECRET READ KEYS FOR HOSPITAL-A

...

BEGIN SECRET READ FOR KEYS EMPLOYER
(...truncated for clearance...)
END SECRET READ KEYS FOR EMPLOYER

BEGIN SECRET READ FOR KEYS INSURANCE
(...truncated for clearance...)
END SECRET READ KEYS FOR INSURANCE
```

After running this command, all the secret reading keys for all the parties are generated. Each key has a bit of meta data encoded to connect to the right PHR. The mapping (see section XXX) defines who will get initial read access to which category of data. The patient should take care of distributing the keys, preferably via an out-of-band channel.

A patient can encrypt data for other parties even if the keys have not yet been distributed.

4.2.2 Writing data

The patient decides to encrypt two messages. The first message can be read by his doctor, the second one by his insurance and his employer. They have access to the personal category of data.

```
>>> python manage.py phr_encrypt patient.json PERSONAL DOCTOR "Hi, Doctor!"
Uploaded to record item ID 55
>>> python manage.py phr_encrypt patient.json PERSONAL INSURANCE,EMPLOYER "Hi, Insurance
    and Employer!"
Uploaded to record item ID 56
```

The identifiers correspond to the items in the PHR of the patient. To verify that the patient has read access, he will decrypt the data he sent to the server.

```
>>> python manage.py phr_decrypt patient.json 55
Record item content:

Hi, Doctor
>>> python manage.py phr_decrypt patient.json 56
Record item content:

Hi, Insurance and Employer
```

4.2.3 Granting write access

By default, no other party has write access. The patient should grant write access for all parties or sub parties.

Let's assume the patient wants to grant Hospital-A write access.

```
>>> python manage.py phr_grant patient.json HEALTH HOSPITAL-A
Granted key for category HEALTH with key ID 22
```

The write key is encrypted for party Hospital-A. If the patient wanted, he could give all hospitals write access by specifying the Hospital party instead of Hospital-A.

4.2.4 Connecting to a PHR

In this step, the keys have been safely distributed to each party. We let each party connect to the PHR of the client.

```
>>> python manage.py phr_connect doctor.json http://127.0.0.1:8000
Paste the keys data, excluding the BEGIN and END block: (...paste key for doctor...)
Connected to record ID 28

...

>>> python manage.py phr_connect insurance.json http://127.0.0.1:8000
Paste the keys data, excluding the BEGIN and END block: (...paste key hospital-a...)
Connected to record ID 28
```

4.2.5 Reading data

In the first step, we encrypted two messages. One for only the doctor (#55), and one for both the insurance and employer (#56).

First the doctor:

```
>>> python manage.py phr_decrypt doctor.json 55
Record item content:

Hi, Doctor
>>> python manage.py phr_decrypt doctor.json 56
Cannot decrypt record item
```

Then the employer and insurance:

```
>>> python manage.py phr_decrypt employer.json 56
Record item content:

Hi, Insurance and Employer
>>> python manage.py phr_decrypt insurance.json 56
Record item content:
```

```
Hi, Insurance and Employer
>>> python manage.py phr_decrypt employer.json 55
Cannot decrypt record item
>>> python manage.py phr_decrypt insurance.json 55
Cannot decrypt record item
```

And to be sure, we verify the health club and hospital cannot read the messages (they have read access to different categories).

```
>>> python manage.py phr_decrypt healthclub-a.json 55
Cannot decrypt record item
>>> python manage.py phr_decrypt healthclub-a.json 55
Cannot decrypt record item
>>> python manage.py phr_decrypt hospital-a.json 56
Cannot decrypt record item
>>> python manage.py phr_decrypt hospital-a.json 56
Cannot decrypt record item
```

4.2.6 Retrieving write access

In step XXX we provided Hospital-A with a write key. The party should retrieve the key from the server.

```
>>> python manage.py phr_retrieve hospital-a.json HEALTH
Key(s) imported for HEALTH.
```

The same does not work for other parties, e.g. Hospital-B and Health Club-A:

```
>>> python manage.py phr_retrieve hospital-b.json HEALTH
No new key(s) imported.
>>> python manage.py phr_retrieve healtclub-a.json HEALTH
No new key(s) imported.
```

Just to verify the keys are imported, one can issue the status command.

```
>>> python manage.py phr_status hospital-a.json
Host:        http://127.0.0.1:8000
Categories:  PERSONAL, HEALTH, TRAINING
Parties:     DOCTOR, INSURANCE, EMPLOYER, HOSPITAL+A+B+C, HEALTHCLUB+A+B+C
Mappings:    PERSONAL -> DOCTOR, INSURANCE, EMPLOYER
             TRAINING -> HEALTHCLUB
             HEALTH -> HOSPITAL

Record ID:   28
Record name: Patient record
Record role: HOSPITAL-A

Master keys: <not set>
Public keys: HEALTH
Secret keys: HOSPITAL-A -> HEALTH
```

For the above case, Hospital-A has connected to the right record, is able to decrypt messages for Hospital-A in the Health category and can encrypt messages for the Health category. Note that the master keys are not available, so Hospital-A cannot generate new keys (only the patient can do this).

4.2.7 Writing a reply

Now Hospital-A has write access, he can write data to the Health category. To verify this, we will write a message to the Health category only for the patient and one to all hospitals.

```
>>> python manage.py phr_encrypt hospital-a.json HEALTH HOSPITAL "Hello patient!""
Uploaded to record item ID 57
>>> python manage.py phr_encrypt hospital-a.json HEALTH HOSPITAL "Hello patient and
    other hospitals!""
Uploaded to record item ID 58
```

To conclude, each addressed party will decode it:

```
>>> python manage.py phr_decrypt hospital-a.json 57
Record item content:

Hello patient!
>>> python manage.py phr_decrypt hospital-b.json 57
Cannot decrypt record item
>>> python manage.py phr_decrypt hospital-c.json 57
Cannot decrypt record item
>>> python manage.py phr_decrypt hospital-a.json 58
Record item content:

Hello patient and other hospitals!
>>> python manage.py phr_decrypt hospital-b.json 58
Record item content:

Hello patient and other hospitals!

>>> python manage.py phr_decrypt hospital-c.json 58
Record item content:

Hello patient and other hospitals!
```

And the patient:

```
>>> python manage.py phr_decrypt patient.json 57
Record item content:

Hello patient!
>>> python manage.py phr_decrypt patient.json 58
Record item content:

Hello patient and other hospitals!
```

4.2.8 Conclusion

In een tabel zetten, zoals Ruud dat deed by HWSec

Requirement 1: 4.2.2 Requirement 2: 4.2.1, 4.2.4 Requirement 3: 4.2.3, 4.2.6, 4.2.7 Requirement 4: 4.2.3, 4.2.6, 4.2.7 (generalized)

5 Discussion

5.1 Limitations

Authentication

Number of keys

Key spreading -> per party, RW attributes?

upfront writing of data

6 References