2017

A General Framework for Secure Sharing of Personal Health Records in Cloud System

Man Ho Au
University of Wollongong, aau@uow.edu.au

Tsz Hon Yuen
University of Hong Kong, Huawei International Pte Ltd, thy738@uow.edu.au

Joseph K. Liu
Monash University, ksliu@i2r.a-star.edu.sg

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Xinyi Huang
Fujian Normal University, xh068@uow.edu.au

See next page for additional authors

Recommended Citation

Au, Man Ho; Yuen, Tsz Hon; Liu, Joseph K.; Susilo, Willy; Huang, Xinyi; and Jiang, Zoe L., "A General Framework for Secure Sharing of Personal Health Records in Cloud System" (2017). *Faculty of Engineering and Information Sciences - Papers: Part B*. 149.
https://ro.uow.edu.au/eispapers1/149

A General Framework for Secure Sharing of Personal Health Records in Cloud System

Abstract

Personal Health Record (PHR) has been developed as a promising solution that allows patient-doctors interactions in a very effective way. Cloud technology has been seen as the prominent candidate to store the sensitive medical record in PHR, but to date, the security protection provided is yet inadequate without impacting the practicality of the system. In this paper, we provide an affirmative answer to this problem by proposing a general framework for secure sharing of PHRs. Our system enables patients to securely store and share their PHR in the cloud server (for example, to their carers), and furthermore the treating doctors can refer the patients' medical record to specialists for research purposes, whenever they are required, while ensuring that the patients' information remain private. Our system also supports cross domain operations (e.g., with different countries regulations).

Keywords

health, personal, sharing, secure, system, framework, cloud, general, records

Disciplines

Engineering | Science and Technology Studies

Publication Details

Au, M. Ho., Yuen, T., Liu, J. K., Susilo, W., Huang, X. & Jiang, Z. L. (2017). A General Framework for Secure Sharing of Personal Health Records in Cloud System. Journal of Computer and System Sciences, Online First

Authors

Man Ho Au, Tsz Hon Yuen, Joseph K. Liu, Willy Susilo, Xinyi Huang, and Zoe L. Jiang

A General Framework for Secure Sharing of Personal Health Records in Cloud System

Man Ho Au[a,], Tsz Hon Yuen[b], Joseph K. Liu*[c], Willy Susilo[d], Xinyi Huang[e], Yang Xiang[f], Zoe L. Jiang[g]

[a]*Department of Computing, The Hong Kong Polytechnic University, Hong Kong*
[b]*Huawei, Singapore*
[c]*Faculty of Information Technology, Monash University, Australia*
[d]*School of Computing and Information Technology , University of Wollongong, Australia*
[e]*School of Mathematics and Computer Science, Fujian Normal University, China*
[f]*School of Information Technology, Deakin University, Australia*
[g]*School of Computer Science and Technology, Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China.*

Abstract

Personal Health Record (PHR) has been developed as a promising solution that allows patient-doctors interactions in a very effective way. Cloud technology has been seen as the prominent candidate to store the sensitive medical record in PHR, but to date, the security protection provided is yet inadequate without impacting the practicality of the system. In this paper, we provide an affirmative answer to this problem by proposing a general framework for secure sharing of PHRs. Our system enables patients to securely store and share their PHR in the cloud server (for example, to their carers), and furthermore the treating doctors can refer the patients' medical record to specialists for research purposes, whenever they are required, while ensuring that the patients' information remain private. Our system also supports cross domain operations (e.g., with different countries regulations).

Keywords: Personal Health Record, Sharing, Cloud computing

*Corresponding author.

1. Introduction

The focus of healthcare has been shifted from healthcare providers' paternalistic approach to patient-oriented approach. In this approach, patients are well informed about their health information. They tend to understand, follow instructions and ask more insightful questions. Allowing patients to access their own records will encourage them to be involved in their own healthcare. This will strengthen the patient-provider relationship and enhance the effectiveness of healthcare management.

CLOUD TECHNOLOGY FOR PHR: Recently, Personal Health Record (PHR) has been developed as a patient-centric model of the logistic of health information by using the *cloud technology*. PHR could be seen as the solution for better management of an individual's health, and as the tool that will empower the patient in correlation with healthcare providers through the ability to provide his/her own medical history. Furthermore, PHR allows a patient to create, retrieve and manage his/her personal health data record from one place to another place, through a cloud server. For example, Alice may first see a doctor at Clinic A. Then the doctor refers her to Specialist B. Alice may also need to take blood sample at laboratory C. Finally Alice goes to Hospital D for acquiring medical treatment. By using PHR, Alice does not need to bring along or transfer his/her paper-based medical record among different healthcare providers, which contributes to the improvement of patient's comfort level by using cloud technology.

Data security is a key ingredient for cloud computing and especially important for cloud-based biomedical applications. Dealing with different national legal regulations and procedures accepted by the medical community requires a careful approach [31]. Thus it is particularly important for PHR to be securely instantiated. One of the major concerns is about whether the patients can actually control the sharing of their sensitive personal health information.

LEGAL REQUIREMENTS: Legislation regarding confidentiality of health information is in place in countries around the world. In US, HIPAA (Health Insurance Portability and Accountability Act) regulates the privacy and confidentiality of health information, and there are sections regarding health research in HIPAA. PIPEDA (Personal Information Protection and Electronic Document Act) in Canada and Data Protection Act 1998 (effective since 2000) in UK are some of the examples that are in place for privacy and confidentiality of health information. In Australia, New South Wales

Health Records and Information Privacy Act 2002 which was effective since September 2004 also stated "The organization that holds health information must not use the information for a purpose other than the purpose for which it was collected unless the use of health information for the secondary purpose is reasonably necessary for research, or the compilation or analysis of statistics in the public interest". The legislation makes the situation more complicated.

A feasible solution is to encrypt the data prior to outsourcing to the cloud. Symmetric encryption does not work as it requires the encryptor and the decryptor to use the same key. It is likely both parties are not the same (e.g. the laboratory encrypts a medical record while a clinic decrypts it). In this case, key management remains the biggest issue. Asymmetric encryption can be used instead. Compared to the traditional public key infrastructure (PKI), Identity-based encryption (IBE) [38] shall be preferred as the encryptor (e.g. laboratory) does not need to obtain the patient's digital certificate in advance which maybe inconvenient or impractical. Instead, it may just use his name or national number as the identity to encrypt the data. The patient can decrypt the data using his own secret key, which is given by a Private Key Generator (PKG). In practice, it may be a government authority.

It seems IBE can act as the security solution for PHR in the *theoretical* framework. However, when we look into the practical scenario, it seems to be difficult to facilitate, and hence impractical. IBE can only allow one specific person to decrypt. Other than this particular person, no one else (except the PKG) can decrypt. It is suitable if there is no sharing of PHR among different persons, but this will lead to an impractical situation.

SHARING OF RECORDS: In some cases, doctors may want to discuss the patient's medical situation with other doctors or researchers. It is especially important if the case is very special or has never been discovered before. It can not only help the patient who may receive assistant from other specialists, but also help future patients with the same symptom. Thus it would be good if the medical system allows the sharing of PHR among specialists or researchers in the same area. IBE cannot facilitate this kind of sharing as it only allows the patient himself/herself to decrypt but not anyone else.

Sharing of PHR is trivial if privacy is not a concern. However, no one wants his/her personal PHR to be disclosed completely even for research purposes. It is also enforced by legal requirements in some countries.

We adopt attribute-based encryption (ABE) as the cryptographic primitive to serve this purpose. By using ABE, access policies are expressed as attributes of users or data, which enables a patient to selectively share his/her PHR among a set of users (with eligible attributes) without knowing their identities. In addition, we also deploy proxy re-encryption mechanism so that the ciphertext for Doctor-A can be transformed to the ciphertext for Doctor-B. This will enrich the secure sharing functionality of PHR.

Our Contributions. We propose a general framework for secure sharing of PHRs. Our system can be deployed using the cloud technology, and hence, it is a very practical and viable solution. Our system possesses the following attractive features, while ensuring the patient's privacy:

1) It allows a patient to securely store and share his/her PHR (e.g. to his/her carers) in the cloud server.

2) A doctor can share his/her patient's medical record to other specialists for research purpose, if it is necessary. Yet the personal information of the patient remains concealed.

3) Our system supports instantiation within the same domain (e.g. in the same country) and different domains. It also supports multiple authorities with different duties for key issuing tasks. It can prevent a single point of failure.

2. Related Works

We review some related works in this section, which consists of two parts. The first part describes some cryptographic primitives that can be used or related to our goal. The second part describes PHR secure sharing protocols in the literature and discusses why these protocols fail to fully fit our goal.

2.1. Cryptographic Primitives

Attribute-Based Encryption (ABE). ABE allows the data owner to share data with others in a more flexible way. At a high level, each potential data recipient is associated with an attribute set S and given the associated private key generated by a third party. The data provider can encrypt data with a set of function. Anyone with an attribute set can successfully decrypt the ciphertext if and only if the set fulfills the function. ABE was first introduced in [37]. Later on several variants and improvements have been proposed [18, 21, 40, 34], including ciphertext-policy ABE (CP-ABE, where ciphertexts are associated with access policies and keys are associated with sets of attributes) and key-policy ABE (where keys are associated with access policies and ciphertexts are associated with sets of attributes). Variants of ABE are proposed later, such as [13, 41] for decentralizing the trusted private key generator; [17] for outsourcing computation, [15] for efficient revocation, ABE for circuits [12], online/offline ABE [14], leakage-resilient ABE [19, 44], dual predicates and policies [3] and multi-authority ABE with large universe [35].

ABE has been found useful in situations with the requirement of fine-grained control on sensitive data such as PHR. An attribute-based infrastructure for PHR systems was proposed in [33], where each patient's health records are encrypted using a variant of CP-ABE that allows direct revocation. To achieve a flexible control on the encrypted data, a variant of ABE that allows delegation of access rights is proposed in [27, 42]. Unfortunately, none of these proposals allows the doctor to share patient's medical record with other specialists while preserving the patient's privacy.

Proxy Re-Encryption (PRE). PRE, first introduced in [5] and formally studied in [2], allows a semi-trusted proxy with a re-encryption key to convert ciphertexts computed under Alice's public key into those can be decrypted using Bob's private key [23, 25, 10]. The desirable feature is that the conversion is performed by the proxy without knowing the corresponding plaintexts.

Attribute-based Proxy Re-Encryption (ABPRE). Integrating ABE and PRE is a promising solution for more flexible data sharing. The first ciphertext-policy ABPRE scheme was proposed by Liang et al. [26], in which a proxy is allowed to transform a ciphertext under a specialized access policy into the one under another access policy (i.e. attribute-based re-encryption). Mizuno and Doi [32, 11] proposed a hybrid PRE scheme (in general) where ciphertexts generated in the context of ABE can be converted to the ones which can be decrypted in the IBE setting. Other works on attribute-based proxy re-encryption include [30, 43]. But all aforementioned schemes support access policy with AND gates only. The first construction of a CCA secure CP-ABPRE supporting any monotonic access policy was given in [24]. However, it only supports one single private key generator (PKG) and is not suitable for a large-scale PHR system which needs multiple authorities with different duties (as required in our system).

2.2. Other Secure PHR Protocols

In the literature, there exists a few secure PHR protocols. Here we discuss some of the state-of-the-art ones. In [22], Li et al. presented a patient-centric framework and a suite of mechanisms for data access control to PHRs stored in semi-trusted servers using ABE. They allow the data owner (the patient) to select the set of users who can access his own PHR. These may include his family member, his doctor or hospital medical staffs. They also include a "break-glass" emergency case such that the emergency department can always access a patient's PHR. It seems the protocol provides a completed picture of PHR instantiation. However there are still some limitations. For example, in case a doctor may need to discuss a particular PHR with other researchers (such as a clinical finding), the protocol does not allow the doctor to do so. Also it requires all users to be placed in the same domain (e.g. same country). But in the real world, doctors or researchers from different countries are likely to jointly discuss for some particular cases. Meanwhile our proposed framework overcomes these limitations.

Technical Motivation. Other existing PHR systems using ABE can be found in [16, 1]. However, they do not provide any sharing mechanism for PHRs. On the other side, the protocol in [29] allows the sharing of personal health information (PHI) (instead of PHR) collected from body sensor networks (BSNs) and securely shared by smartphone or opportunistic computing devices for emergency purpose. Thus their concept is totally different from ours. The other ABPREs (discussed in Sec. 2.1) do not allow multiple authorities to split the task of issuing user secret keys. Our proposed system provides all desired features as described in last section while no single scheme in the literature does. This contributes the main technical motiviation of this paper.

3. Overview

3.1. Entities

There are a number of entities in our PHR system:

1) The Private Key Generator (PKG): It is further divided into two types:

i) Central Authorities (CA): They are responsible for generating secret keys for the PHR system, e.g. Health Department, state govenment, etc. We employ the multi-authority setting for CA to avoid the key escrow problem of a single authority. We assume that at least one authority is honest. The CAs are responsible for issuing keys to users according to their unique identifiers.

ii) Attribute Authorities (AA): The authorities who are responsible for assigning attributes to user's secret key. For example, an hospital can assign

the attributes like “Hosptial A”, “City = NY” to all its staff. The medical association can assign the attributes related to medical professional license like “Cardiologist”, “Psychiatrist” to different doctors or nurses. Universities, laboratories and pharmacies are possible AAs.
2) Users: The patients, doctors, nurses, researchers and the emergency departments are treated similarly when they obtain keys from the CAs and the AAs. Therefore, they perform the same decryption process using their own attribute-based keys.
3) Cloud Server: It is responsible for storage purpose only. It can be accessed by all parties.

3.2. Functional Requirements

We require our system to achieve the following functions: **1) Data Confidentiality**: Unauthorized users and cloud server cannot decrypt a PHR document even they are colluding. Colluding CAs also cannot decrypt any PHR document either, provided that at least one of them is honest. **2) User Revocation**: If a user’s attribute is no longer valid, the user cannot access the PHR using that attribute. **3) Secure Sharing**: If a user is able to decrypt a PHR document, the user can also securely share the decryption right to another set of users (with a particular set of attribute). **4) Emergency Access**: The emergency department can decrypt any PHR in the emergency situation.

3.3. Threat Model

We consider the threat from three entities. For the CA, as we deploy multiple CAs setting, we require at least one CA to be honest and not curious. Other remaining CAs can be honest but curious. That means they may try to find out as much secret information from the transcripts but they will honestly follow the protocol. For the cloud server, it may try to access the files stored inside. For users, they may try to access the files beyond their privileges. For example, a doctor from “Hospital B” may try to access the files which can be accessed only to doctors from “Hospital A”.

3.4. Notation

Notations used in our system are shown in Table 1.

Table 1: Frequently Used Notations

d	index of the CA
D	total number of CAs in the system
$\mathbb{D}$	set of different CAs
k	index of the AA
K	total number of AAs in the system
$\mathbb{U}_k$	set of attributes managed by AA_k
n_k	the maximum number of attributes each user can get from AA_k
$\mathbb{K}$	set of different AAs
$\mathbb{U} = \bigcup_{k=1}^{K} \mathbb{U}_k$	attribute universe
a	an attribute
$\mathbb{A}$	an access policy for a PHR document
id	identifier of a user
S_{id}	id's set of attributes obtained from different AAs
GPK	global public parameter
$\mathsf{CPK}_d, \mathsf{CAPK}_d$	public keys of CA_d
CMSK_d	secret keys of CA_d
$\mathsf{APK}_k, \mathsf{ACPK}_k$	public keys of AA_k
AMSK_k	secret keys of AA_k
$\mathsf{ucsk}_{\mathsf{id},d}$	user-central-key for id, obtained from CA_d (part of his decryption key)
$\mathsf{ucpk}_{\mathsf{id},d}$	user-central-public-key for id, obtained from CA_d (used to obtain attributes from AAs for his decryption key)
$\mathsf{uask}_{a,\mathsf{id}}$	user-attribute-key for id with attribute a
$\mathsf{DK}_{\mathsf{id}}$	decryption key for id
RK	re-encryption key

4. Secure Sharing (Same Domain)

We first consider the case of secure sharing of PHR within the *same* domain. In practice, it maybe the case that all patients in the same country use the same healthcare system.

4.1. Definition

Access Policy. We define how the access policy, like "(Hospital A AND Cardiologist) OR Patient ID = Alice OR ER Dept", can be expressed as any monotonic access formula. We review their definitions.

Definition 1 (Access Structure [4, 40]). *Let* $\{P_1, \ldots, P_n\}$ *be a set of parties. A collection* $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}}$ *is monotone if* $\forall$ B,C : *if* $B \in \mathbb{A}$ *and* $B \subseteq C$ *then* $C \in \mathbb{A}$. *A monotone access structure is a monotone collection* $\mathbb{A}$ *of non-empty subsets of* $\{P_1, \ldots, P_n\}$. *The sets in* $\mathbb{A}$ *are called the authorized sets, and the sets not in* $\mathbb{A}$ *are called the unauthorized sets.*

In our PHR scheme, the role of the user is taken by the attributes (e.g. Cardiologist, Hospital A). Thus, the access structure $\mathbb{A}$ contains the authorized sets of attributes. It is shown in [4] that any monotone access structure can be realized by a linear secret sharing scheme.

Definition 2 (Linear Secret-Sharing Schemes (LSSS) [40]). *A secret sharing scheme* Π *over a set of parties* $\mathbb{P}$ *is called linear (over* $\mathbb{Z}_p$*) if* **1)** *The shares for each party form a vector over* $\mathbb{Z}_p$*.* **2)** *There exists a* share-generating matrix A *for* Π *which has* l *rows and* n *columns. For* $i = 1, \ldots, l$*, the* i^{th} *row of* A *is labeled by a party* $\rho(i)$ *(*ρ *is a function from* $\{1, \ldots, l\}$ *to* $\mathbb{P}$*). When we consider the column vector* $\vec{v} = (s, r_2, \ldots, r_n)$*, where* $s \in \mathbb{Z}_p$ *is the secret to be shared and* $r_2, \ldots, r_n \in \mathbb{Z}_p$ *are randomly chosen, then* $A\vec{v}$ *is the vector of* l *shares of the secret* s *according to* Π*. The share* $(A\vec{v})_i$ *belongs to party* $\rho(i)$*.*

Every LSSS defined above enjoys the linear reconstruction property [4]: Suppose that Π is an LSSS for access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be an authorized set, and let $I \subset \{1, \ldots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. There exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} \omega_i \lambda_i = s$. Furthermore, these constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix A. For any unauthorized set, no such constants exist. In this paper, we use LSSS matrix (A, ρ) to express an access policy associated to a ciphertext. Due to space limitation, readers may refer to [40] for the algorithmic details.

Cryptographic Primitives. Our construction uses some cryptographic primitives: (1) *Pairings.* Suppose $\mathbb{G}, \mathbb{G}_T$ are cyclic groups of prime order p. Then $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is pairing if $\forall g \in \mathbb{G}_1, \mathfrak{h} \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $\hat{e}(g^a, \mathfrak{h}^b) = \hat{e}(g, \mathfrak{h})^{ab}$ and non-degeneracy. (2) *Target Collision Resistant (TCR) Hash Functions.* Given a random element x which is from the valid domain of a TCR hash function H, no polynomial time adversary A can find $y \neq x$ such that $H(x) = H(y)$. (3) *Secure Signature Scheme.* Denote $\Sigma_{sign} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ as a secure signature scheme. KeyGen outputs a signing key SignKey and a verification key VerifyKey. Sign takes a signing key SignKey and a message m to produce a signature $\tilde{\sigma}$. Verify takes a verification key VerifyKey, a message m and a signature $\tilde{\sigma}$ to output TRUE or FALSE representing a valid signature and invalid signature respectively. We deploy the BLS signature [6] due to its efficiency.

4.2. Main Idea

The main idea of our system is to deploy attribute-based encryption which provides a good platform for a particular set of users (with the same attribute) to securely access some data stored in the cloud server. The sharing functionality can be instantiated by using proxy re-encryption mechanism. The proxy can re-encrypt a ciphertext for another party by an instruction given by the original decrypting party. In this way, the original decrypting party can share the access right with another group of users securely, while the proxy itself knows nothing about the data.

4.3. Detailed Description of Our System

Our system is motivated from [28] and [43]. Detailed step-by-step construction is depicted in the subsequent framed boxes.

Setup. The system first defines some common public parameters. Each CA and AA generates its own public key and secret key by itself. This part contains the three algorithms.

- $\mathsf{GlobalSetup}(\psi) \rightarrow \mathsf{GPK}$. *The algorithm takes as input the security parameter* ψ *and outputs the global public parameter* GPK *of the system. It can be run by any trusted authority.*

Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a type 3 pairing with the base field size q, the embedding degree k and the group size p. The algorithm randomly chooses $g \in \mathbb{G}_1$, $\mathfrak{g}, \mathfrak{u} \in \mathbb{G}_2$, $\iota \in \mathbb{Z}_p$ and sets $h = g^{\iota}, \mathfrak{h} = \mathfrak{g}^{\iota}$. Denote Σ_{sign} as a secure signature scheme. Define the TCR hash functions $H_1 : \{0,1\}^{3\psi} \rightarrow \mathbb{Z}_p^*$, $H_2 : \mathbb{G}_T \rightarrow \{0,1\}^{3\psi}$, $H_3 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_4 : \{0,1\}^{\psi} \rightarrow \mathbb{Z}_p^*$, $H_5 : \{0,1\}^* \rightarrow \mathbb{G}_2$. Let $\zeta : \mathbb{U} \rightarrow \mathbb{K}$ be a function that maps an attribute to the index of the corresponding AA. The global public parameter is published as $\mathsf{GPK} = (g, h, \mathfrak{g}, \mathfrak{h}, \mathfrak{u}, \Sigma_{sign}, H_1, H_2, H_3, H_4, H_5, \zeta)$.

- $\mathsf{CA\text{-}Setup}(\mathsf{GPK}, d) \rightarrow (\mathsf{CPK}_d, \mathsf{CAPK}_d, \mathsf{CMSK}_d)$. *Each* CA_d *generates its own public keys* $(\mathsf{CPK}_d,\ \mathsf{CAPK}_d)$ *and secret keys* CMSK_d.

CA_d runs $\mathsf{KeyGen} \rightarrow (\mathsf{SignKey}_d, \mathsf{VerifyKey}_d)$ and picks a random exponent $\alpha_d \in \mathbb{Z}_p$. CA_d publishes its public parameter $\mathsf{CPK}_d = \hat{e}(g, \mathfrak{g})^{\alpha_d}$, $\mathsf{CAPK}_d = \mathsf{VerifyKey}_d$. CA_d sets its master secret key $\mathsf{CMSK}_d = (\alpha_d, \mathsf{SignKey}_d)$.

- $\mathsf{AA\text{-}Setup}(\mathsf{GPK}, k, \mathbb{U}_k, n_k) \rightarrow (\mathsf{APK}_k, \mathsf{ACPK}_k,\ \mathsf{AMSK}_k)$. *Each AA_k generates its own public keys $(\mathsf{APK}_k, \mathsf{ACPK}_k)$ and secret keys AMSK_k based on attributes under its management $\mathbb{U}_k$. Each user has at most n_k attributes managed by AA_k.*

For each attribute $a \in \mathbb{U}_k$, AA_k picks $s_a \in \mathbb{Z}_p$ and sets $T_a = g^{s_a}, \mathfrak{T}_a = \mathfrak{g}^{s_a}$. For each $d \in \mathbb{D}$, AA_k randomly chooses $v_{k,d} \in \mathbb{Z}_p$ and sets $V_{k,d} = g^{v_{k,d}}$. AA_k publishes its public parameter $\mathsf{APK}_k = \{T_a, \mathfrak{T}_a | a \in \mathbb{U}_k\}, \mathsf{ACPK}_k = \{V_{k,d} | d \in \mathbb{D}\}$. AA_k sets its master secret key $\mathsf{AMSK}_k = (\{s_a | a \in \mathbb{U}_k\}, \{v_{k,d} | d \in \mathbb{D}\})$.

Key Generation. In order to obtain an attribute-based decryption key, each user id contacts all CAs to obtain his partial secret key from each CA. The role of CA is to certify that the user has the correct identity id. Suppose the user is a cardiologist in hospital A. He should obtain a partial secret key with respect to the attribute “Hospital A” from the AA “Hospital Authority”, and should obtain another partial secret key with respect to the attribute “Cardiologist” from the AA “Medical Association”. He does not need to contact AAs that manage attributes unrelated to him. Finally, his decryption key is the combination of all his partial secret keys obtained from CAs and AAs.

This part contains two algorithms: the first one is for a user to obtain key from CA, while the second one is for a user to obtain key for AA.

A Key Generation Scenario. We illustrate the process by which a user obtains his decryption key here. Let’s assume there are three CAs. In practice, these CAs could be the health Department, the state government and the immigration department. Assume user S_{id} who wishes to obtain his key is a cardiologist in hospital A. Firstly, S_{id} sends his identity id to all the CAs. Each CA validates the identity of S_{id}, and issues a user-central-key and user-central-public-key for this user. Then, S_{id} contacts the Hospital Authority to obtain his user-attribute-key for his attribute “Hospital A”. S_{id} submits to the Hospital Authority the three user-central-public-keys he obtained from the three CAs, together with the request that he would like to obtain user-attribute-key for attribute “Hospital A”. The Hospital Authority validates the user-central-public-key and issues the user-attribute-key

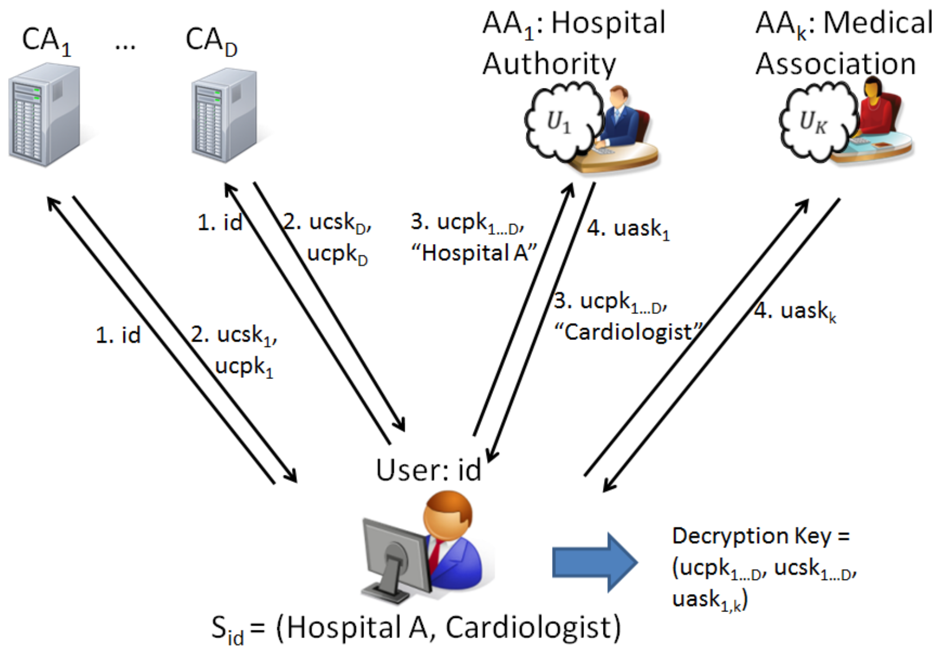
CA_1
...
CA_D
AA_1: Hospital Authority
AA_k: Medical Association
U_1
U_K
1. id
2. $ucsk_D$, $ucpk_D$
3. $ucpk_{1...D}$, "Hospital A"
4. $uask_1$
1. id
2. $ucsk_1$, $ucpk_1$
3. $ucpk_{1...D}$, "Cardiologist"
4. $uask_k$
User: id
Decryption Key = ($ucpk_{1...D}$, $ucsk_{1...D}$, $uask_{1,k}$)
S_{id} = (Hospital A, Cardiologist)

Figure 1: Obtaining keys from CAs and AAs.

for attribute "Hospital A". Likewise, S_{id} contacts the Medical Association following the same procedure to obtain his user-attribute-key for attribute "Cardiologist". Finally, S_{id} combines the obtained user-central-public-keys, user-central-keys, user-attribute-keys to form his decryption key. The process is shown in Figure 1.

- $\mathsf{CA\text{-}KeyGen}(\mathsf{id}, \mathsf{GPK}, \{V_{k,d}|k \in \mathbb{K}\}, \mathsf{CMSK}_d) \rightarrow (\mathsf{ucsk}_{\mathsf{id},d}, \mathsf{ucpk}_{\mathsf{id},d})$. *A user with identifier* id *visits CA_d for obtaining a* **user-central-key** $\mathsf{ucsk}_{\mathsf{id},d}$ *(used as part of his decryption key) and* **user-central-public-key** $\mathsf{ucpk}_{\mathsf{id},d}$ *(used to obtain attributes from AAs for his decryption key).*

When a user submits his id to CA_d to request the user-central-key, CA_d randomly picks $r_{\mathsf{id},d} \in \mathbb{Z}_p$, then sets the user-central-key: $\mathsf{ucsk}_{\mathsf{id},d} = \mathfrak{g}^{\alpha_d}\mathfrak{h}^{r_{\mathsf{id},d}}$, $\mathfrak{L}_{\mathsf{id},d} = \mathfrak{g}^{r_{\mathsf{id},d}}$. For $k \in [1, K]$, CA_d computes $\Gamma_{\mathsf{id},d,k} = \mathfrak{g}^{v_{k,d} \cdot r_{\mathsf{id},d}}$. CA_d computes $\sigma_{\mathsf{id},d} \leftarrow \mathsf{Sign}(\mathsf{SignKey}_d, \mathsf{id}||d||\mathfrak{L}_{\mathsf{id},d}||\{\Gamma_{\mathsf{id},d,k}\}_{k\in[1,K]})$. Let the user-central-public-key $\mathsf{ucpk}_{\mathsf{id},d} = (\mathsf{id}, d, \mathfrak{L}_{\mathsf{id},d}, \{\Gamma_{\mathsf{id},d,k} \mid k \in \mathbb{K}\}, \sigma_{\mathsf{id},d})$.

- $\mathsf{AA\text{-}KeyGen}(a, \{\mathsf{ucpk}_{\mathsf{id},d} | d \in \mathbb{D}\}, \{\mathsf{VerifyKey}_d \, | d \in \mathbb{D}\}, \mathsf{AMSK}_k, \mathsf{GPK}) \rightarrow \mathsf{uask}_{a,\mathsf{id}}$ or $\perp$. *A user with identifier* id *visits AA_k for obtaining a* **user-attribute-key** $\mathsf{uask}_{a,\mathsf{id}}$ *for some attribute a (used as part of his decryption key).*

When a user submits his $\{\mathsf{ucpk}_{\mathsf{id},d} | d \in \mathbb{D}\}$ to AA_k to request the user-attribute-key for attribute $a \in \mathbb{U}_k$, AA_k parses $\mathsf{ucpk}_{\mathsf{id},d}$ into $(\mathsf{id}, d, \mathfrak{L}_{\mathsf{id},d}, \{\Gamma_{\mathsf{id},d,k} | k \in \mathbb{K}\}, \sigma_{\mathsf{id},d})$.

1. For all $d \in [1, D]$, it checks whether $\mathsf{TRUE} \leftarrow \mathsf{Verify}(\mathsf{VerifyKey}_d, \mathsf{id}||d||\mathfrak{L}_{\mathsf{id},d}||\{\Gamma_{\mathsf{id},d,k}\}_{k \in [1,K]}, \sigma_{\mathsf{id},d}) \wedge \hat{e}(g, \Gamma_{\mathsf{id},d,k}) = \hat{e}(V_{k,d}, \mathfrak{L}_{\mathsf{id},d})$. If there is any failure, AA_k outputs $\perp$ to user which means that the submitted $\mathsf{ucpk}_{\mathsf{id},d}$ are invalid.
2. For all $d \in [1, D]$, AA_k outputs the user-attribute-key $\mathsf{uask}_{a,\mathsf{id}}$ to user where

$$\mathsf{uask}_{a,\mathsf{id}} = \prod_{d=1}^{D} (\Gamma_{\mathsf{id},d,k})^{s_a / v_{k,d}} \tag{1}$$

$$= \prod_{d=1}^{D} \mathfrak{g}^{r_{\mathsf{id},d} s_a} = \mathfrak{T}_a^{\sum_{d=1}^{D} r_{\mathsf{id},d}} \tag{2}$$

Without knowing the value of $r_{\mathsf{id},d}$, by running (1), AA_k can compute the value as (2).

After obtaining keys from CAs and AAs, a user id, with attribute set S_{id}, has a ***decryption-key*** defined as $\mathsf{DK}_{\mathsf{id}} = (\{\mathsf{ucsk}_{\mathsf{id},d}, \mathfrak{L}_{\mathsf{id},d} | d \in \mathbb{D}\}, \{\mathsf{uask}_{a,\mathsf{id}} | a \in S_{\mathsf{id}}\})$.

PHR Encryption and Decryption. For each PHR file, it is encrypted with a fine-grained access policy. Only authorized users are allowed to decrypt. The data owner is the patient, Alice. She obtains a secret key from the CAs and her key is assigned with the attribute "Patient ID = Alice" from her hospital (AA). For example, when she obtains her medical test result from a laboratory, the result can be encrypted under the policy "(Hospital A AND Cardiologist) OR Patient ID = Alice OR ER Dept". Therefore, both Alice, her doctor and the emergency department can access her record. The encryption and decryption algorithms are described as follow.

- $\mathsf{PHR\text{-}Encrypt}(M, \mathbb{A}, \mathsf{GPK}, \{\mathsf{APK}_k\}, \{\mathsf{CPK}_d | d \in \mathbb{D}\}) \rightarrow \mathsf{CT}$. *It encrypts a medical record M based on the access policy $\mathbb{A}$ (e.g. "(Hosptial A AND Cardiologist) OR Patient ID = Alice") and stores the resulting ciphertext in the cloud server.*

Suppose $\mathbb{A}$ is the access policy and $M \in \{0,1\}^{2\psi}$ is the message to be encrypted. M is further divided into $M_1 \in \{0,1\}^{\psi}$ which contains the patient's personal information; and $M_2 \in \{0,1\}^{\psi}$ which contains the content of the medical record. Define $\mathbb{A}_1$ and $\mathbb{A}_2$ as the access policy to encrypt M_1 and M_2 respectively. $\mathbb{A}_b$ is expressed by an LSSS matrix (A_b, ρ_b) for $b \in \{1,2\}$, where: (1) A_b is an $l \times n$ matrix and ρ_b maps each row A_x of A_b to an attribute $\rho_b(x)$. It is required that ρ_b will not map two different rows to the same attribute[1]; (2) $\mathbb{A}_b$ must contain the policy "OR Patient ID = id" such that a patient can always access his own PHR; and (3) $\mathbb{A}_b$ must contain the policy "OR ER Dept" such that the emergency department can always access the health record in case of emergency. Both M_1 and M_2 are encrypted using the following algorithm:

For $b \in \{1,2\}$, the encryptor chooses a random $\gamma \in \{0,1\}^{\psi}$ and runs the following steps:

1. The algorithm chooses a random $\beta \in \{0,1\}^{\psi}$ and computes $s = H_1(M_b, \beta, \gamma)$.
2. It sets a vector $\vec{v} = (s, v_2, \ldots, v_n) \in \mathbb{Z}_p^n$ for some random $v_2, \ldots, v_n \in \mathbb{Z}_p$.
3. Let $A_x \cdot \vec{v}$ be the inner product of the x^{th} row of A_b and the vector $\vec{v}$.
4. For each $x \in [1, l]$, it randomly picks $r_x \in \mathbb{Z}_p$ and computes: $C = (M_b || \beta || \gamma) \oplus H_2(\prod_{d=1}^{D} \hat{e}(g, \mathfrak{g})^{\alpha_d \cdot s})$, $C' = g^s$, $\{C_x = h^{A_x \cdot \vec{v}} \cdot T_{\rho_b(x)}^{-r_x},\ C'_x = g^{r_x}\}_{x \in [1,l]}$, $\mathfrak{B} = \mathfrak{u}^s$, $\sigma = H_3(C, \mathfrak{B}, \{C_x, C'_x\}_{x \in [1,l]}, \mathbb{A}_b)^s$.

The ciphertext is $\mathsf{CT}_b = (C, C', \{C_x, C'_x\}_{x \in [1,l]}, \mathfrak{B}, \sigma,\ \mathbb{A}_b)$. Recall that $T_{\rho_b(x)}$ can be calculated from all APK.

Finally the ciphertext $(\mathsf{CT}_1, \mathsf{CT}_2)$ are stored in the cloud server. Note that the value γ is the same for both ciphertexts to ensure that CT_1 and CT_2 are encrypted at the same time.

- $\mathsf{PHR\text{-}Decrypt}(\mathsf{CT}, \mathsf{GPK}, \{\mathsf{APK}_k\}, \mathsf{DK}_{\mathsf{id}}) \to M$. *The user* id *decrypts a ciphertext* CT *in the cloud to retrieve the medical record* M*. If* CT *consists of two parts* CT_1 *and* CT_2*, then each of them undergoes the following algorithms:*

The ciphertext is parsed into $(C, C', \{C_x, C'_x\}_{x\in[1,l]}, \mathfrak{B}, \sigma, \mathbb{A} = (A, \rho))$, and the decryption-key $\mathsf{DK}_{\mathsf{id}}$ is parsed into $(\{\mathsf{ucsk}_{\mathsf{id},d}, \mathfrak{L}_{\mathsf{id},d} | d \in \mathbb{D}\}, \{\mathsf{uask}_{a,\mathsf{id}} | a \in S_{\mathsf{id}}\})$. It outputs $\perp$ if it fails any of the following checking:

$$\begin{aligned} &\hat{e}(C', \mathfrak{u}) = \hat{e}(g, \mathfrak{B}), \\ \wedge \quad &\hat{e}(\sigma, \mathfrak{u}) = \hat{e}(H_3(C, \mathfrak{B}, \{C_x, C'_x\}_{x\in[1,l]}, \mathbb{A}), \mathfrak{B}), \\ \wedge \quad &\hat{e}(C', \mathfrak{h}) = \prod_{\rho(x)\in S_{\mathsf{id}}} \left(\hat{e}(C_x, \mathfrak{g}) \cdot \hat{e}(C'_x, \mathfrak{T}_{\rho(x)})\right)^{\omega_x}. \end{aligned} \tag{3}$$

Otherwise, the algorithm computes

$$\begin{aligned} \mathsf{ucsk}_{\mathsf{id}} &= \prod_{d=1}^{D} \mathsf{ucsk}_{\mathsf{id},d} = \mathfrak{g}^{\sum_{d=1}^{D} \alpha_d} \mathfrak{h}^{\sum_{d=1}^{D} r_{\mathsf{id},d}} = \mathfrak{g}^{\alpha} \mathfrak{h}^{r_{\mathsf{id}}}, \\ \mathfrak{L}_{\mathsf{id}} &= \prod_{d=1}^{D} \mathfrak{L}_{\mathsf{id},d} = \mathfrak{g}^{\sum_{d=1}^{D} r_{\mathsf{id},d}} = \mathfrak{g}^{r_{\mathsf{id}}}, \end{aligned} \tag{4}$$

where $\alpha = \sum_{d=1}^{D} \alpha_d$, $r_{\mathsf{id}} = \sum_{d=1}^{D} r_{\mathsf{id},d}$. Note that $\forall a \in S_{\mathsf{id}}$, $\mathsf{uask}_{a,\mathsf{id}} = \mathfrak{T}_a^{\sum_{d=1}^{D} r_{\mathsf{id},d}} = \mathfrak{T}_a^{r_{\mathsf{id}}}$.
If S_{id} satisfies the access policy (A, ρ), the algorithm computes constants $\omega_x \in \mathbb{Z}_N$ such that $\sum_{\rho(x)\in S_{\mathsf{id}}} \omega_x A_x = (1, 0, \ldots, 0)$. Then it computes $Z = \frac{\hat{e}(C', \mathsf{ucsk}_{\mathsf{id}})}{\prod_{\rho(x)\in S_{\mathsf{id}}} \left(\hat{e}(C_x, \mathfrak{L}_{\mathsf{id}}) \cdot \hat{e}(C'_x, \mathsf{uask}_{\rho(x),\mathsf{id}})\right)^{\omega_x}}$. It computes $M||\beta||\gamma = C \oplus H_2(Z)$.

[1] This restriction is crucial to the security proof. As in [28], we can use encoding technique to remove this restriction.

It outputs M if $C' = g^{H_1(M,\beta,\gamma)}$. If CT consists of two parts, it outputs $\perp$ if the decrypted value γ are different for both parts.

Secure Sharing. There are a few possible sharing methods for our PHR system. A patient may want to share his entire PHR record to his family. A doctor may want to share some medical test results to a researcher for academic purpose, without disclosing any personal information of the patient.

Therefore, our PHR system first classifies a PHR record M into two parts: personal information M_1 may contain the patient's name, home address, etc; and medical record M_2 which does not contain sensitive personal information. They are encrypted by different access policy based on the actual need.

Our PHR system uses proxy re-encryption for secure sharing. The user sends the proxy re-encryption key to the cloud server and asks the server to re-encrypt the desired ciphertext to some new attributes. Using previous example, the cardiologist in Hospital A can access Alice's PHR record. If he wants to share this record with some professors in University B, he can re-encrypt the ciphertext of M_2 to the attribute "University B AND Professor AND Medicine Dept". On the other hand, Alice can re-encrypt the ciphertext of M_1 and M_2 to the attribute "ID = Bob" to her family member Bob. An example of secure sharing is shown in Figure 2.

A Secure Sharing Scenario. We illustrate the process by which a user shares part of his PHR record with another party. Assume a cardiologist in hospital A, S_{id}, would like to share M2 of Alice's PHR with all professors in the medical department of university B. Firstly, S_{id} uses algorithm $\mathsf{ReKeyGen}$ to generate a re-encryption key with $\tilde{\mathbb{A}} = \{``UniversityB'' \wedge ``DeptofMedicine''\}$. The re-encryption key is sent to the sever. Then, the server invokes PHR-ReEnc, using the received re-encryption key, to perform re-encryption on M2 of Alice. After that, the proxy re-encryption of M2 will be sent to the intended recipient(s), i.e., the professor(s) in the medical department of university B. Finally, the professors execute PHR-ReDec using their own decryption keys to obtain M2 of Alice. The process is shown in Figure 2.

- $\mathsf{ReKeyGen}(\mathsf{DK}_{\mathsf{id}}, \tilde{\mathbb{A}}, \mathsf{GPK}) \to \mathsf{RK}$. *A doctor with identifier* id *wants to re-encrypt some medical records to researchers satisfying some*

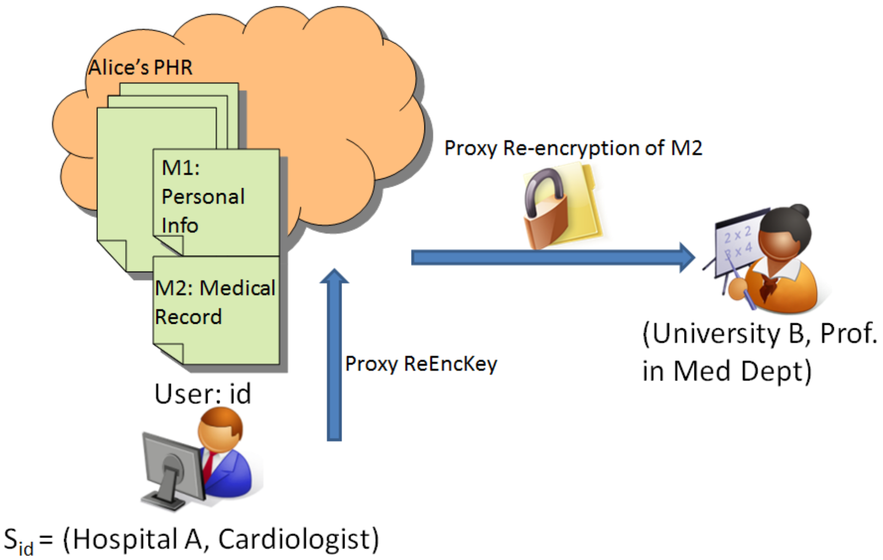
Alice's PHR
M1: Personal Info
M2: Medical Record
Proxy Re-encryption of M2
Proxy ReEncKey
(University B, Prof. in Med Dept)
User: id
S$_{id}$ = (Hospital A, Cardiologist)

Figure 2: Secure Sharing

access policy $\tilde{\mathbb{A}}$ (e.g. "University X AND Dept of Medicine"). It first generates a re-encryption key RK *as follows.*

The decryption-key $\mathsf{DK}_{\mathsf{id}}$ is parsed into $(\{\mathsf{ucsk}_{\mathsf{id},d}, L_{\mathsf{id},d}|d \in \mathbb{D}\}, \{\mathsf{uask}_{a,\mathsf{id}}|a \in S_{\mathsf{id}}\})$, $\tilde{\mathbb{A}}$ is the access policy which is expressed by an LSSS matrix $(\tilde{A}, \tilde{\rho})$, where $\tilde{A}$ is an $\tilde{l} \times \tilde{n}$ matrix and $\tilde{\rho}$ maps each row $\tilde{A}_x$ of $\tilde{A}$ to an attribute $\tilde{\rho}(x)$. The delegator does the followings:

1. The delegator chooses a random $\tilde{\beta}, \delta \in \{0,1\}^{\psi}$ and computes $\tilde{s} = H_1(\delta, \tilde{\beta})$.
2. It sets a vector $\vec{\tilde{v}} = (\tilde{s}, \tilde{v}_2, \ldots, \tilde{v}_n) \in \mathbb{Z}_p^n$ for some random $\tilde{v}_2, \ldots, \tilde{v}_n \in \mathbb{Z}_p$.
3. Let $\tilde{A}_x \cdot \vec{\tilde{v}}$ be the inner product of the x^{th} row of $\tilde{A}$ and the vector $\vec{\tilde{v}}$.
4. For each $x \in [1, \tilde{l}]$, it randomly picks $\tilde{r}_x \in \mathbb{Z}_p$ and computes: $\tilde{C} = (\delta||\tilde{\beta}) \oplus H_2(\prod_{d=1}^{D} \hat{e}(g, \mathfrak{g})^{\alpha_d \cdot \tilde{s}})$, $\tilde{C}' = g^{\tilde{s}}$, $\{\tilde{C}_x = h^{\tilde{A}_x \cdot \vec{\tilde{v}}} \cdot T_{\rho(x)}^{-\tilde{r}_x}$, $\tilde{C}'_x = g^{\tilde{r}_x}\}_{x \in [1, \tilde{l}]}$, $\tilde{\sigma} = H_5(\tilde{C}, \tilde{C}', \{\tilde{C}_x, \tilde{C}'_x\}_{x \in [1,\tilde{l}]}, \tilde{\mathbb{A}})^{\tilde{s}}$. Denote rk_5 as $(\tilde{C}, \tilde{C}', \{\tilde{C}_x, \tilde{C}'_x\}_{x \in [1,\tilde{l}]}, \tilde{\sigma}, \tilde{\mathbb{A}})$.

5. It randomly picks $\theta \in \mathbb{Z}_p$ and sets $\mathsf{rk}_1 = \prod_{d=1}^{D} \mathsf{ucsk}_{\mathsf{id},d}^{H_4(\delta)} \cdot \mathfrak{g}_1^{\theta}$, $\mathsf{rk}_2 = g^{\theta}$, $\mathsf{rk}_3 = \prod_{d=1}^{D} \mathfrak{L}_{\mathsf{id},d}^{H_4(\delta)}$, $\mathsf{rk}_{4,a} = \mathsf{usak}_{a,\mathsf{id}}^{H_4(\delta)}\ \forall a \in S_{\mathsf{id}}$.
6. It outputs the re-encryption key $\mathsf{RK} = (\mathsf{rk}_1, \mathsf{rk}_2, \mathsf{rk}_3, \mathsf{rk}_5, \{\mathsf{rk}_{4,a}\}, S_{\mathsf{id}})$

Note that the patient can forward his PHR to his carers in a similar way.

- $\mathsf{PHR\text{-}ReEnc}(\mathsf{RK}, \mathsf{CT}) \to \mathsf{CT}_R$. *If the doctor trusts the cloud server on forwarding the correct ciphertext* CT*, he can send the re-encryption key* RK *to the server; and the server outputs the re-encrypted ciphertext* CT_R *to the researcher (In this case, the cloud server is semi-trusted: The honest-but-curious server is trusted to perform the correct re-encryption, but it cannot access the content of the PHR.) If the doctor does not trust the cloud server, he can do the re-encryption by himself.*

  For a ciphertext $\mathsf{CT} = (\mathsf{CT}_1, \mathsf{CT}_2)$*, the algorithm should only re-encrypt* CT_2 *(the medical record without personal information) if it is used for research purpose. If a patient wants to share his PHR with his family, he can re-encrypt both* CT_1 *and* CT_2*.*

The re-encryption key RK is parsed into $(\mathsf{rk}_1, \mathsf{rk}_2, \mathsf{rk}_3, \mathsf{rk}_5, \{\mathsf{rk}_{4,a}\}, S_{\mathsf{id}})$ and the ciphertext is parsed into $(C, C', \{C_x, C'_x\}_{x \in [1,l]}, \mathfrak{B}, \sigma, \mathbb{A} = (A, \rho))$. It outputs $\perp$ if it fails any of the following checking:

- S_{id} does not satisfies $\mathbb{A}$;
- the ciphertext fails the validity check of equation 3;
- the re-encryption key $\mathsf{rk}_5 = (\tilde{C}, \tilde{C}', \{\tilde{C}_x, \tilde{C}'_x\}_{x \in [1,\tilde{l}]}, \tilde{\sigma}, \tilde{\mathbb{A}})$ is invalid when: $\hat{e}(g, \tilde{\sigma}) \neq \hat{e}(\tilde{C}', H_5(\tilde{C}, \tilde{C}', \{\tilde{C}_x, \tilde{C}'_x\}_{x \in [1,\tilde{l}]}, \tilde{\mathbb{A}}))$.

Otherwise, it computes constants $\omega_x \in \mathbb{Z}_p$ such that $\sum_{\rho(x) \in S_{\mathsf{id}}} \omega_x A_x = (1, 0, \ldots, 0)$. It calculates

$$B' = \frac{\hat{e}(C', \mathsf{rk}_1)}{\hat{e}(\mathsf{rk}_2, \mathfrak{B}) \cdot \prod_{\rho(x) \in S_{\mathsf{id}}} (\hat{e}(C_x, \mathsf{rk}_3) \cdot \hat{e}(C'_x, \mathsf{rk}_{4,x}))^{\omega_x}}. \tag{5}$$

It outputs the re-encrypted ciphertext $\mathsf{CT}_R = (C, \{C_x, C'_x\}_{x\in[1,l]}, \mathfrak{B}, \sigma, \mathsf{rk}_5, B', S_{\mathsf{id}}, \mathbb{A})$.

- $\mathsf{PHR\text{-}ReDec}(\mathsf{CT}_R, \mathsf{GPK}, \{\mathsf{APK}_k\}, \mathsf{DK}_{\mathsf{id}'}) \rightarrow M$. *When the researcher* id' *receives a re-encrypted ciphertext* CT_R*, he runs the re-decryption algorithm to obtain the PHR record* M *using his own decryption key* $\mathsf{DK}_{\mathsf{id}'}$*.*

The re-encrypted ciphertext CT_R is parsed into $(C, \{C_x, C'_x\}_{x\in[1,l]}, \mathfrak{B}, \sigma, \mathsf{rk}_5, B', S_{\mathsf{id}}, \mathbb{A})$, and the decryption key $\mathsf{DK}_{\mathsf{id}'}$ is parsed into $(\{\mathsf{ucsk}_{\mathsf{id}',d}, \mathsf{ucpk}_{\mathsf{id}',d} | d \in \mathbb{D}\}, \{\mathsf{uask}_{a,\mathsf{id}'} | a \in S_{\mathsf{id}'}\})$.

1. It parses rk_5 as $(\tilde{C}, \tilde{C}', \{\tilde{C}_x, \tilde{C}'_x\}_{x\in[1,\tilde{l}]}, \tilde{\sigma}, \tilde{\mathbb{A}} = (\tilde{A}, \tilde{\rho}))$. It outputs $\perp$ if $S_{\mathsf{id}'}$ does not satisfies $\tilde{\mathbb{A}}$; or $\hat{e}(g, \tilde{\sigma}) \neq \hat{e}(\tilde{C}', H_3(\tilde{C}, \tilde{C}', \{\tilde{C}_x, \tilde{C}'_x\}_{x\in[1,\tilde{l}]}, \tilde{\mathbb{A}}))$. Otherwise, it computes $\mathsf{ucsk}_{\mathsf{id}'}$ and $L_{\mathsf{id}'}$ as in equation 4. It computes constants $\omega_x \in \mathbb{Z}_N$ such that $\sum_{\tilde{\rho}(x)\in S_{\mathsf{id}'}} \omega_x \tilde{A}_x = (1, 0, \ldots, 0)$. Then it computes $Z = \frac{\hat{e}(\tilde{C}', \mathsf{ucsk}_{\mathsf{id}'})}{\prod_{\tilde{\rho}(x)\in S_{\mathsf{id}'}} \left(\hat{e}(\tilde{C}_x, L_{\mathsf{id}'})\cdot\hat{e}(\tilde{C}'_x, \mathsf{uask}_{\tilde{\rho}(x),\mathsf{id}'})\right)^{\omega_x}}$. It computes $\delta || \tilde{\beta} = H_2(Z) \oplus \tilde{C}$. It outputs $\perp$ if $\tilde{C}' \neq g^{H_1(\delta, \tilde{\beta})}$.
2. It calculates $M || \beta || \gamma = H_2(B'^{\frac{1}{H_4(\delta)}}) \oplus C$. It outputs $\perp$ if S_{id} does not satisfies $\mathbb{A}$; or $B \neq u^{H_1(M,\beta)}$; or $\sigma \neq H_3(C, B, \{C_x, C'_x\}_{x\in[1,l]}, \mathbb{A})^{H_1(M,\beta,\gamma)}$. Otherwise, it outputs the message M.
3. If the re-encrypted ciphertext has two parts and the decrypted value γ are different, output $\perp$. Otherwise, output both messages.

Break-glass. To ensure break-glass access of PHR files in case of emergency, our system requires that all PHR files must be encrypted with the policy "OR ER Dept". Therefore, the emergency department, having the valid attribute-based key, can access the patient's PHR files. To avoid the abuse of the break-glass option, the emergency staff needs to verify his identity and the emergency situation. Then the staff will obtain the re-encrypted ciphertext of the patient's PHR records.

User Revocation. We consider the case of user revocation in our PHR system. In addition to attributes, users also obtain keys related to the time domain. Ciphertext are updated according to the time when there is user revocation. A revoked user cannot obtain the updated key related to time. Therefore, he cannot access the PHR files anymore. The advantage of our proposed construction is that the cloud server does not need any secret information when performing ciphertext update, and it does not need to know who is revoked.

Sahai *et al.* [36] proposed a generic construction to allow a third party (cloud server) to disqualify revoked users from accessing data that was encrypted in the past. This process can be done without access to any sensitive information by the cloud server. At the same time, newly encrypted data is not decryptable by the revoked user's key. Roughly speaking, they defined *piecewise CP-ABE* in which each user has two keys with similar structure. One key is associated with the user's attribute, while the other key is associated with time. A ciphertext can be decrypted only by using two valid keys together. A revoked user cannot update the key associated with time. The cloud server only needs to re-encrypt the ciphertext according to time if necessary. Note that the cloud server does not need any secret information when performing ciphertext update, and it does not need to know who is revoked. Our proposed scheme is compatible to the piecewise CP-ABE and hence can be extended for key revocation.

4.4. Using Hybrid Encryption

As a PHR document is sometimes very large (e.g. includes X ray images), hybrid encryption can be used to improve the efficiency: Instead of encrypting the PHR itself, PHR-Encrypt and PHR-ReEnc encrypt a randomly generated key which is used to symmetric encrypt the PHR document. In this way, the size of the "original message" (which is now actually the symmetric key) is much smaller and the encryption (and decryption) algorithm will be more efficient.

4.5. Security Analysis

We analyze the security of our proposed system. First we show that it achieves data confidentiality. Our scheme is selectively secure against chosen ciphertext attack (CCA), for both original ciphertext and re-encrypted ciphertext. In short, our security model protects the confidentiality of PHR encrypted for the access policy $\mathbb{A}$ even if: (1) only one CA is honest, other CAs are corrupted, (2) all AAs are corrupted, and (3) all users not satisfying $\mathbb{A}$ are corrupted. The strong *chosen ciphertext security* guarantees confidentiality even if the adversary obtains decryption of other ciphertexts for the access policy $\mathbb{A}$.

User revocation can be easily seen in our scheme description. Secure sharing is realized as our encryption scheme achieves CCA security against re-encrypted ciphertext. That is, no unauthorized party including the cloud server can decrypt the ciphertext. Emergency access can be facilitated as "OR ER Dept" is always included in the policy so that ER Dept can always decrypt any PHR document for emergency situation. If the patient accidentally forgets to include the emergency department, it is noticeable from the ciphertext since the access policy is included as part of the ciphertext.

We can prove our scheme in the even stronger *adpative* security model if we extend our scheme to the composite order bilinear groups, and employ the dual system encryption [39] in the security proof, which is similar to the proof in [28]. We give the selective-attribute secure scheme in this paper because of its simplicity and the ease of understanding.

Selective-Attribute CCA Security Model for Original Ciphertext. The security model is defined by the following game run between a challenger $\mathcal{B}$ and an adversary $\mathcal{A}$. $\mathcal{A}$ can corrupt CAs and AAs by specifying $\mathbb{K}_c \subset \mathbb{K}$ and $\mathbb{D}_c \subset \mathbb{D}$ after seeing the public parameters[2], where $\mathbb{D} \setminus \mathbb{D}_c \neq \emptyset$ and $\mathbb{K} \setminus \mathbb{K}_c \neq \emptyset$.

Setup: The challenger $\mathcal{B}$ runs $\mathsf{GlobalSetup}(\psi)$, $\mathsf{CASetup}(\mathsf{GPK}, d)$ $(\forall d \in [1, D])$ and $\mathsf{AASetup}(\mathsf{GPK}, k, U_k)$ $(\forall k \in [1, K])$. The values $\mathsf{GPK}, \{\mathsf{CPK}_d, \mathsf{CAPK}_d | d \in \mathbb{D}\}$ and $\{\mathsf{APK}_k, \mathsf{ACPK}_k | k \in \mathbb{K}\}$ are given to the adversary $\mathcal{A}$. $\mathcal{A}$ specifies index sets $\mathbb{D}_c \subset \mathbb{D}$ and $\mathbb{K}_c \subset \mathbb{K}$ as the corrupted CAs and AAs respectively, where $\mathbb{D} \setminus \mathbb{D}_c \neq \emptyset$ and $\mathbb{K} \setminus \mathbb{K}_c \neq \emptyset$. $\mathcal{A}$ specifies an access policy $\mathbb{A}^*$. Then $\mathcal{B}$ gives $\mathbb{A}^*$, $\{\mathsf{CMSK}_d | d \in \mathbb{D}_c\}$ and $\{\mathsf{AMSK}_k | k \in \mathbb{K}_c\}$ to $\mathcal{A}$.

Query Phase 1: The adversary can query the following oracles: **(i)** $\mathsf{CKQ}(\mathsf{id}, d)$ where $d \in \mathbb{D} \setminus \mathbb{D}_c$: (User-central-key Query) $\mathcal{A}$ queries with a pair (id, d), where id is an identifier and $d \in \mathbb{D} \setminus \mathbb{D}_c$ is the index of an uncorrupted

[2] This is stronger than static corruption model used in [8, 9], where the adversary has to specify the authorities to corrupt before seeing the public parameters. But on the other aspect, it is weaker than the model in [20], where corrupted authorities are set by the adversary.

CA, and obtains the corresponding user-central-key $(\mathsf{ucsk}_{\mathsf{id},d}, \mathsf{ucpk}_{\mathsf{id},d})$. **(ii)** $\mathsf{AKQ}(a, k, \{\mathsf{ucpk}_{\mathsf{id},d} | d \in \mathbb{D}\})$ where $k \in \mathbb{K} \setminus \mathbb{K}_c$: (User-attribute-key Query) $\mathcal{A}$ queries with $(a, k, \{\mathsf{ucpk}_{\mathsf{id},d} | d \in \mathbb{D}\})$, where $k \in \mathbb{K} \setminus \mathbb{K}_c$ is the index of an uncorrupted AA, $\{\mathsf{ucpk}_{\mathsf{id},d} | d \in \mathbb{D}\}$ are id's user-central-public-keys, and a is an attribute in $\mathbb{U}_k$. The oracle returns a user-attribute-key $\mathsf{uask}_{a,\mathsf{id}}$ or $\perp$ if $\{\mathsf{ucpk}_{\mathsf{id},d}\}$ are invalid. **(iii)** $\mathsf{RKQ}(S_{\mathsf{id}}, \tilde{\mathbb{A}})$: (Re-encryption-key Query) $\mathcal{A}$ queries with the user attributes S_{id}, and an access structure $\tilde{\mathbb{A}}$, and obtains the re-encryption key $\mathsf{RK} \leftarrow \mathsf{ReKeyGen}(\mathsf{DK}_{\mathsf{id}}, \tilde{\mathbb{A}}, \mathsf{GPK})$, where $\mathsf{DK}_{\mathsf{id}}$ is the decryption key for S_{id} computed from the CAs and AAs. **(iv)** $\mathsf{REQ}(S_{\mathsf{id}}, \tilde{\mathbb{A}}, \mathsf{CT}_{\mathbb{A}})$: (Re-encryption Query) $\mathcal{A}$ queries with the user attributes S_{id}, an access structure $\tilde{\mathbb{A}}$ and a ciphertext $\mathsf{CT}_{\mathbb{A}}$ for the access structure $\mathbb{A}$. Denote the re-encryption key $\mathsf{RK} \leftarrow \mathsf{ReKeyGen}(\mathsf{DK}_{\mathsf{id}}, \tilde{\mathbb{A}}, \mathsf{GPK})$, where $\mathsf{DK}_{\mathsf{id}}$ is the decryption key for S_{id} computed from the CAs and AAs. $\mathcal{A}$ obtains the re-encrypted ciphertext $\mathsf{CT}_R \leftarrow \mathsf{ReKeyGen}(\mathsf{DK}_{\mathsf{id}}, \tilde{\mathbb{A}}, \mathsf{GPK})$ or the $\perp$ symbol for failure. **(v)** $\mathsf{DQ}(\mathsf{CT}, S_{\mathsf{id}})$: (Decryption Query) $\mathcal{A}$ queries with a ciphertext CT, the decryptor's set of attributes S_{id}, and obtains the plaintext $M \leftarrow \mathsf{PHR}-\mathsf{Decrypt}(\mathsf{CT}, \mathsf{GPK}, \{\mathsf{APK}_k\}, \mathsf{DK}_{\mathsf{id}})$ or the $\perp$ symbol for failure, where $\mathsf{DK}_{\mathsf{id}}$ is the decryption key for S_{id} computed from the CAs and AAs. **(vi)** $\mathsf{RDQ}(\mathsf{CT}_R, S_{\mathsf{id}'})$: (Re-Decryption Query) $\mathcal{A}$ queries with a ciphertext CT_R, the decryptor's set of attributes $S_{\mathsf{id}'}$, and obtains the corresponding plaintext $M \leftarrow \mathsf{PHR}-\mathsf{ReDecrypt}(\mathsf{CT}_R, \mathsf{GPK}, \{\mathsf{APK}_k\}, \mathsf{DK}_{\mathsf{id}'})$ or the $\perp$ symbol for failure, where $\mathsf{DK}_{\mathsf{id}'}$ is the decryption key for $S_{\mathsf{id}'}$ computed from the CAs and AAs.
Challenge Phase: $\mathcal{A}$ submits two equal-length messages M_0^*, M_1^*. $\mathcal{B}$ flips a random coin $\beta \in \{0, 1\}$ and sends a ciphertext CT^* to $\mathcal{A}$ an encryption of M_β^* under $\mathbb{A}^*$.
Query Phase 2: $\mathcal{A}$ further queries as in **Query Phase 1**.
Guess: $\mathcal{A}$ submits a guess β' for β.

For a fixed id, the related attribute set is defined as $A_{\mathsf{id}} = \{a \mid \mathsf{AKQ}(a, k, \{\mathsf{ucpk}_{\mathsf{id},d} | d \in \mathbb{D}\})$ is made by $\mathcal{A}\}$. $\mathcal{A}$ wins the game if $\beta' = \beta$ under the *restriction* that

1. there is no A_{id} such that $A_{\mathsf{id}} \cup (\bigcup_{k_c \in \mathbb{K}_c} \mathbb{U}_{k_c})$ can satisfy the challenge access policy $\mathbb{A}^*$;
2. there is no DQ on CT^* with input attributes S_{id} satisfying $\mathbb{A}^*$;
3. there is no $A_{\mathsf{id}'}$ such that $A_{\mathsf{id}'} \cup (\bigcup_{k_c \in \mathbb{K}_c} \mathbb{U}_{k_c})$ can satisfy some access policy $\tilde{\mathbb{A}}$ where $\mathsf{RKQ}(S_{\mathsf{id}}, \tilde{\mathbb{A}})$ or $\mathsf{REQ}(S_{\mathsf{id}}, \tilde{\mathbb{A}}, \mathsf{CT}^*)$ was queried and $S_{\mathsf{id}} \cup (\bigcup_{k_c \in \mathbb{K}_c} \mathbb{U}_{k_c})$ can satisfy the challenge access policy $\mathbb{A}^*$;

4. there is no RDQ for any $\mathsf{CT}_{\mathbb{A}'}$ with input attributes $S_{\mathsf{id}'}$ satisfying $\mathbb{A}'$, where $\mathsf{CT}_{\mathbb{A}'}$ is a derivative of CT^*. Derived from the definition of derivative for proxy re-encryption [7], the derivative of CT^* is defined as follows: **(i)** CT^* is a derivative of itself. **(ii)** If $\mathcal{A}$ has issued $\mathsf{RKQ}(S_{\mathsf{id}}, \mathbb{A}')$ and obtains the re-encryption key RK, $S_{\mathsf{id}} \in \mathbb{A}^*$, and $\mathsf{CT}_{\mathbb{A}'} \leftarrow \mathsf{PHR} - \mathsf{ReEnc}(\mathsf{RK}, \mathsf{CT}^*)$, then $\mathsf{CT}_{\mathbb{A}'}$ is a derivative of CT^*. **(iii)** If $\mathcal{A}$ has issued $\mathsf{REQ}(S_{\mathsf{id}}, \mathbb{A}', \mathsf{CT}^*)$ with $S_{\mathsf{id}} \in \mathbb{A}^*$ and obtains the re-encrypted ciphertext $\mathsf{CT}_{\mathbb{A}'}$, then $\mathsf{CT}_{\mathbb{A}'}$ is a derivative of CT^*.

The advantage of $\mathcal{A}$ is defined as $|\Pr[\beta = \beta'] - 1/2|$.

Definition 3. *A system is selective-attribute secure against IND-CCA-Or if for all polynomial-time adversary $\mathcal{A}$ in the game above, the advantage of $\mathcal{A}$ is negligible.*

Remarks: (1) We assume that a user with identifier id requests for the central key from each CA_d only once, i.e., for each id there is only one set of user-central-keys, $\{\mathsf{ucpk}_{\mathsf{id},d} | d \in \mathbb{D}\}$. This is not a restriction, but can help simplify the system description. Using a timestamp as another index for ucpk can remove this assumption. (2) In the security model above, $\mathcal{A}$ has the master secret keys $\{\mathsf{CMSK}_d | d \in \mathbb{D}_c\}$, so the user only needs to query $\mathsf{CKQ}(\mathsf{id}, d)$ for $d \in \mathbb{D} \setminus \mathbb{D}_c$ to get $(\mathsf{ucsk}_{\mathsf{id},d}, \mathsf{ucpk}_{\mathsf{id},d})$, and the user can generate $\{(\mathsf{ucsk}_{\mathsf{id},d}, \mathsf{ucpk}_{\mathsf{id},d}) | d \in \mathbb{D}_c\}$ if they are needed for querying AKQ. (3) The model above can be converted to adaptive-attribute security by allowing $\mathcal{A}$ to output the challenge access policy $\mathbb{A}^*$ in the challenge phase instead.

Selective-Attribute CCA Security Model for Re-encrypted Ciphertext. The security definition for the re-encrypted ciphertext can be defined similarly.

1. **Setup.** Same as the previous game.
2. **Query Phase 1.** Same as the previous game.
3. **Challenge.** $\mathcal{A}$ submits two equal-length messages M_0^*, M_1^* and a set of attributes S satisfying an access policy $\mathbb{A}$. $\mathcal{B}$ flips a random coin $\beta \in \{0, 1\}$ and calculates a ciphertext CT^* which is the encryption of M_β^* under $\mathbb{A}$. $\mathcal{B}$ calculates the re-encryption key RK from S to the challenge access policy $\mathbb{A}^*$ and returns the re-encrypted challenge ciphertext CT_R^* computed from $\mathsf{PHR}\text{-}\mathsf{ReEnc}$ using RK and CT^*. $\mathcal{B}$
4. **Query Phase 2.** Same as the previous game.

5. **Guess.** Same as the previous game.

$\mathcal{A}$ wins the game if $\beta' = \beta$ under the same *restriction* (for AKQ and RKQ) except that: (1) there is no restriction on DQ query; (2) there is no RDQ query on CT^*_R.

Security Proofs. We first introduce the assumption used in the theorem. It is modified from the decisional q-parallel BDHE assumption [40], for using it in the asymmetric pairing setting.

Definition 4. Asymmetric Decisional q-parallel BDHE Assumption. *Given a bilinear group* $\mathsf{BG} = (p, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$*, define a tuple* $\mathsf{y} = g, g^a, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}}, \mathfrak{g}, \mathfrak{g}^s, \mathfrak{g}^a, \ldots, \mathfrak{g}^{a^q}, \forall_{1\leq j\leq q}\ g^{s\cdot b_j}, g^{a/b_j}, \ldots, g^{a^q/b_j}, \forall_{1\leq j\leq q}\ \mathfrak{g}^{s\cdot b_j}, \mathfrak{g}^{a/b_j}, \ldots, \mathfrak{g}^{a^q/b_j}, \mathfrak{g}^{a^{q+2}/b_j}, \ldots, \mathfrak{g}^{a^{2q}/b_j}, \forall_{1\leq j,k\leq q,k\neq j}\ g^{a\cdot s\cdot b_k/b_j}, \ldots, g^{a^q\cdot s\cdot b_k/b_j}$*, where* $a, s, b_1, \ldots, b_q \in_R \mathbb{Z}_p$*,* g *is a generator of* $\mathbb{G}_1$ *and* $\mathfrak{g}$ *is a generator of* $\mathbb{G}_2$*. Denote* $T_0 = \hat{e}(g, \mathfrak{g})^{a^{q+1}\cdot s}$ *and* $T_1 \in_R \mathbb{G}_T$*. The Asymmetric Decisional q-parallel BDHE problem is to give* (BG, y, T_b) *to an adversary* $\mathcal{A}$*, and* $\mathcal{A}$ *decides if* $b = 0/1$*. Define the advantage of* $\mathcal{A}$ *as* $|\Pr[\mathcal{A}(\mathsf{BG}, \mathsf{y}, T_b) = b] - \frac{1}{2}|$*. The Asymmetric Decisional q-parallel BDHE Assumption holds in* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ *if no probabilistic polynomial time algorithm has non-negligible advantage.*

As discussed in [43], the re-randomizability guarantee (needed for delegation) and the CCA2 security seems to be contradictory. We also use the similar approach as in [43] to deal with this problem: we prevent adversaries from modifying the message or the access structure by signing just on partial ciphertext components (not signing on C'); re-encryption is just limited to updating partial ciphertext components. Based on this observation, we adopt the same technique as [43] by just sign on partial ciphertext components. We now give the security proofs for the original ciphertext and the re-encrypted ciphertext.

Theorem 1. *Our scheme is selective-attribute secure against IND-CCA-Or if the asymmetric decisional q-parallel BDHE assumption holds in the random oracle model.*

Proof: Suppose there exists an adversary $\mathcal{A}$ who can break the IND-CCA-Or security of our scheme. Given the decisional q-parallel BDHE problem instance $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, y, T)$, we construct an algorithm $\mathcal{C}$ to decide whether $T = \hat{e}(g, \mathfrak{g})^{a^{q+1}s}$ or not.

Setup. $\mathcal{C}$ chooses a random $\zeta \in \mathbb{Z}_p^*$ and sets $h = g^a$, $\mathfrak{h} = \mathfrak{g}^a$, $\mathfrak{u} = \mathfrak{g}^\zeta$ $\mathcal{C}$ sets the rest of GPK as in $\mathsf{GlobalSetup}$ and sends GPK to $\mathcal{A}$.

$\mathcal{A}$ returns the challenge access structure (M^*, ρ^*) to $\mathcal{C}$, where M^* is an $l^* \times n^*$ matrix, $l^*, n^* \leq q$.

$\mathcal{C}$ chooses some random $\alpha'_d \in \mathbb{Z}_p^*$ and sets $\mathsf{CPK}_d = \hat{e}(g^a, \mathfrak{g}^{a^q}) \cdot \hat{e}(g, \mathfrak{g})^{\alpha'_d}$. It implicitly sets $\alpha_d = a^{q+1} + \alpha'_d$. It runs $(\mathsf{SignKey}_d, \mathsf{VerifyKey}_d) \leftarrow \mathsf{KeyGen}$ and sets $\mathsf{CAPK}_d = \mathsf{VerifyKey}_d$.

$\mathcal{C}$ randomly chooses $z_x \in \mathbb{Z}_p$ and computes APK_k as (for all $x \in \mathbb{U}_k$): $T_x = g^{z_x} \cdot \prod_{i \in X} g^{\frac{aM^*_{i,1}}{b_i} + \frac{a^2 M^*_{i,2}}{b_i} + \cdots + \frac{a^{n^*} M^*_{i,n^*}}{b_i}}$, $\mathfrak{T}_x = \mathfrak{g}^{z_x} \cdot \prod_{i \in X} \mathfrak{g}^{\frac{aM^*_{i,1}}{b_i} + \frac{a^2 M^*_{i,2}}{b_i} + \cdots + \frac{a^{n^*} M^*_{i,n^*}}{b_i}}$, where X denote the set of indices i such that $\rho^*(i) = x$, for $i \in [1, l^*]$. $\mathcal{C}$ randomly chooses $v_{k,d} \in \mathbb{Z}_p$ and computes ACPK_k as $V_{k,d} = g^{v_{k,d}}$. $\mathcal{C}$ sends $\{\mathsf{CPK}_d, \mathsf{CAPK}_d\}_{d \in \mathbb{D}}, \{\mathsf{APK}_k, \mathsf{ACPK}_k\}_{k \in \mathbb{K}})$ to $\mathcal{A}$.

$\mathcal{C}$ also maintains some lists which are initially empty: (1) $H_1^{List}, \ldots, H_5^{List}$, $UASK^{List}$ stores the tuple (input; output; tag) for the oracle queries for $H_1, \ldots, H_6, \mathsf{AKQ}$ respectively. (2) RK^{List} stores the tuple (input; output; tag) for the oracle queries for RKQ, where tag consists of $\delta, \widetilde{\beta}$ and three bits b_1, b_2, b_3, denote that the re-encryption key is randomly chosen, generated by REQ or RKQ respectively. (3) RE^{List} stores the tuple (input; output; tag) for the oracle queries for REQ, where tag consists of three bits $b_1||b_2||b_3$, denote that the re-encrypted ciphertext is generated under a valid re-encryption key, under a randomly chosen re-encryption key, or generated without any re-encryption key respectively.

$\mathsf{Query\ Phase\ 1}$. $\mathcal{C}$ answers the queries to the oracles as follows: (1) H_1, H_2, H_4: For every query, if there is an entry (input; output; $\perp$) in the corresponding H list, $\mathcal{C}$ returns the corresponding output. Otherwise, $\mathcal{C}$ returns a random output in the range and stores (input; output; $\perp$) in the corresponding H list. (2) H_3: For every query, if there is an entry (input; output; $\cdot$) in H_3^{List}, $\mathcal{C}$ returns the corresponding output. Otherwise, $\mathcal{C}$ picks a random $\xi_1 \in \mathbb{Z}_p$, returns g^{ξ_1} and stores (input; g^{ξ_1}; ξ_1) in H_3^{List}. (3) H_5: For every query, if there is an entry (input; output; $\cdot$) in H_5^{List}, $\mathcal{C}$ returns the corresponding output. Otherwise, $\mathcal{C}$ picks a random $\xi_2 \in \mathbb{Z}_p$, returns $\mathfrak{g}^{\xi_2}$ and stores (input; $\mathfrak{g}^{\xi_2}$; ξ_2) in H_5^{List}. (4) CKQ: On input id and $d \in \mathbb{D} \setminus \mathbb{D}_c$, $\mathcal{C}$ randomly chooses $r'_{\mathsf{id},d} \in \mathbb{Z}_p$, $(w_{1,d}, w_{2,d}, \ldots, w_{n^*,d}) \in \mathbb{Z}_p^{n^*}$ such that $w_{1,d} = -1$ and $\sum_{j=1}^{n^*} w_{j,d} \cdot M^*_{i,j} = 0$ for all i[3]. $\mathcal{C}$ computes: $\mathfrak{L}_{\mathsf{id},d} =$

[3] Such a vector $(w_{1,d}, w_{2,d}, \ldots, w_{n^*,d})$ must exist by the convention of LSSS.

$\mathfrak{g}^{r'_{\mathsf{id},d}+w_{1,d}a^q+\ldots+w_{n^*,d}a^{q-n^*+1}}$, $\Gamma_{\mathsf{id},d,k}=\mathfrak{L}_{\mathsf{id},d}^{v_{k,d}}$, $\mathsf{ucsk}_{\mathsf{id},d}=\mathfrak{g}^{\alpha'_d}\cdot\mathfrak{g}^{ar'_{\mathsf{id},d}}\cdot\prod_{i=2}^{n^*}\mathfrak{g}^{a^{q+2-i}\cdot w_{i,d}}$ $=\mathfrak{g}^{\alpha_d}(\mathfrak{g}^a)^{r_{\mathsf{id},d}+w_{1,d}a^q+\ldots+w_{n^*,d}a^{q-n^*+1}}$. $\mathcal{C}$ generates the signature $\sigma_{\mathsf{id},d}$ and returns $\mathsf{ucsk}_{\mathsf{id},d}, \mathsf{ucpk}_{\mathsf{id},d}=(\mathsf{id},d,\mathfrak{L}_{\mathsf{id},d},\{\Gamma_{\mathsf{id},d,k}|k\in\mathbb{K}\},\sigma_{\mathsf{id},d})$. (5) AKQ: On input a, $k\in\mathbb{K}\setminus\mathbb{K}_c$ and $\mathsf{ucpk}_{\mathsf{id},d}$ for $d\in\mathbb{D}$, $\mathcal{C}$ first verifies the signature in $\mathsf{ucpk}_{\mathsf{id},d}$ as in AA-KeyGen. If it passes the verification, $\mathcal{C}$ computes:

$$\begin{aligned}
\mathsf{uask}_{a,\mathsf{id}} &= \mathfrak{T}_a^{z_a}\cdot\prod_{d=1}^{D}\prod_{i\in X}\prod_{j=1}^{n^*}\left(\mathfrak{g}^{\frac{a^j}{b_i}\cdot r'_{\mathsf{id},d}}\cdot\prod_{y=1,y\neq j}^{n^*}\mathfrak{g}^{\frac{a^{q+1+j-y}}{b_i}\cdot w_{y,d}}\right)^{M^*_{i,j}}\\
&= \mathfrak{T}_a^{z_a}\cdot\prod_{d=1}^{D}\mathfrak{g}^{(r'_{\mathsf{id},d}+\sum_{y=1}^{n^*}w_{y,d}a^{q-y+1})\cdot\sum_{i\in X}\sum_{j=1}^{n^*}\frac{a^jM^*_{i,j}}{b_i}}\\
&= \prod_{d=1}^{D}\mathfrak{g}^{(r'_{\mathsf{id},d}+\sum\limits_{y=1}^{n^*}w_{y,d}a^{q-y+1})(z_a+\sum\limits_{i\in X}\frac{aM^*_{i,1}}{b_i}+\cdots+\frac{a^{n^*}M^*_{i,n^*}}{b_i})}\\
&= \mathfrak{g}^{\sum_{d=1}^{D}r_{\mathsf{id},d}s_a}
\end{aligned}$$

Recall that since $\sum_{j=1}^{n^*}w_{j,d}\cdot M^*_{i,j}=0$, the terms for $y=j$ is canceled out in the above equation. (6) RKQ: On input $S_{\mathsf{id}},\tilde{\mathbb{A}}$, if $((S_{\mathsf{id}},\tilde{\mathbb{A}});\mathsf{RK};(\delta,\tilde{\beta},\cdot,0,1))\in RK^{List}$, $\mathcal{C}$ returns RK to $\mathcal{A}$. Otherwise:

- If $S_{\mathsf{id}}\in\mathbb{A}^*$ and $(S',\mathsf{uask})\in UASK^{List}$ (for any $S'\in\tilde{\mathbb{A}}$), $\mathcal{C}$ aborts the simulation (due to the restriction of the security model).

- If $S_{\mathsf{id}}\in\mathbb{A}^*$ and $(S',\mathsf{uask})\notin UASK^{List}$ (for all $S'\in\tilde{\mathbb{A}}$), $\mathcal{C}$ checks if $((S_{\mathsf{id}},\tilde{\mathbb{A}});\mathsf{RK};(\delta,\tilde{\beta},1,1,0))\in RK^{List}$. If true, $\mathcal{C}$ returns RK to $\mathcal{A}$ and resets $b_2=0,b_3=1$. Else, $\mathcal{C}$ chooses some random $\theta,\sigma,r'\in\mathbb{Z}_p$, $\delta,\tilde{\beta}\in\{0,1\}^{\psi}$ and $\mathfrak{K}\in\mathbb{G}_2$ for all $a\in S_{\mathsf{id}}$. $\mathcal{C}$ sets $\mathsf{rk}_1=\mathfrak{K}^{\sigma}\cdot\mathfrak{g}_1^{\theta},\mathsf{rk}_2=g^{\theta},\mathsf{rk}_3=\mathfrak{g}^{r'\sigma},\mathsf{rk}_{4,a}=\mathfrak{T}_a^{r'\sigma}$, and constructs rk_5 as in the real scheme. Finally $\mathcal{C}$ returns $\mathsf{RK}=(\mathsf{rk}_1,\mathsf{rk}_2,\mathsf{rk}_3,\{\mathsf{rk}_4\},\mathsf{rk}_5,S_{\mathsf{id}})$ and adds $((S_{\mathsf{id}},\tilde{\mathbb{A}});\mathsf{RK};(\delta,\tilde{\beta},1,0,1))$ to RK^{List}.

- Otherwise, $S_{\mathsf{id}}\notin\mathbb{A}^*$. If $((S_{\mathsf{id}},\tilde{\mathbb{A}});\mathsf{RK};(\delta,\tilde{\beta},0,1,0))\in RK^{List}$, $\mathcal{C}$ returns RK and resets $b_2=0,b_3=1$. Else $\mathcal{C}$ first constructs RK as in the real scheme, returns RK to $\mathcal{A}$ and adds $((S_{\mathsf{id}},\tilde{\mathbb{A}});\mathsf{RK};(\delta,\tilde{\beta},0,0,1))$ to RK^{List}.

(7) REQ: On input $(S_{\mathsf{id}},\tilde{\mathbb{A}},\mathsf{CT}_{\mathbb{A}})$, $\mathcal{C}$ aborts if $\mathsf{CT}_{\mathbb{A}}$ fails the validity checking (i.e. S_{id} does not satisfy $\mathbb{A}$ or it fails the checking of equation 3.). Otherwise:

- If $S_{\mathsf{id}} \in \mathbb{A}^*$ and $(S', \mathsf{uask}) \in UASK^{List}$ (for any $S' \in \tilde{\mathbb{A}}$), $\mathcal{C}$ checks if there exists $(\cdot; s) \in H_1^{List}$ such that $C' = g^s$. If no such tuple exists, $\mathcal{C}$ returns $\perp$. Else $\mathcal{C}$ checks if $((S_{\mathsf{id}}, \tilde{\mathbb{A}}); \perp; (\delta, \tilde{\beta}, \perp, 1, \perp)) \in RK^{List}$. If no, $\mathcal{C}$ chooses $\delta, \tilde{\beta} \in \{0,1\}^{\psi}$, generates rk_5 as in the real scheme and constructs $B' = (\hat{e}(g^a, \mathfrak{g}^{a^q})^D \cdot \hat{e}(g, \mathfrak{g})^{\sum_{d=1}^{D} \alpha'_d})^{s\xi_2}$, where $\xi_2 = H_4(\delta)$. Finally, $\mathcal{C}$ returns the re-encrypted ciphertext CT_R to $\mathcal{A}$ (with B') and adds $((S_{\mathsf{id}}, \tilde{\mathbb{A}}); \perp; (\delta, \tilde{\beta}, \perp, 1, \perp))$ to RK^{List} and adds $((S_{\mathsf{id}}, \tilde{\mathbb{A}}); \mathsf{CT}_R; (0,0,1))$ to RE^{List}.

- If $S_{\mathsf{id}} \in \mathbb{A}^*$ and $(S', \mathsf{uask}) \notin UASK^{List}$ (for all $S' \in \tilde{\mathbb{A}}$), $\mathcal{C}$ first constructs the re-encryption key RK as the second case of RKQ and further re-encrypts $\mathsf{CT}_{\mathbb{A}}$ to a ciphertext CT_R for $\mathcal{A}$. $\mathcal{C}$ adds $((S_{\mathsf{id}}, \tilde{\mathbb{A}}); \mathsf{RK}; (\delta, \tilde{\beta}, 1, 1, 0))$ to RK^{List} and adds $((S_{\mathsf{id}}, \tilde{\mathbb{A}}); \mathsf{CT}_R; (0,1,0))$ to RE^{List}.

- If $S_{\mathsf{id}} \notin \mathbb{A}^*$, $\mathcal{C}$ first constructs the re-encryption key RK as the third case of RKQ and further re-encrypts $\mathsf{CT}_{\mathbb{A}}$ to a ciphertext CT_R for $\mathcal{A}$. $\mathcal{C}$ adds $((S_{\mathsf{id}}, \tilde{\mathbb{A}}); \mathsf{RK}; (\delta, \tilde{\beta}, 0, 1, 0))$ to RK^{List} and adds $((S_{\mathsf{id}}, \tilde{\mathbb{A}}); \mathsf{CT}_R; (1,0,0))$ to RE^{List}.

(8) DQ: On input $(\mathsf{CT}_{\mathbb{A}}, S_{\mathsf{id}})$, $\mathcal{C}$ aborts if $\mathsf{CT}_{\mathbb{A}}$ fails the validity checking (i.e. S_{id} does not satisfy $\mathbb{A}$ or it fails the checking of equation 3.). Otherwise:

- If $(S', \mathsf{uask}) \in UASK^{List}$ (for any $S' \in \tilde{\mathbb{A}}$), $\mathcal{C}$ recovers the message as in the real scheme using uask.

- Else, $\mathcal{C}$ checks if $((M, \beta, \gamma); s) \in H_1^{List}$ such that $C' = g^s$ and $(R, \delta_1) \in H_2^{List}$ such that $R = (M||\beta||\gamma) \oplus \delta_1$. $\mathcal{C}$ outputs $\perp$ if no such tuple exists; or outputs M otherwise.

(9) RDQ: On input $(\mathsf{CT}_R, S_{\mathsf{id}'})$ with respect to $\mathbb{A}' = (M', \rho')$, $\mathcal{C}$ outputs $\perp$ if one of the following is true:

- there are no tuples $((M, \beta, \gamma); s), ((\delta, \tilde{\beta}); \tilde{s})$ in H_1^{List} such that $\mathfrak{B} = \mathfrak{u}^s$ and $\tilde{C}' = g^{\tilde{s}}$;

- the ciphertext fails the validity check of equation 3.

Otherwise, $\mathcal{C}$ proceeds as follow:

- If $((S_{\mathsf{id}}, \mathbb{A}'); \mathsf{RK}; (\delta, \tilde{\beta}, 1, 0, 1)) \in RK^{List}$ or $((S_{\mathsf{id}}, \mathbb{A}'); \mathsf{CT}_R; (0, 1, 0)) \in RE^{List}$, $\mathcal{C}$ checks if

  $$\hat{e}(\prod_{\rho'(x) \in S_{\mathsf{id}'}} \tilde{C}_x^{w_x}, \mathfrak{g}) = \hat{e}(\tilde{C}', \mathfrak{h}) \cdot \prod_{\rho'(x) \in S_{\mathsf{id}'}} \hat{e}(\tilde{C}'_x, \mathfrak{T}_{\rho'(x)}^{-w_x}), \tag{6}$$

  where with the knowledge of (M', ρ'), $\mathcal{C}$ can find some constants w_x such that $\sum_{\rho'(x) \in S_{\mathsf{id}'}} w_x M'_x = (1, 0, \ldots, 0)$. If equation 6 does not hold, $\mathcal{C}$ outputs $\perp$. Otherwise, $\mathcal{C}$ reconstructs $C' = g^s$ with the knowledge of s and checks if equation 3 holds. If not, $\mathcal{C}$ outputs $\perp$. Otherwise, $\mathcal{C}$ uses the above random re-encryption key RK to check the validity of B' as in equation 5. If it does not hold, $\mathcal{C}$ outputs $\perp$. Otherwise, $\mathcal{C}$ checks if $(R; \delta_1) \in H_2^{List}$ for $\delta_1 = C \oplus (M||\beta||\gamma)$ and $R = (\hat{e}(g^a, \mathfrak{g}^{a^q})^D \cdot \hat{e}(g, \mathfrak{g})^{\sum_{d=1,D} \alpha_d})^s$. If no such tuple exists, $\mathcal{C}$ outputs $\perp$. Else $\mathcal{C}$ returns M to $\mathcal{A}$. (Note that $\mathcal{C}$ checks the derivatives of the challenge ciphertext in the above manner.)

- Else if $(S_{\mathsf{id}'}, \mathsf{uask}) \in UASK^{List}$, $\mathcal{C}$ recovers M as in the real scheme using uask.

- Else $\mathcal{C}$ outputs $\perp$ if any of the following is true: (1) Equation 3 or 6 does not hold; (2) If no tuple $(R; \delta_1) \in H_2^{List}$ exists for $\delta_1 = C \oplus (M||\beta||\gamma)$ and $R = (\hat{e}(g^a, \mathfrak{g}^{a^q})^D \cdot \hat{e}(g, \mathfrak{g})^{\sum_{d=1,D} \alpha_d})^s$; or (3) $\sigma = H_3(C, \mathfrak{B}, \{C_x, C'_x\}_{x \in [1,l]}, \mathbb{A})^s$ or $B' = R^{H_4(\delta)}$ does not hold.

  Otherwise, $\mathcal{C}$ outputs M.

Challenge. $\mathcal{A}$ outputs M_0^*, M_1^* to $\mathcal{C}$. $\mathcal{C}$ chooses a random bit b and responds as follows.

For each row i of M^*, set $x^* = \rho^*(i)$. For all $i \in [1, l^*]$, denote R_i as the set of all $i \neq y$ such that $\rho^*(i) = \rho^*(y)$. $\mathcal{C}$ chooses some random $\chi_2, \ldots, \chi_{n^*}, r'_1, \ldots, r'_{l^*} \in \mathbb{Z}_p$ and implicitly sets the vector $\vec{v} = (s, sa + \chi_2, \ldots, sa^{n^*-1} + \chi_{n^*})$ by setting: $C'_i = g^{r'_i + sb_i}$, $C_i = T_{x^*}^{-r'_i} \cdot \prod_{j=2}^{n^*} g^{a \cdot \chi_j M^*_{i,j}} \cdot g^{-b_i s \cdot z_{x^*}} \cdot \prod_{y \in R_i} \prod_{j=1}^{n^*} (g^{\frac{a^j s b_i}{b_y}})^{-M^*_{y,j}}$. $\mathcal{C}$ chooses some random $\beta^*, \gamma^* \in \{0, 1\}^{\psi}$ and $C^* \in \{0, 1\}^{3\psi}$, sets $C' = g^s$, $\mathfrak{B}^* = \mathfrak{u}^s = \mathfrak{g}^{s\zeta}$ and puts $(T^D \cdot \hat{e}(g, \mathfrak{g})^{\sum_{i=1}^{D} \alpha'_d}; C^* \oplus (M_b^*||\beta^*, \gamma^*))$ in H_2^{List}. $\mathcal{C}$ chooses a random $\xi_1^* \in \mathbb{Z}_p$, sets $\sigma^* = (g^s)^{\xi_1^*}$ and puts $((C^*, \mathfrak{B}^*, \{C_x, C'_x\}_{x \in [1,l^*]}, \mathbb{A}^*); g^{\xi_1^*}; \xi_1^*)$ in H_3^{List}.

$\mathcal{C}$ returns the challenge ciphertext $\mathsf{CT}^* = (C^*, C', \{C_x, C'_x\}_{x \in [1,l]}, \mathfrak{B}^*, \sigma^*, \mathbb{A}^*)$.

Query Phase 2. Same as the Query Phase 1.

Guess. $\mathcal{A}$ outputs a guess b'. If $b = b'$, then $\mathcal{C}$ answers its challenger that $T = \hat{e}(g, \mathfrak{g})^{a^{q+1}s}$; else $\mathcal{C}$ decides that $T \in_R \mathbb{G}_T$.

Analysis. We consider the cases for simulation failure:

- $\mathcal{A}$ queries $(M_b^*, \beta^*, \gamma^*)$ to H_1 or $T^D \cdot \hat{e}(g, \mathfrak{g})^{\sum_{i=1}^{D} \alpha'_d}$ to H_2 in phase 1. Denote these events as H_1^* and H_2^* respectively, and the probability of these events as $Adv^{TCR}_{H_1^*,\mathcal{A}}$ and $Adv^{TCR}_{H_2^*,\mathcal{A}}$ respectively.

- $\mathcal{A}$ submits a valid original ciphertext to REQ without issuing the query H_1. Denote this event as $\mathsf{ReEncErr}$ and it appears with probability $\leq \frac{q_{\mathsf{REQ}}}{p}$, where q_{REQ} is the number of REQ queries.

- $\mathcal{A}$ submits a valid original ciphertext to DQ or RDQ without querying $Z = \prod_{d=1}^{D} \hat{e}(g, \mathfrak{g})^{\alpha_d H_1(M,\beta,\gamma)}$ to H_2. Let valid be the event that the ciphertext is valid, $\mathsf{QueryH_1}$ be the event that $\mathcal{A}$ has queried (M, β, γ) to H_1 and $\mathsf{QueryH_2}$ be the event that $\mathcal{A}$ has queried Z to H_2. From the simulation, we have $\Pr[\mathsf{valid}|\neg\mathsf{QueryH_2}] \leq \Pr[\mathsf{QueryH_1}|\neg\mathsf{QueryH_2}] + \Pr[\mathsf{valid}|\neg\mathsf{QueryH_1} \wedge \neg\mathsf{QueryH_2}] \leq \frac{q_{H_1}}{2^{3\psi}} + \frac{1}{p}$, $\Pr[\mathsf{valid}|\neg\mathsf{QueryH_1}] \leq \frac{q_{H_2}}{p} + \frac{1}{p}$, where q_{H_1} and q_{H_2} are the number of H_1 and H_2 queries respectively. Denote DecErr as the event that $\mathsf{valid}|(\neg\mathsf{QueryH_1} \vee \neg\mathsf{QueryH_2})$. Then we have $\Pr[\mathsf{DecErr}] = (\frac{q_{H_1}}{2^{3\psi}} + \frac{2+q_{H_2}}{p}) \cdot (q_{\mathsf{DQ}} + q_{\mathsf{RDQ}})$, where q_{DQ} and q_{RDQ} are the number of DQ and RDQ queries respectively.

Let Bad be the event that $(\mathsf{H}_1^*|\neg\mathsf{H}_2^*)\vee\mathsf{H}_2^*\vee\mathsf{ReEncErr}\vee\mathsf{DecErr}$. Then we have $\mathcal{A}$'s advantage: $\epsilon = |\Pr[b = b'] - \frac{1}{2}| \leq \frac{1}{2}\Pr[\mathsf{Bad}] = \frac{1}{2}\Pr[(\mathsf{H}_1^*|\neg\mathsf{H}_2^*)\vee\mathsf{H}_2^*\vee\mathsf{ReEncErr}\vee \mathsf{DecErr}] \leq \frac{1}{2}(Adv^{TCR}_{H_2^*,\mathcal{A}} + \frac{q_{H_1}+q_{H_1}(q_{\mathsf{DQ}}+q_{\mathsf{RDQ}})}{2^{3\psi}} + \frac{(2+q_{H_2})(q_{\mathsf{DQ}}+q_{\mathsf{RDQ}})+q_{\mathsf{REQ}}}{p})$. Therefore, the advantage of solving the problem instance is at least $\frac{1}{q_{H_2}}(Adv^{TCR}_{H_2^*,\mathcal{A}}) \geq \frac{1}{q_{H_2}}(2\epsilon - \frac{q_{H_1}+q_{H_1}(q_{\mathsf{DQ}}+q_{\mathsf{RDQ}})}{2^{3\psi}} - \frac{(2+q_{H_2})(q_{\mathsf{DQ}}+q_{\mathsf{RDQ}})+q_{\mathsf{REQ}}}{p})$, which is not negligible if ϵ is not negligible. Finally, it is straightforward to see that the above simulation runs in polynomial time.

Theorem 2. *Our scheme is selective-attribute secure against IND-CCA-Re if the asymmetric decisional q-parallel BDHE assumption holds in the random oracle model.*

Proof: The security proof is almost the same as the IND-CCA-Or security proof. We only sketch the differences below: The Setup and the Guess phase are the same as before. The $\mathsf{Query\ Phase}$ 1 and 2 are almost the same, except following the query constraints in the IND-CCA-Re security model.

$\mathsf{Challenge.}$ $\mathcal{A}$ outputs M_0^*, M_1^* and an access policy $\mathbb{A} = (A, \rho)$ to $\mathcal{C}$ (A is an $l \times n$ matrix). $\mathcal{C}$ chooses a random bit b and responds as follows.

$\mathcal{C}$ encrypts M_b^* for $\mathbb{A}$ as in the real $\mathsf{PHR\text{-}Encrypt}$ scheme and obtains $\mathsf{CT}_{\mathbb{A}} = (C, C'\{C_x, C'_x\}_{x\in[1,l]}, \mathfrak{B}, \sigma, \mathbb{A})$.

During the $\mathsf{PHR\text{-}Encrypt}$ computation, $((M_b^*, \beta, \gamma); s)$ is stored in H_1^{List}. $\mathcal{C}$ picks some random $\tilde{\beta}, \delta^* \in \{0,1\}^{\psi}$ and issues the $H_4(\delta^*)$ query to obtain ξ_2^*. $\mathcal{C}$ sets $B' = (\hat{e}(g^a, \mathfrak{g}^{a^q})^D \cdot \hat{e}(g, \mathfrak{g})^{\sum_{d=1}^{D} \alpha'_d})^{s\xi_2^*}$.

Recall that the challenge access policy is (M^*, ρ^*) where M^* is an $l^* \times n^*$ matrix). Denote $x^* = \rho^*(i)$. $\mathcal{C}$ chooses some random $\chi_2, \ldots, \chi_{n^*}, r'_1, \ldots, r'_{l^*} \in \mathbb{Z}_p$ by setting: $\tilde{C}'_i = g^{r'_i + sb_i}$, $\tilde{C}_i = T_{x^*}^{-r'_i} \cdot \prod_{j=2}^{n^*} g^{a\cdot\chi_j M^*_{i,j}} \cdot g^{-b_i s z_{x^*}} \cdot \prod_{y\in R_i} \prod_{j=1}^{n^*} (g^{\frac{a^j s b_i}{b_k}})^{-M^*_{y,j}}$, where for all $i \in [1, l^*]$, denote R_i as the set of all $i \neq y$ such that $\rho^*(i) = \rho^*(y)$.

$\mathcal{C}$ chooses some random $\beta^*, \gamma^* \in \{0,1\}^{\psi}$ and $\tilde{C} \in \{0,1\}^{3\psi}$, sets $\tilde{C}' = g^s$, $\mathfrak{B}^* = \mathfrak{u}^s = \mathfrak{g}^{s\zeta}$ and puts $(T^D \cdot \hat{e}(g, \mathfrak{g})^{\sum_{i=1}^{D} \alpha'_d}; \tilde{C} \oplus (M_b^* || \beta^*, \gamma^*))$ in H_2^{List}.

$\mathcal{C}$ picks a random $\xi_3^* \in \mathbb{Z}_p$ and sets $\tilde{\sigma} = (g^s)^{\xi_3^*}$. $\mathcal{C}$ puts $((\tilde{C}, \tilde{C}', \{\tilde{C}_i, \tilde{C}'_i\}_{i\in[1,l^*]}, \mathbb{A}^*); \tilde{\sigma}; \xi_3^*)$ in H_5^{List}.

$\mathcal{C}$ sets $\mathsf{rk}_5^* = (\tilde{C}, \tilde{C}', \{\tilde{C}_i, \tilde{C}'_i\}_{i\in[1,l^*]}, \tilde{\sigma}, \mathbb{A}^*)$. $\mathcal{C}$ returns the challenge re-encrypted ciphertext $\mathsf{CT}_R^* = (C, \{C_x, C'_x\}_{x\in[1,l]}, \mathfrak{B}, \sigma, \mathsf{rk}_5^*, \mathfrak{B}^*, S_{\mathsf{id}}, \mathbb{A})$.

$\mathsf{Analysis.}$ It is obvious that $\mathsf{CT}_{\mathbb{A}}$ is the challenge phase is valid. If $T = \hat{e}(g, \mathfrak{g})^{a^{q+1}s}$, then rk_5^* is (part of) a valid re-encryption key. The challenge re-encrypted ciphertext is re-encrypted with this key. If $T \in \mathbb{G}_T$, the challenge ciphertext is independent of the bit b chosen by $\mathcal{C}$ in the view of $\mathcal{A}$.

The probability analysis is almost the same as the previous proof, except that we have to take the event H_5^* into account: the event that $\mathcal{A}$ has queried $(\tilde{C}, \tilde{C}', \{\tilde{C}_i, \tilde{C}'_i\}_{i\in[1,l^*]}, \mathbb{A}^*)$ in $\mathsf{Query\ Phase\ 1}$. Now, the advantage of solving the problem instance is at least $\frac{1}{q_{H_2}}(Adv^{TCR}_{H_2^*,\mathcal{A}}) \geq \frac{1}{q_{H_2}}(2\epsilon - \frac{q_{H_1} + q_{H_1}(q_{\mathsf{DQ}} + q_{\mathsf{RDQ}})}{2^{3\psi}} - \frac{(2+q_{H_2})(q_{\mathsf{DQ}} + q_{\mathsf{RDQ}}) + q_{\mathsf{REQ}} + q_{H_5}}{p})$, where q_{H_5} is the number of H_5 oracle queries. We omit the details.

4.6. Efficiency Analysis

We first evaluate the storage of our system. Let $B_{\mathbb{G}_1}, B_{\mathbb{G}_2}$ and $B_{\mathbb{G}_T}$ denote the size of a group element in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ respectively while $B_{\mathbb{Z}_p}$ denote the size of a group element in $\mathbb{Z}_p$. Assume 160-bit elliptic curve is used for the system. Also assume the following setting: $|n_k| = 5$, $D = 5$, $K = 5$,

$|S_{\mathsf{id}}| = 15$, $l = \tilde{l} = 3$, $|\mathbb{A}| = |\tilde{\mathbb{A}}| = 5$ and the description of an attribute is encoded into a 160-bit string. The sizes of various keys are shown in Table 2.

Table 2: Sizes of various keys

	CPK_d+ CAPK_d	CMSK_k	APK_k+ ACPK_k	AMSK_k	ucsk_d	ucpk_d	uask_a	$\mathsf{DK}_{\mathsf{id}}$	RK	CT
group element	$1B_{\mathbb{G}_T}+$ $1B_{\mathbb{G}_2}$	$2B_{\mathbb{Z}_p}$	$(\vert n_k\vert + D)B_{\mathbb{G}_1}$ $+\vert n_k\vert B_{\mathbb{G}_2}$	$(\vert n_k\vert+$ $D)B_{\mathbb{Z}_p}$	$1\ B_{\mathbb{G}_2}$	$(K+1)B_{\mathbb{G}_2}$ $+1B_{\mathbb{G}_1}$	$1B_{\mathbb{G}_2}$	$(2D+$ $\vert S_{\mathsf{id}}\vert)B_{\mathbb{G}_2}$	$(2\tilde{l}+3)B_{\mathbb{G}_1}+$ $(\vert S_{\mathsf{id}}\vert+2)B_{\mathbb{G}_2}+$ $\vert\mathbb{A}\vert+\vert S_{\mathsf{id}}\vert+2l$ bits	$(2l+2\tilde{l}+3)B_{\mathbb{G}_1}$ $+1B_{\mathbb{G}_2}+1B_{\mathbb{G}_T}+$ $\vert\mathbb{A}\vert+\vert\tilde{\mathbb{A}}\vert+5l$ bits
bits	1432	318	3990	1590	478	3020	478	11950	9592	3857

Next we evaluate the computation cost of our system. Let $E_{\mathbb{G}_1}, E_{\mathbb{G}_2}$ and $E_{\mathbb{G}_T}$ denote the exponentiation of a group element in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ respectively while P denote a pairing operation. We omit other lightweight algorithms such as multiplication and hashing. We assume the same setting as above.

After that, we implemented our scheme by using the PBC library (version 0.5.14), GMP library (6.0.0) with VS .NET 2010. The testbed is Intel Xeon CPUE5-2680 2.70GHz, 4 GB Ram, Windows 7 32 bits. We tested with the MNT curve (type d159) and the BN curve (type f) in the PBC library, and found that the former curve gives a better performance for our scheme. The result is summarized in Table 3, which is the average running time for 10 iterations, in millisecond.

Table 3: Running Time

	CA-Setup	AA-Setup	CA-KeyGen	AA-KeyGen	PHR-Encrypt	PHR-Decrypt	Re-KeyGen	PHR-ReEnc	PHR-ReDec
algo.	$E_{\mathbb{G}_T}$	$(\vert n_k\vert + D)E_{\mathbb{G}_1}$ $+\vert n_k\vert E_{\mathbb{G}_2}$	$(3+K)E_{\mathbb{G}_2}$ $+1E_{\mathbb{G}_1}$	$4D\ P+$ $DE_{\mathbb{G}_2}$	$(3+2l)E_{\mathbb{G}_1}$ $+\ E_{\mathbb{G}_2}+E_{\mathbb{G}_T}$	$(6+4l)P+$ $1E_{\mathbb{G}_1}+2lE_{\mathbb{G}_T}$	$(3l+3)E_{\mathbb{G}_1}+$ $(\vert S_{\mathsf{id}}\vert+2)E_{\mathbb{G}_2}+1E_{\mathbb{G}_T}$	$(2l+2\tilde{l}+9)P$ $+(l+\tilde{l})E_{\mathbb{G}_T}$	$(3+2\tilde{l})P+3E_{\mathbb{G}_1}$ $+(\tilde{l}+1)E_{\mathbb{G}_T}$
msec.	37.6	181	234	727.2	56.2	299.8	271.6	506.2	250

5. Cross Domain Operations

Our framework is flexible to support different operations on PHR across multiple domains, where each domain could represent a single country with its own set of regulations regarding the storage or handling of PHRs. We consider the following two cross domain operations.

5.1. Complete Transfer of PHRs to another domain

This operation is needed when one domain would like to transfer all of its records to another domain. For instance, if two domains are merging, or a new domain is scheduled to replace an existing one. The operation requires that a vast amount of PHRs in an existing domain to be transferred to another domain in such a way that the access policy of these PHRs in the target domain could be different. We assume this operation is initiated and handled by the set of central authorities.

Our framework utilises the cryptographic primitive called ABE, which is a form of public key encryption with a set of CAs. When the set of CAs cooperate, they can decrypt all PHRs in our system[4]. Furthermore, since it is a public key encryption system, the set of CAs can, without the cooperation of the target domain, re-encrypt the PHR file under the access policy required by the target domain. The algorithm is described below in the framed box of PHR-Transfer.

> PHR-Transfer. Given an encryption of a PHR file $(C, C', \{C_x, C'_x\}_{x\in[1,l]}, B, \sigma, \mathbb{A} = (A, \rho))$, the set of CAs, with secret keys $\alpha_1, \ldots, \alpha_d$, can recover the underlying message (regardless of the policy) as: $M||\beta||\gamma = C \oplus H_2(\prod_{d=1}^{D} \hat{e}(C', g)^{\alpha_d})$. After recovering M, the set of CAs can encrypt M for the target domain using the algorithm PHR-Encrypt.

5.2. User initiated PHR transfer (cross domain PHR sharing)

This operation allows a user who has access to a certain PHR to share this record with users in a different domain. For instance, a patient immigrating to another country may wish to transfer his/her PHR to the domain in the new country.

The idea of cross domain PHR sharing is similar and any user who has access to a PHR record can conduct the sharing by first issuing a PHR-Decrypt on the record in the current domain, which requires his/her secret key. After that, the user issues PHR-Encrypt in the foreign domain, which only requires the public parameters of the foreign domain.

[4] That is the reason we assume at least one of the CAs is honest and will not abuse its power.

6. Concluding Remarks

In this paper, we proposed a general framework of secure sharing of PHRs. Our system enables patients to securely store and share their PHR in the cloud server to their carers or family members. Treating doctors can further refer the patients' medical record to specialists for research purposes, while the patients' personal information remain private. In addition, cross domains operations can be supported. We provided a concrete instantiation of our system. We also gave a simulation result for it. We believe our system is practical and can be deployed by various medical systems in the world.

We also remark that although the mechanism proposed in this paper is specifically targeted for PHR, we do not limit the application of the mechanism to other areas such as data sharing in the cloud or secure vehicular network communication.

Acknowledgement

The work of X. Huang is supported by National Natural Science Foundation of China (61472083), Program for New Century Excellent Talents in Fujian University (JA14067), Distinguished Young Scholars Fund of Fujian (2016J06013), and Fujian Normal University Innovative Research Team (IRTL1207). Zoe L. Jiang is supported in part by National Natural Science Foundation of China (No. 61402136).

References

[1] J. Akinyele, C. Lehmann, M. Green, M. Pagano, Z. Peterson, and A. Rubin. Self-protecting electronic medical records using attribute-based encryption. Cryptology ePrint Archive, Report 2010/565, 2010.

[2] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *NDSS*. The Internet Society, 2005.

[3] N. Attrapadung and S. Yamada. Duality in ABE: converting attribute based encryption for dual predicate and dual policy via computational encodings. In *CT-RSA 2015*, volume 9048 of *LNCS*, pages 87–105. Springer, 2015.

[4] A. Beimel. *Secure Schemes for Secret Sharing and Key Distribution.* PhD thesis, Israel Institute of Technology, Israel, 1996.

[5] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, volume 1403 of *LNCS*, pages 127–144. Springer, 1998.

[6] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, volume 2248 of *LNCS*, pages 514–532. Springer, 2001.

[7] R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *CCS '07*, pages 185–194. ACM, 2007.

[8] M. Chase. Multi-authority attribute based encryption. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 515–534. Springer, 2007.

[9] M. Chase and S. S. M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *CCS 2009*, pages 121–130. ACM, 2009.

[10] H. Deng, Q. Wu, B. Qin, W. Susilo, J. K. Liu, and W. Shi. Asymmetric cross-cryptosystem re-encryption applicable to efficient and secure mobile access to outsourced data. In *ASIACCS*, pages 393–404. ACM, 2015.

[11] H. Deng, Q. Wu, B. Qin, W. Susilo, J. K. Liu, and W. Shi. Asymmetric cross-cryptosystem re-encryption applicable to efficient and secure mobile access to outsourced data. In F. Bao, S. Miller, J. Zhou, and G. Ahn, editors, *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '15, Singapore, April 14-17, 2015*, pages 393–404. ACM, 2015.

[12] S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO 2013*, volume 8043 of *LNCS*, pages 479–499. Springer, 2013.

[13] J. Han, W. Susilo, Y. Mu, and J. Yan. Privacy-preserving decentralized key-policy attribute-based encryption. *IEEE Trans. Parallel Distrib. Syst.*, 23(11):2150–2162, 2012.

[14] S. Hohenberger and B. Waters. Online/offline attribute-based encryption. In *Public-Key Cryptography*, volume 8383 of *LNCS*, pages 293–310. Springer, 2014.

[15] J. Hur and D. K. Noh. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Trans. Parallel Distrib. Syst.*, 22(7):1214–1221, 2011.

[16] L. Ibraimi, M. Asim, and M. Petkovic. Secure management of personal health records by applying attribute-based encryption. Number TR-CTI in CTIT technical report series, Enschede, July 2009. Centre for Telematics and Information Technology, University of Twente.

[17] J. Lai, R. H. Deng, C. Guan, and J. Weng. Attribute-based encryption with verifiable outsourced decryption. *IEEE T. on Information Forensics and Security*, 8(8):1343–1354, 2013.

[18] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, volume 6110 of *LNCS*, pages 62–91. Springer, 2010.

[19] A. B. Lewko, Y. Rouselakis, and B. Waters. Achieving leakage resilience through dual system encryption. In Y. Ishai, editor, *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, volume 6597 of *Lecture Notes in Computer Science*, pages 70–88. Springer, 2011.

[20] A. B. Lewko and B. Waters. Decentralizing attribute-based encryption. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 568–588. Springer, 2011.

[21] A. B. Lewko and B. Waters. Unbounded HIBE and attribute-based encryption. In *EUROCRYPT*, volume 6632 of *LNCS*, pages 547–567. Springer, 2011.

[22] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *IEEE Trans. Paral. Dist. Sys.*, 24(1):131–143, 2013.

[23] K. Liang, M. H. Au, J. K. Liu, W. Susilo, D. S. Wong, G. Yang, T. V. X. Phuong, and Q. Xie. A dfa-based functional proxy re-encryption scheme for secure public cloud data sharing. *IEEE T. on Information Forensics and Security*, 9(10):1667–1680, 2014.

[24] K. Liang, L. Fang, D. S. Wong, and W. Susilo. A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security. Cryptology ePrint Archive, Report 2013/236, 2013.

[25] K. Liang, J. K. Liu, D. S. Wong, and W. Susilo. An efficient cloud-based revocable identity-based proxy re-encryption scheme for public clouds data sharing. In *ESORICS*, volume 8712 of *Lecture Notes in Computer Science*, pages 257–272. Springer, 2014.

[26] X. Liang, Z. Cao, H. Lin, and J. Shao. Attribute based proxy re-encryption with delegating capabilities. In *ASIACCS*, pages 276–286. ACM, 2009.

[27] X. Liang, R. Lu, X. Lin, and X. S. Shen. Patient self-controllable access policy on phi in ehealthcare systems. In *AHIC*, pages 1–5, 2010.

[28] Z. Liu, Z. Cao, Q. Huang, D. S. Wong, and T. H. Yuen. Fully secure multi-authority ciphertext-policy attribute-based encryption without random oracles. In *ESORICS*, volume 6879 of *LNCS*, pages 278–297. Springer, 2011.

[29] R. Lu, X. Lin, and X. S. Shen. Spoc: A secure and privacy-preserving opportunistic computing framework for mobile-healthcare emergency. *IEEE Trans. Parallel Distrib. Syst.*, 24(3):614–624, 2013.

[30] S. Luo, J. Hu, and Z. Chen. Ciphertext policy attribute-based proxy re-encryption. In *ICICS*, volume 6476 of *LNCS*, pages 401–415. Springer, 2010.

[31] I. E. Magnin and J. Montagnat. The grid and the biomedical community: Achievements and open issues. presented at the EGEE User Forum, CERN, Geneva, Switzerland, 2006.

[32] T. Mizuno and H. Doi. Hybrid proxy re-encryption scheme for attribute-based encryption. In *Inscrypt*, volume 6151 of *LNCS*, pages 288–302. Springer, 2009.

[33] S. Narayan, M. Gagné, and R. Safavi-Naini. Privacy preserving ehr system using attribute-based infrastructure. In *CCSW*, pages 47–52. ACM, 2010.

[34] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *CCS*, pages 195–203. ACM, 2007.

[35] Y. Rouselakis and B. Waters. Efficient statically-secure large-universe multi-authority attribute-based encryption. *IACR Cryptology ePrint Archive*, 2015:16, 2015. To appear in FC 2015.

[36] A. Sahai, H. Seyalioglu, and B. Waters. Dynamic credentials and ciphertext delegation for attribute-based encryption. In *CRYPTO*, volume 7417 of *LNCS*, pages 199–217. Springer, 2012.

[37] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005.

[38] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO 84*, volume 196 of *LNCS*, pages 47–53. Springer-Verlag, 1984.

[39] B. Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *CRYPTO*, volume 5677 of *LNCS*, pages 619–636. Springer, 2009.

[40] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography*, volume 6571 of *LNCS*, pages 53–70. Springer, 2011.

[41] Q. Wu, B. Qin, L. Zhang, J. Domingo-Ferrer, O. Farràs, and J. A. Manjón. Contributory broadcast encryption with efficient encryption and short ciphertexts. *IEEE Trans. Computers*, 65(2):466–479, 2016.

[42] F. Xhafa, J. Wang, X. Chen, J. K. Liu, J. Li, and P. Krause. An efficient PHR service system supporting fuzzy keyword search and fine-grained access control. *Soft Comput.*, 18(9):1795–1802, 2014.

[43] S. Yu, C. Wang, K. Ren, and W. Lou. Attribute based data sharing with attribute revocation. In *ASIACCS*, pages 261–270. ACM, 2010.

[44] Z. Yu, M. H. Au, Q. Xu, R. Yang, and J. Han. Leakage-resilient functional encryption via pair encodings. In J. K. Liu and R. Steinfeld, editors, *Information Security and Privacy - 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part I*, volume 9722 of *Lecture Notes in Computer Science*, pages 443–460. Springer, 2016.