Secure Data Management - Practical Assignment

Suzanne Maquelin(s2661640) Kes Greuter (s1932764)
Djoshua Moonen (s2660539) Adriaan de Vos (s2606240)

November 15th 2021

1 Introduction

Personal health data is considered sensitive information as this information can often uniquely identify a patient and it contains confidential information about their health that they are not able to change if the information is leaked. Therefore, there has been a lot of research on the security of Personal Health Record (PHR) systems [9]. These systems need to adhere to strict regulations regarding the storage, processing, and accessing of sensitive health information.

For this assignment, we will design and implement a demonstrator of a PHR system. In this system, the patient is in control of their own data and it also has the ability to provide read access to (parts of) their health records to concerned third parties. The following requirements are copied from the assignment description as they provide the guidelines for our decision-making process, our design phase, and our code implementation:

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

Storing data such as health records on a remote server brings certain threats. The server may get compromised in which case the confidentiality of the data is at stake. A solution to this problem is storing only encrypted data on the server. This leads to some challenges such as the distribution of keys and allowing for fine-grained access control.

The Ciphertext-Policy Attribute-Based Encryption (CP-ABE) access control model [4] aims to resolve these challenges. Our implementation is based on this model. A description of CP-ABE and specifics on the variant we use can be found in Section 5.

First, the assumptions on which the demonstrator is built are described in Section 2. The data model is given in Section 3. Section 4 sets out the access policies. The system architecture that is used for the implementation of CP-ABE is described in Section 6. The report is concluded with a discussion on the advantages and disadvantages of our designed solution in Section 7.

2 Assumptions

As the provided assignment only contains four requirements there are still many things not clearly defined. In this section, we will enumerate the assumptions we made during the design and development phase. The assumptions we made have been discussed with all party members and they influence the outcome of our project. To make it more clear why we have made the decisions throughout this document, and to make the research process reproducible we have included them below:

First, we assume authentication and access control with the Trusted Authority (TA) and file repository is in place. This is necessary for the key distribution as well as controlling who can insert information into a patient record. This comes down to a classical access control model. This model is enhanced in our demonstrator by using CP-ABE. Using CP-ABE eliminates the need to authenticate parties who download from the database and hence eliminates the risk of the access control manager being fooled to give out data to an unauthorized party. Further, the data uploaded is encrypted which ensures data confidentiality.

Second, users with upload rights are assumed to respect the access policies. For example, we assume that a hospital will upload a file with access policies allowing only the patient to read it. The demonstrator makes the user uploading responsible for stating who has access to the uploaded file. The users (except patients) uploading to a record of a certain patient should only allow that patient access. Patients uploading a record are not restricted.

Third, we assume that once a patient has provided his doctor, insurance or employer with read access, the patient will not switch to another doctor, insurance or employer. This assumption is necessary as read access cannot be revoked in the demonstrator. This assumption and how to create a system without it will be further discussed in Section 7.

Fourth, we assume that the patient can provide additional read access after initialization of the system, therefore the Trusted Authority which is used in CP-ABE to setup and key generation will always be online. In traditional CP-ABE, the TA is only required to be online during the setup phase. However, as explained in Section 6.3.3 the patient providing read access to new parties requires new keys to be generated.

3 Data model

This section will describe our designed data model for a PHR system which contains the entities, their attributes, and the relation between them. A graphical overview is given in Figure 1. The actors within this system are defined by the assignment requirements and are: patient, doctor, insurance, employer, hospital and health club. In addition, we define personal health records that can contain hospital-related information as well as training-related information. Below we will give a definition and description of all these elements within our PHR system:

Patient — A patient has one record and is treated by none, one or multiple hospitals. A patient can have none, one or multiple doctor(s), employer(s), and insurance(s). Furthermore, a patient can be a member of none, one or multiple health clubs. The information that is known about a patient within the PHR system is the patient ID, First name, Last name, Address, Postal code and Country.

Insurance — An insurance within the PHR system is linked to the patient. An insurance has none, one or multiple patient(s). The information that is known within the PHR system is the Insurance ID and the name of an Insurance.

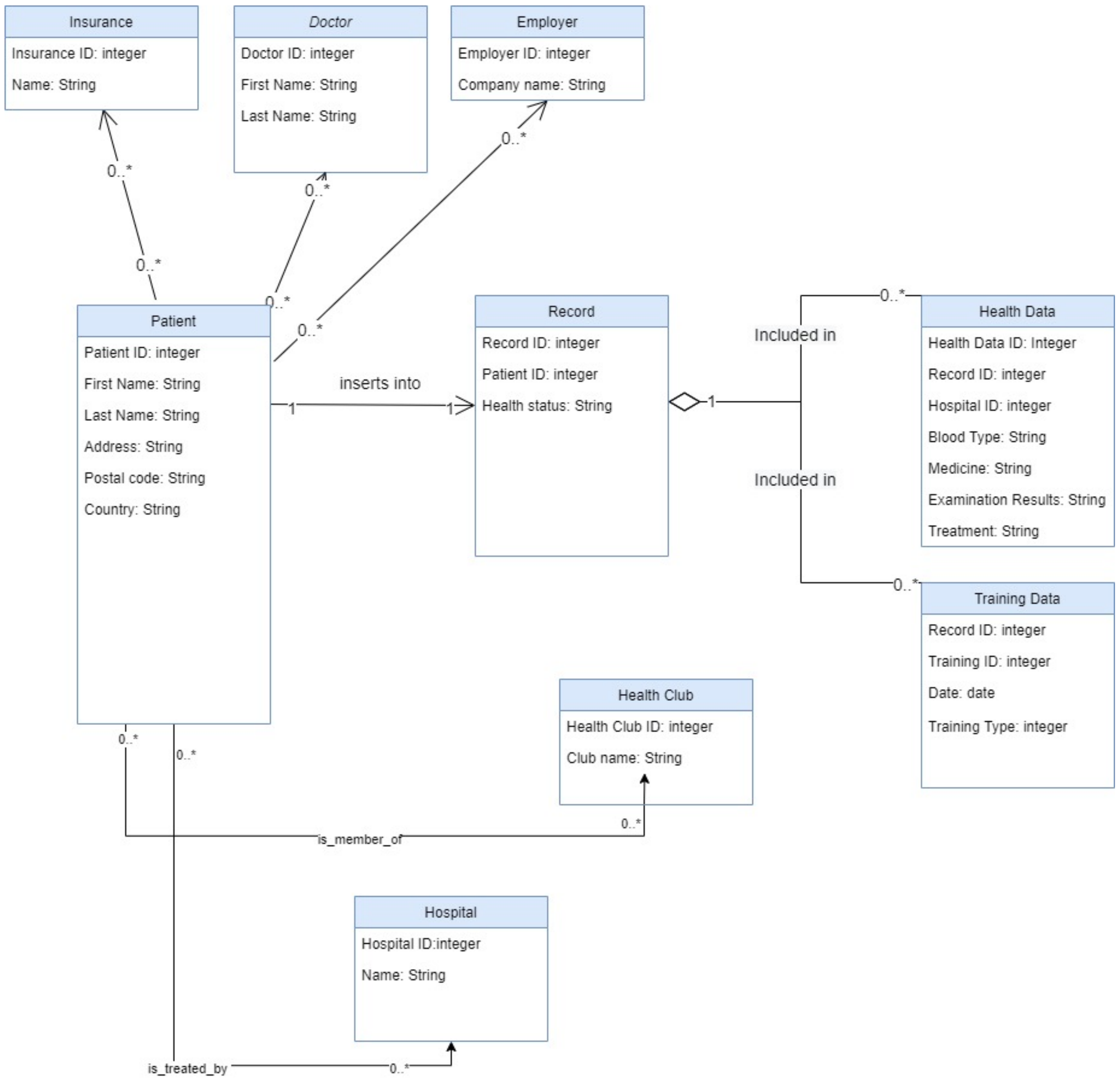


Figure 1: Data model

Doctor — A doctor in the PHR system is linked to a patient. The function of a doctor could be a general practitioner or a specialist assigned to a patient involved. A doctor can have none, one or multiple patients. The information that is known within the PHR system is the Doctor ID, the First name and the Last name. Note that the class of Doctor is not associated with the hospital within the data model. This is because the Doctor involved does not necessarily work within the hospital and has different rights than the hospital staff.

Employer — An employer can have none, one or multiple employees that are a patient within the PHR system. The information known about an Employer is the Employer ID and the Company name.

Hospital — A hospital has none, one or multiple patients. A hospital can insert patient health data for any patient that is treated within that specific hospital. The information known about a hospital is the hospital ID and the name of the hospital.

Health club — A health club is a club in which a member can exercise. A health club has a Health Club ID and a club name. A health club can have none, one or multiple members that are a patient within the PHR system.

Record— A Record contains none, one or more Health Data and none or more Training Data. The record includes Record ID, Patient ID and Health status.

Training Data — Training data records contain information about an executed training by the patient on a certain date. When a patient has completed a training, the information will be stored by creating a training data entry. Training data contains the following information; Training ID, Record ID, Training Type and Date. A record can have none, one or multiple Training data entries.

Health Data — Health data records contain information about treatments of a patient at a certain hospital. It includes the Health Data ID, the Record ID, Blood Type, Medicine, Examination Results and Treatment. A record can have none, one or multiple Health Data.

4 Access policies

Role \ Record	Health data	Training data
Patient	read/write	read/write
Hospital	write	-
Health club	-	write
Doctor	read subset	read subset
Insurance	read subset	read subset
Employer	read subset	read subset

Table 1: Access Policies

Table 1 gives a compact overview of the read/write access for each role. These access policies are determined based on the given requirements from the assignment. However, diverging from the requirements, a patient has not only read access but also write access to the training data. The patient has to be able to write training data because it needs to re-encrypt them to allow granular read permissions to other parties.

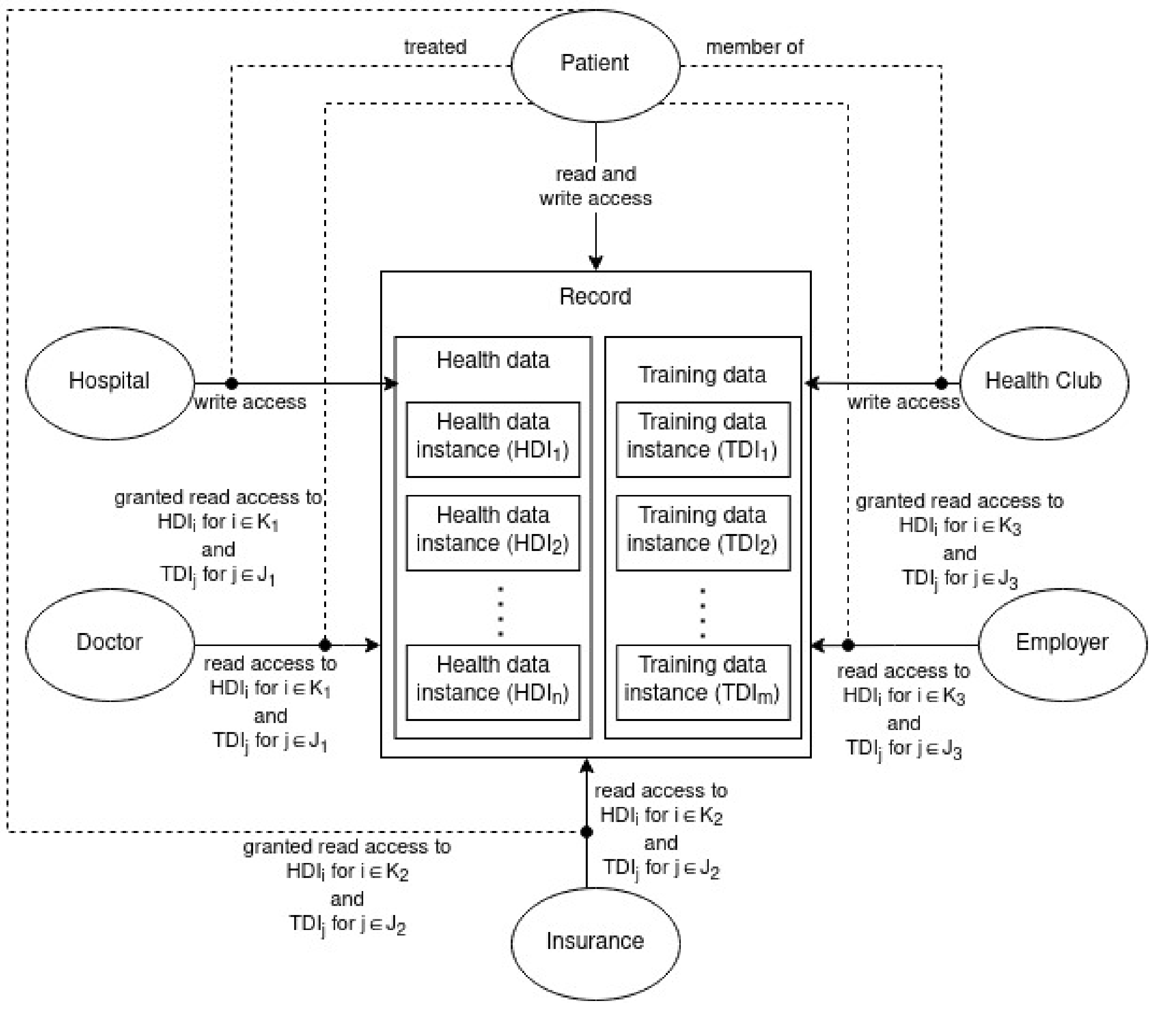

Patient
treated
member of
read and write access
Record
Health data
Health data instance (HDI_1)
Health data instance (HDI_2)
Health data instance (HDI_n)
Training data
Training data instance (TDI_1)
Training data instance (TDI_2)
Training data instance (TDI_m)
Hospital
write access
Health Club
write access
granted read access to HDI_i for $i \in K_1$ and TDI_j for $j \in J_1$
Doctor
read access to HDI_i for $i \in K_1$ and TDI_j for $j \in J_1$
granted read access to HDI_i for $i \in K_3$ and TDI_j for $j \in J_3$
Employer
read access to HDI_i for $i \in K_3$ and TDI_j for $j \in J_3$
read access to HDI_i for $i \in K_2$ and TDI_j for $j \in J_2$
granted read access to HDI_i for $i \in K_2$ and TDI_j for $j \in J_2$
Insurance

Figure 2: Access model

A more extensive overview with the conditions on each read/write access is shown in Figure 2. In the figure, the dotted lines show the conditions which need to be satisfied for the access rules. All dotted lines stem from the patient role since the patient can grant access or indirectly influence access by being a member of a health club or being a patient in a certain hospital. The patient can grant read access to a subset of his health data and/or training data by registering one of the following entities as related to him:

- A doctor
- An insurance company
- An employer

To represent partial access to data, the notation HDI_i for $i \in K$ and TDI_j for $j \in J$ is used. The HDI_i and TDI_j stand for Health Data Instance part i and Training Data Instance part j respectively. The set K is used to indicate only a subset of all HDIs can be accessed, and J similarly for TDIs.

A hospital is granted write access to health data of a patient only when it has treated that patient. A health club is allowed to write to training data only when the patient is a member of the club.

5 Access control model

The Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme is used in the access control model to enforce the access rules discussed in Section 4. CP-ABE can be used to enforce the read access policies, however, it cannot enforce the write rules as the encrypted uploads are not decrypted and verified by the server. Classical access control can mediate this shortcoming.

5.1 CP-ABE

CP-ABE is an encryption scheme that allows for fine-grained access control. It is typically used in a database setting to allow for more security than plaintext storing. This in combination with incorporating the access control into the encryption eliminates the need for an access control manager in the database. Another advantage of CP-ABE compared to public key encryption is that the user sharing data with multiple other users only need to encrypt once and does not need to know their exact identities.

There are four parts to the scheme, namely the setup, the key generation, the encryption and the decryption. The setup is performed by a Trusted Authority (TA). The scheme makes use of groups and bilinear maps, which are mathematical constructs. During the setup, the TA will use these mathematical constructs to generate a master key and a public key. Also, a set of attributes is generated. A subset of these attributes is assigned to each user. During the key generation stage, the TA will generate a secret key for each user which is associated with the user's subset of attributes. Each user will thus possess only one public key and one secret key. After this stage, the TA can go offline. Each user is allowed to encrypt messages and upload them to the database. The encryption process takes the plaintext as well as an access policy, which consists of logical operators and attributes. Each user is also allowed to download any ciphertext from the database. A ciphertext can only be decrypted by a secret key that contains the attributes that satisfy the access policy used during encryption.

Important to note is that CP-ABE is proven collusion resistant. This means that users colluding does not allow for decrypting any files that cannot be decrypted by one of the individuals. So, users with a different set of attributes cannot combine their secret keys to gain additional access rights.

There exist many variations of CP-ABE schemes [3][4][8]. In some schemes, it is not possible to update the set of attributes determined during setup at a later phase. When it is necessary to do so, the setup, key generation and distribution, as well as encryption, should be redone. The version used for our demonstrator, as described in section 5.2, does not have this downside. The set of attributes can be extended at any time after setup. Of course, adding attributes to a user's attribute set makes it necessary to do a key generation for that specific user. Important to note in this scenario is that this is not the same as key revocation. The users still have their old secret key (which may contain permissions they should not have anymore) and any downloaded ciphertext. Key revocation is a possible addition to CP-ABE [7].

5.2 FAME: Fast Attribute-based Message Encryption

The CP-ABE variant used for our solution is described in [3]. This variant follows the general outline of CP-ABE with the four algorithms: setup, key generation, encryption and decryption. However, the mathematics in these algorithms differs from the original CP-ABE described by Bethencourt [4]. Amongst other things, type-III pairings are used which offers "good performance and flexibility for high security parameters" [6]. The main advantages of this version are computational efficiency (especially for decryption) and the security proof under standard assumptions. Further, the scheme provides the freedom of choosing arbitrary attributes, as these are mapped by a hash function to group elements, and produces short ciphertexts and keys, which makes it memory efficient as well.

The paper compares its performance and security assumptions to three other papers. One of these papers is the original CP-ABE scheme by Bethencourt [4]. Another is a scheme described is CGW [5] on which FAME builds. This scheme provides the mathematics for the shorter ciphertexts and keys as well as faster decryption but does not allow for arbitrary attributes. The third paper in the comparison is a scheme by Waters [8], which is based on access control by Linear Secret Sharing Scheme matrix M over the attributes within a system, using the d-Parallel Bilinear Diffie Hellman Exponent assumption. The ciphertext size and the encryption of this scheme are equal to the scheme by Bethencourt [4]. Figure 4 in the Appendix describes the CP-ABE scheme FAME. The scheme used a hash function that maps arbitrary binary strings to elements of group G [3].

6 System Architecture

Instead of implementing FAME ourselves, we use the implementation in Python provided by the authors of the paper on GitHub [2]. In this code the four algorithms: setup, key generation, encryption, and decryption are implemented. Our system invokes these algorithms and builds additional functionality such as adding new users, checking and storing uploads, and updating a user's secret key after adding new attributes.

The system exists of four classes: `Role`, `TA`, `User` and `PHR_Repo`. The `Role` class is the smallest and contains an ENUM of possible user roles. The `User` class acts as an user of the system and provides functionality for encryption and decryption of messages. The `PHR_Repo` class acts as a file repository and provides upload and download functionality for patient records. The `TA` class has the most functionality as the Trusted Authority is responsible for the FAME setup, key generation, adding new users, updating attribute sets and checking permissions for uploads.

6.1 Hybrid cryptosystem

CP-ABE is an encryption scheme that makes use of a mathematical construct called a group. The messages that can be encrypted by CP-ABE should be elements of a certain group. To allow for the freedom to encrypt any message, a hybrid cryptosystem can be used. A hybrid cryptosystem combines a public key encryption scheme and a private key encryption scheme. The public key encryption is used to encrypt the symmetric key. The symmetric encryption scheme is used to encrypt the actual message [1]. CP-ABE will be the public key encryption scheme in the hybrid system. For symmetric encryption, AES is used.

Our implementation slightly differs from a traditional hybrid cryptosystem. The public key encryption is not used to encrypt the key for the symmetric encryption, but to encrypt the random group element that is used to generate the symmetric key.

6.2 Implementing FAME

6.2.1 Attributes

The attributes in the system reflect the roles of the users, described in Section 3 and Section 4, namely `PATIENT`, `DOCTOR`, `HOSPITAL`, `HEALTH CLUB`, `INSURANCE` and `EMPLOYER`. Then there are n more attributes, one per patient, namely `RELATED-TO-`*patient_id*, where the *patient_id* identifies a patient. This last type of attribute is used to identify users that have a relation with the patient and should be granted certain rights because of that relation.

A user's attribute set depends on his role and the patients with which it has a relation. A patient, with patient id 1, will always have the attribute set {`PATIENT`, `RELATED-TO-1`}. An employer that has employees with patient ids 1 and 5 will have an attribute set {`EMPLOYER`, `RELATED-TO-1`, `RELATED-TO-5`}. Note that different users can have the same attribute set, such as users with the role hospital.

The attribute set of a user is determined upon setup but can be changed during the running time of the system. These changes are explained in Section 6.3.3. The attribute set during setup is dependent on the role of a user.

6.2.2 Algorithms

Below we describe how we use the algorithms that are described in Section 5.2.

The setup algorithm is called in the `TA` class, which returns the master key (msk) and public key (pk). The public key will be distributed by the TA upon User creation.

The key generation is called by the TA as well as it requires the master key. The public key, master key and attribute set of an user are given as input and the algorithm returns a user-specific secret key. This secret key is then sent to the user.

The encryption functionality is implemented in `User`. For encryption, the hybrid encryption described in Section 6.1 is used. First, a random group element is generated. Next, the group element is hashed by SHA256 to create an AES key. The key is used to encrypt the file with AES encryption and the group element is encrypted by calling the FAME encryption algorithm. Both outputs are combined to a ciphertext.

The encryption of the group element with the FAME encryption algorithm requires an access policy. When the user encrypting is not a patient, this policy will always be `PATIENT` AND `RELATED-TO-`id, where id is the id of the patient whose record is updated. The reason is that only the patient is in charge of who should have read access to the files. This means that a file uploaded by a non-patient should be re-encrypted by the patient to allow read access to different users. An example of this re-encryption is shown in Figure 3. When the user encrypting is a patient, the policy can be any combination of attributes, but should at least contain `PATIENT` AND `RELATED-TO-`id such that the patient keeps access. According to our second assumption, the users respect the access policies.

Decryption is performed by the user as well. First, the FAME decryption algorithm is called to decrypt the group element. Next, the group element is hashed to get the AES key. Last, the AES key is used to

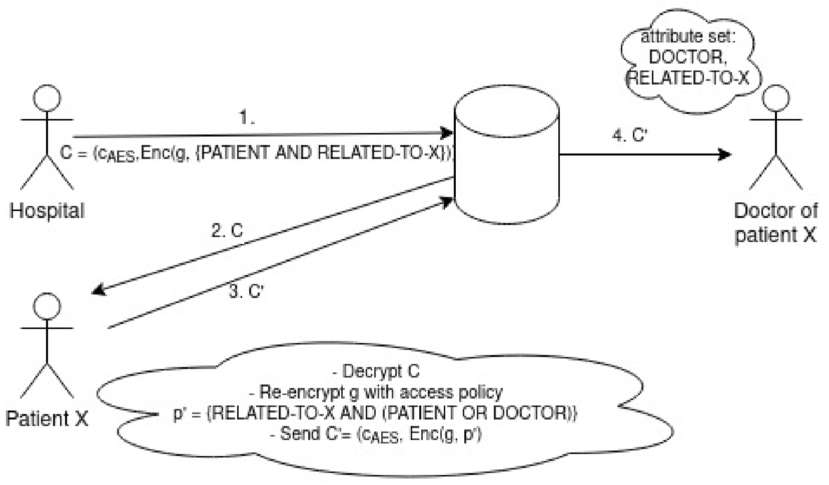

attribute set: DOCTOR, RELATED-TO-X
1.
C = (c_AES,Enc(g, {PATIENT AND RELATED-TO-X}))
4. C'
Hospital
Doctor of patient X
2. C
3. C'
Patient X
- Decrypt C
- Re-encrypt g with access policy
p' = {RELATED-TO-X AND (PATIENT OR DOCTOR)}
- Send C'= (c_AES, Enc(g, p')

Figure 3: Example of re-encryption

decrypt the file. In the first step, the FAME decryption will only be successful when the access policy is satisfied by the user key related to its attribute set.

6.3 Code functionalities

6.3.1 Access control for uploads

As previously discussed, CP-ABE does not enforce any rules on uploading to the database. Only patients, hospitals and health clubs should be allowed to upload and only to a record of a patient they are related to. The `PHR_Repo` calls the TA to perform this check since the TA is the only one with access to the required information.
The function `can_user_do_upload(self, user_id, patient_id)` checks if a given user is allowed to upload a file the record of the patient with `patient_id`. The function checks the user's attributes, which are stored in the TA. When this attribute set contains:

- `PATIENT` or `HOSPITAL` or `HEALTHCLUB`
- and `RELATED-TO-`*patient_id*

the upload should be allowed.

6.3.2 New users

New users can be added by the Trusted Authority. The order can only be given by the admin with id -1. New users are added with the function add_new_user(self, admin_id, role). If the user is a hospital, health club, doctor, insurance or employer, the user is assigned the attribute corresponding to their role; HOSPITAL, HEALTHCLUB, DOCTOR, INSURANCE or EMPLOYER. If the user is a patient, the attribute PATIENT is given to the user as well as the attribute "RELATED-TO user_id".

6.3.3 New relations

A patient can give read access to certain parts of its records to its doctor, employer or insurance. To make this possible, the patient should indicate who his doctor/employer/insurance is. The indicated party receives a new attribute `RELATED-TO-`id where id is the patient's id. This functionality is defined in the TA as the TA is the only one allowed to update the user's attribute sets. After this update, a new key is generated for the doctor/employer/insurance and they are notified to download it. This functionality requires the TA to be online after the setup phase.

6.3.4 PHR repositories

Personal Health Record Repository acts as a file server for the uploading, replacing and downloading of single records. Thus the users communicate with the PHR Repository. The PHR Repository in turn communicates with the TA upon receiving a file upload request. This is necessary to enforce write-access control on uploads. Additionally, a user can request file id's of health data and training data for a patient, or download a patient's whole record, thus all files in the record.

6.4 Testing

A PHR system works with sensitive information, which makes testing the integrity of the system very important. We have written extensive tests to ensure everything is working as intended. Tests are included for the PHR Repository, TA and User class.
An example of such an test, which we want to highlight, is included in the User test. It tests whether a user can decrypt a ciphertext with a secret key that does not satisfy the access policy. The test is called `test_invalid_decrypt_from_send`. It is integral for the integrity of the system that this is properly implemented to prevent unauthorized access to health records.

7 Discussion

Our implementation provides a good solution to the given requirements by using an efficient variant of the CP-ABE scheme called FAME, complemented with classical access control to ensure complete confidentiality of the patient records within the PHR system. All health records are stored encrypted which eliminates the risk of a compromised Access Control Manager which could compromise the confidentiality of the data.

Currently, the state of the system is only stored in memory and not saved in a database or on the disk. Our implementation works as a "live" system which has some risks that accompany this decision. The first risk is that stopping the system will cause complete data loss as all user information and their medical data is gone when the program closes. This means that the program can never be stopped to be updated and that any error or power outages will be fatal for the availability of the data. Secondly, there is also a risk of not scaling well with a huge amount of users and health records as it is all stored in random access memory which is often limited to a few gigabytes in storage. To solve these issues, the implementation should be extended to save the users and permissions in a relational database. And to store all health records on the disk. This ensures better reliability and the ability to scale to several terabytes of data.

Another limitation of our implementation is that it does not implement key revocation. Based on the assignment description this was not required so we have made the assumption that patients do not switch doctors, employers, or insurances. This allowed us to reduce the complexity of our implementation significantly. However, if the need arises to have this functionality this can be added by extending our current implementation. For some use cases, it would be very useful to allow a user to revoke access to part of his health record, or to revoke access to a doctor that is removed from the system. This problem can be solved by implementing a mediated CP-ABE scheme in which a proxy is utilized, an example of such a mediated CP-ABE scheme is the 2009 scheme of Ibraimi [7].

One of our assumptions ensures that the users always provide their correct identifier when communicating with the trusted authority and the file repository. This would not work in practice, as a malicious user could replace this identifier to retrieve keys from other users, or to do malicious uploads. In addition, we also assume that every user sets valid access policies when creating new entries as explained more extensively in section 2. To mitigate this issue there should be authentication of the user by, for example, a login system with a username and password. We decided to not implement this, as we wanted to focus on the CP-ABE part instead of spending time implementing classic access control for key distribution. Therefore, for our implementation, every user is assumed to only use their own identifier.

References

[1] Hybrid cryptosystem. `https://en.wikipedia.org/wiki/Hybrid_cryptosystem`.

[2] S. Agrawal. Github attribute based encryption. `https://github.com/sagrawal87/ABE`.

[3] S. Agrawal and M. Chase. Fame: fast attribute-based message encryption. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 665–682, 2017.

[4] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE symposium on security and privacy (SP'07)*, pages 321–334. IEEE, 2007.

[5] J. Chen, R. Gay, and H. Wee. Improved dual system abe in prime-order groups via predicate encodings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 595–624. Springer, 2015.

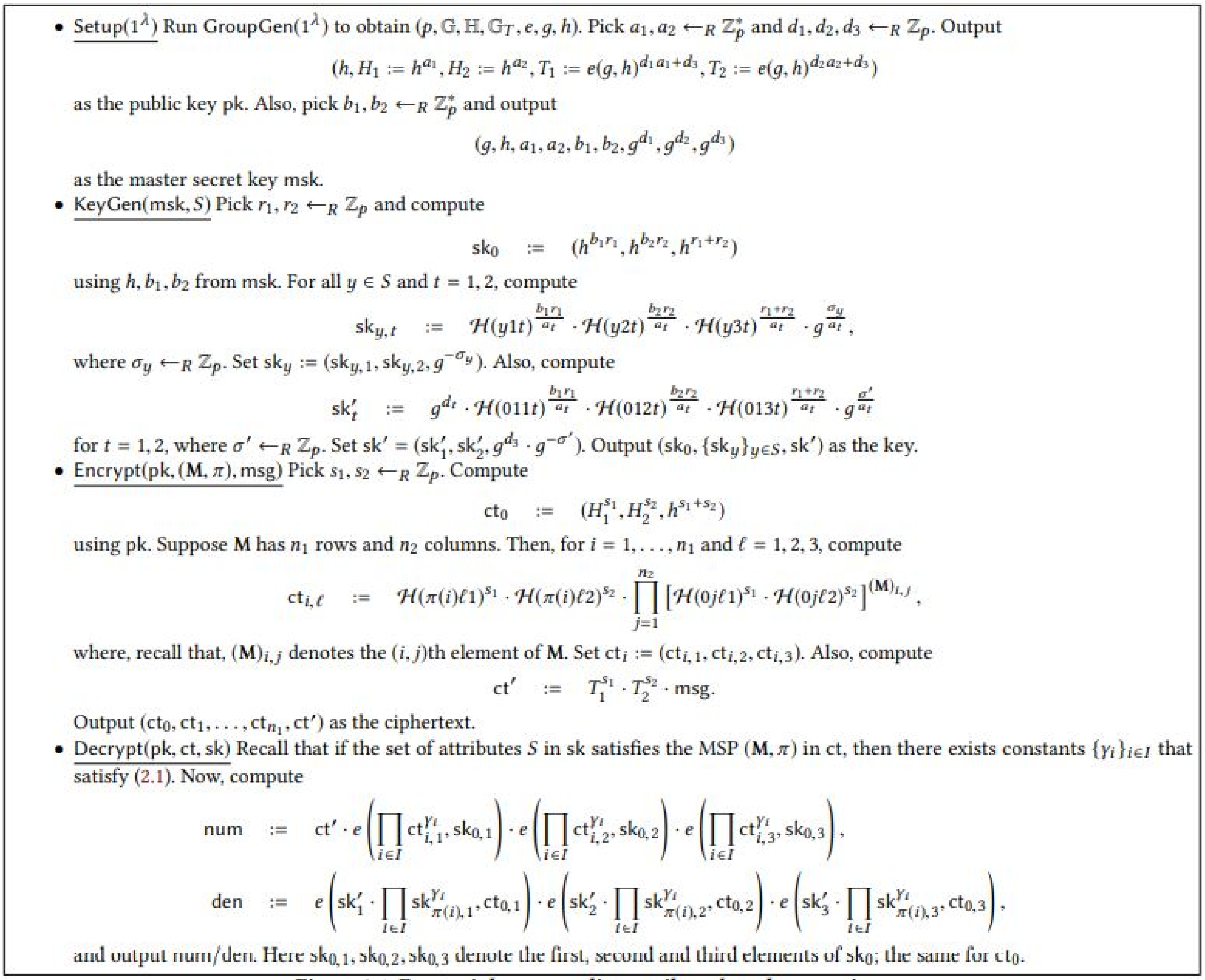
[6] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[7] L. Ibraimi, M. Petkovic, S. Nikova, P. Hartel, and W. Jonker. Mediated ciphertext-policy attribute-based encryption and its application. In *International Workshop on Information Security Applications*, pages 309–323. Springer, 2009.

[8] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *International Workshop on Public Key Cryptography*, pages 53–70. Springer, 2011.

[9] K. T. Win, W. Susilo, and Y. Mu. Personal health record systems and their security protection. *Journal of medical systems*, 30(4):309–315, 2006.

8 Appendix



- $\underline{\mathsf{Setup}(1^\lambda)}$ Run $\mathsf{GroupGen}(1^\lambda)$ to obtain $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h)$. Pick $a_1, a_2 \leftarrow_R \mathbb{Z}_p^*$ and $d_1, d_2, d_3 \leftarrow_R \mathbb{Z}_p$. Output

$$(h, H_1 := h^{a_1}, H_2 := h^{a_2}, T_1 := e(g,h)^{d_1 a_1 + d_3}, T_2 := e(g,h)^{d_2 a_2 + d_3})$$

as the public key pk. Also, pick $b_1, b_2 \leftarrow_R \mathbb{Z}_p^*$ and output

$$(g, h, a_1, a_2, b_1, b_2, g^{d_1}, g^{d_2}, g^{d_3})$$

as the master secret key msk.

- $\underline{\mathsf{KeyGen}(\mathsf{msk}, S)}$ Pick $r_1, r_2 \leftarrow_R \mathbb{Z}_p$ and compute

$$\mathsf{sk}_0 \quad := \quad (h^{b_1 r_1}, h^{b_2 r_2}, h^{r_1 + r_2})$$

using h, b_1, b_2 from msk. For all $y \in S$ and $t = 1, 2$, compute

$$\mathsf{sk}_{y,t} \quad := \quad \mathcal{H}(y1t)^{\frac{b_1 r_1}{a_t}} \cdot \mathcal{H}(y2t)^{\frac{b_2 r_2}{a_t}} \cdot \mathcal{H}(y3t)^{\frac{r_1 + r_2}{a_t}} \cdot g^{\frac{\sigma_y}{a_t}},$$

where $\sigma_y \leftarrow_R \mathbb{Z}_p$. Set $\mathsf{sk}_y := (\mathsf{sk}_{y,1}, \mathsf{sk}_{y,2}, g^{-\sigma_y})$. Also, compute

$$\mathsf{sk}'_t \quad := \quad g^{d_t} \cdot \mathcal{H}(011t)^{\frac{b_1 r_1}{a_t}} \cdot \mathcal{H}(012t)^{\frac{b_2 r_2}{a_t}} \cdot \mathcal{H}(013t)^{\frac{r_1 + r_2}{a_t}} \cdot g^{\frac{\sigma'}{a_t}}$$

for $t = 1, 2$, where $\sigma' \leftarrow_R \mathbb{Z}_p$. Set $\mathsf{sk}' = (\mathsf{sk}'_1, \mathsf{sk}'_2, g^{d_3} \cdot g^{-\sigma'})$. Output $(\mathsf{sk}_0, \{\mathsf{sk}_y\}_{y \in S}, \mathsf{sk}')$ as the key.

- $\underline{\mathsf{Encrypt}(\mathsf{pk}, (\mathbf{M}, \pi), \mathsf{msg})}$ Pick $s_1, s_2 \leftarrow_R \mathbb{Z}_p$. Compute

$$\mathsf{ct}_0 \quad := \quad (H_1^{s_1}, H_2^{s_2}, h^{s_1 + s_2})$$

using pk. Suppose $\mathbf{M}$ has n_1 rows and n_2 columns. Then, for $i = 1, \ldots, n_1$ and $\ell = 1, 2, 3$, compute

$$\mathsf{ct}_{i,\ell} \quad := \quad \mathcal{H}(\pi(i)\ell 1)^{s_1} \cdot \mathcal{H}(\pi(i)\ell 2)^{s_2} \cdot \prod_{j=1}^{n_2} \left[\mathcal{H}(0j\ell 1)^{s_1} \cdot \mathcal{H}(0j\ell 2)^{s_2}\right]^{(\mathbf{M})_{i,j}},$$

where, recall that, $(\mathbf{M})_{i,j}$ denotes the (i,j)th element of $\mathbf{M}$. Set $\mathsf{ct}_i := (\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}, \mathsf{ct}_{i,3})$. Also, compute

$$\mathsf{ct}' \quad := \quad T_1^{s_1} \cdot T_2^{s_2} \cdot \mathsf{msg}.$$

Output $(\mathsf{ct}_0, \mathsf{ct}_1, \ldots, \mathsf{ct}_{n_1}, \mathsf{ct}')$ as the ciphertext.

- $\underline{\mathsf{Decrypt}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk})}$ Recall that if the set of attributes S in sk satisfies the MSP $(\mathbf{M}, \pi)$ in ct, then there exists constants $\{\gamma_i\}_{i \in I}$ that satisfy (2.1). Now, compute

$$\mathsf{num} \quad := \quad \mathsf{ct}' \cdot e\left(\prod_{i \in I} \mathsf{ct}_{i,1}^{\gamma_i}, \mathsf{sk}_{0,1}\right) \cdot e\left(\prod_{i \in I} \mathsf{ct}_{i,2}^{\gamma_i}, \mathsf{sk}_{0,2}\right) \cdot e\left(\prod_{i \in I} \mathsf{ct}_{i,3}^{\gamma_i}, \mathsf{sk}_{0,3}\right),$$

$$\mathsf{den} \quad := \quad e\left(\mathsf{sk}'_1 \cdot \prod_{i \in I} \mathsf{sk}_{\pi(i),1}^{\gamma_i}, \mathsf{ct}_{0,1}\right) \cdot e\left(\mathsf{sk}'_2 \cdot \prod_{i \in I} \mathsf{sk}_{\pi(i),2}^{\gamma_i}, \mathsf{ct}_{0,2}\right) \cdot e\left(\mathsf{sk}'_3 \cdot \prod_{i \in I} \mathsf{sk}_{\pi(i),3}^{\gamma_i}, \mathsf{ct}_{0,3}\right),$$

and output num/den. Here $\mathsf{sk}_{0,1}, \mathsf{sk}_{0,2}, \mathsf{sk}_{0,3}$ denote the first, second and third elements of sk_0; the same for ct_0.

Figure 4: FAME: CP-ABE based scheme [3]