(Revised Version)
Privacy-preserving Smart IoT-based Healthcare Big Data Storage and Self-adaptive Access Control System

Yang Yang[a,b,c,d], Xianghan Zheng[a,c,d], Wenzhong Guo[a,c,d], Ximeng Liu[a,b,c,d], Victor Chang[e]

[a] *College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China.*
[b] *University Key Laboratory of Information Security of Network Systems (Fuzhou University), Fujian Province, China.*
[c] *Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, Fuzhou, China*
[d] *Key Laboratory of Spatial Data Mining & Information Sharing, Ministry of Education, Fuzhou, China*
[e] *IBSS, Xi'an Jiaotong-Liverpool University, Suzhou, China.*

Abstract

In this paper, a privacy-preserving smart IoT-based healthcare big data storage system with self-adaptive access control is proposed. The aim is to ensure the security of patients' healthcare data, realize access control for normal and emergency scenarios, and support smart deduplication to save the storage space in big data storage system. The medical files generated by the healthcare IoT network are encrypted and transferred to the storage system, which can be securely shared among the healthcare staff from different medical domains leveraging a cross-domain access control policy. The traditional access control technology allows the authorized data users to decrypt patient's sensitive medical data, but also hampers the first-aid treatment when the patient's life is threatened because the on-site first-aid personnel are not permitted to get patient's historical medical data. To deal with this dilemma, we propose a secure system to devise a novel two-fold access control mechanism, which is self-adaptive for both normal and emergency situations. In normal application, the healthcare staff with proper attribute secret keys can have the data access privilege; in emergency application, patien-

*Corresponding author: Xianghan Zheng
Email addresses: yang.yang.research@gmail.com (Yang Yang), xianghan.zheng@fzu.edu.cn (Xianghan Zheng), guowenzhong@fzu.edu.cn (Wenzhong Guo), snbnix@gmail.com (Ximeng Liu), ic.victor.chang@gmail.com (Victor Chang)

t's historical medical data can be recovered using a password-based break-glass access mechanism. To save the storage overhead in the big data storage system, a secure deduplication method is designed to eliminate the duplicate medical files with identical data, which may be encrypted with different access policies. A highlight of this smart secure deduplication method is that the remaining medical file after the deduplication can be accessed by all the data users authorized by the different original access policies. This smart healthcare big data storage system is formally proved secure, and extensive comparison and simulations demonstrate its efficiency.

Keywords: privacy-preserving, healthcare big data storage, internet-of-things, self-adaptive access control, smart deduplication

---

1. Introduction

The development of Internet of Things (IoT) technology [1] makes it possible for the medical institutes to provide high-quality, convenient and pervasive healthcare services. A collection of tiny wireless sensor nodes can be implanted into (or adhere on the surface of) the patient to monitor the health condition and collect the vital physiological data, which are helpful for both emergency medical decisions and chronic disease detection [12, 32]. The elderly people can use wearable or implanted medical sensors to access modern medical services anywhere and anytime to improve their quality of life.

When the physiological data is collected by the medical IoT network, it is transmitted to the healthcare big data center for storage and disease diagnosis. To protect the privacy of patients, the medical file needs to be encrypted before the transmission to prevent the eavesdropping on the public domains. The patient enforces access policy on the protected data to define the authorized attributes and the relationships. Only the users (such as the physician, nurses, anesthetist or the family of the patients) with the proper attribute secret keys have the privilege to decrypt the ciphertext. This encryption method is called attribute-based encryption [33].

In modern medical system, a patient with unknown symptoms may be diagnosed and treated in several hospitals with different medical records stored. Hence, it be-

comes necessary to realize cross-domain secure data sharing system to facilitate the patient's treatment among different hospitals. For example, the examination report produced by hospital A can be accessed by the doctors in hospital B. The encrypted medical files generated by different hospitals are sent to the public clouds for storage and pervasive data access [9, 10]. The patient defines the cross-domain access policy for his protected healthcare records. Each medical staff registers to his/her medical institute to get the attribute secret key, which is utilized to decrypt patient's encrypted files.

The emergency situations may occur in medical systems, for instance, a car accident happens or the patient suddenly faints. In these emergency scenarios, the electronic medical records of the patents are urgently demanded to save their lives. However, the ambulance personnel at the scene often do not have the permission to access the encrypted medical files. In this situation, the security mechanism to protect the data privacy may hinder the emergency rescue of the patient's life. Therefore, it is crucial to design a break-glass access method for the ambulance personnel to access to the electronic medical files even though they do not have the related attribute secret key. At the same time, the break-glass access method should be manageable and accountable to prevent malicious data access by the attacker.

In addition to these security protection concerns, the same healthcare data may be encrypted by different medical staff using different access policies in the healthcare big data storage system. These ciphertext are simultaneously outsourced to the public cloud, which can occupy a huge storage space. In order to save storage space and transfer overhead, an effective way is to eliminate the redundant ciphertext of the same message in the big data storage system, which is called deduplication.

1.1. Our Contributions

To deal with the above security concerns, we propose a privacy-preserving healthcare big data storage and self-adaptive access control system with smart deduplication. Our main contributions are summarized as follows.

- *Smart cross-domain data sharing*: In real application, patients may belong to different medical institutes (such as different hospitals and clinics). The system

is divided into several medical domains according to the medical institutes and supports that the encrypted health files of a patient from medical institute MI_1 can be securely shared among the healthcare medics or researchers from medical institutes $(MI_1, \cdots, MI_n)$. Patients' medical records are encrypted using attribute-based encryption with a cross-domain access policy, so that it can be access by the authorized users in the entire system.

- *Smart self-adaptive access control*: The access control mechanism in this system is self-adaptive to normal and emergency situations. The patients and healthcare staff register to their own medical domain and get the attribute secret keys, which can be used in normal circumstances to access patients' encrypted files. In emergency situations, a break-glass access methodology is designed so that all the historical medical files of a patient can be recovered by a password-based break-glass key.

- *Smart deduplication*: This system supports smart deduplication over attribute-based encrypted data to save the storage space and reduce the transfer cost between the public cloud and data users, which has three phases. Firstly, we can test whether the ciphertext is a valid one. Next, we can test whether the ciphertexts contains identical medical file. Lastly, the ciphertext is re-encrypted using a combined access policy such that all the original defined authorized users of the original ciphertexts can access the encrypted data. During the deduplication process, no plaintext message is leaked to the public clouds.

- *Simulation and security*: This system is comprehensively compared with the other related schemes. We also conduct simulations of these schemes on modern test-bed to test the performance. The comparison and simulation results indicate that our system achieves versatile useful functions and our proposed work is efficient in terms of the storage and computation costs. This system is validated to be indistinguishable against chosen plaintext attack based on the decisional bilinear Diffie-Hellman assumption.

1.2. Related Work

1.2.1. Break-glass Access

In 2009, Brucker et al. [6] developed an access control model with break-glass to prevent system stagnation that may lead to loss of life, and a security architecture supporting break-glass. Later, they integrated break-glass mechanism into attribute based encryption scheme [7] to enable secure logging, which could be used to analyze the users' behaviors during the break-glass access process. Marinovic et al. [24] presented a new break-glass model named Rampole, which could put integrity constraint into the decision process so that a policy maker can manage the break-glass access authority in a fine-grained manner. Maw et al. [25] proposed a break-glass access model for the health care system with wireless sensor network, which considered the enforcement of access permissions across medical domains. However, these research papers [6, 7, 24, 25] only present the architectures rather than the concrete scheme. In 2016, Zhang et al. [35] suggested a password based break-glass access scheme, which is constructed based on two-factor encryption: password based encryption and master secret key based encryption.

1.2.2. Secure Deduplication

In 2013, Bellare et al. [3] proposed a message-locked encryption primitive to achieve secure deduplication, in which the encryption and decryption keys are derived from the message. They extend their work [3] to interactive message-locked encryption [4] with both upload and download protocols. The deduplication systems in both research projects [3, 4] can be interactive. Stanek et al. [29] suggested a secure storage system supporting deduplication in cloud. In 2014, Li et al. [17] proposed a system to control the convergent keys of deduplication. It is illustrated by the Dekey technology, which is constructed based on secret sharing. Later, they developed a scheme to tackle the problem of deduplication with differential privileges in a hybrid cloud architecture [18] based on public cloud and private cloud [8]. The proposal is secure against collusion attack and proved indistinguishable of the file token/duplicate-check token. In 2015, a new secure deduplication system was demonstrated in [20], which is constructed based on password authenticated key exchange protocol. Since these se-

cure deduplication systems do not consider the access control problem, Cui et al. [11] proposed an secure deduplication system with attribute based access control, which is based on zero-knowledge proof.

1.2.3. Attribute Based Encryption

In 2007, an attribute-based encryption (ABE) algorithm is presented by Ostrovsky [27] such that the secret key of users can represent both monotone and non-monotone access policy. In order to reduce the decryption computation overhead, Green et al. [13] suggested to outsource the decryption burden, so that the user can recover the message with lightweight computation. To validate the correctness of the transformed ciphertext, the verifiable outsourced decryption problem is studied in [16, 28, 23] to provide an effective way for the correctness verification. The traitor tracing problem in ABE is investigated in [21, 26] to deduce the identity of the malicious users who have sold the secret key for monetary benefits. The secure search over encrypted ABE data is studied in [19, 30] to provide efficient keyword search. For example, the data user sends a keyword trapdoor to the server to issue a search query. The server responds with the match ciphertext that contain the same keyword. To extend the ABE security for video content, a time-domain attribute based access control scheme is proposed by Yang et al. [31] to protect the cloud based video content sharing, which embeds the time into the ciphertext and keys to realize time control. To reduce the trust on a single authority, the multi-authority ABE scheme are investigated in [14, 15, 34, 22].

2. Preliminary

In this section, we review some basic notions and definitions used in the remaining sections. The main notations are listed in Table 1.

2.1. Access Policy

Definition 1 (Access Structure [2]). Define $\{P_1, \cdots, P_n\}$ as the entities. $\mathbb{A} \subseteq 2^{\{P_1,\cdots,P_n\}}$ is monotone if any B and C satisfies: $C \in \mathbb{A}$ when $B \in \mathbb{A}$ and $B \subseteq C$. An access structure (respectively, monotone access structure) is a collection (resp. monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, \cdots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,\cdots,P_n\}} \backslash \{\phi\}$.

Table 1: Notations

Notation	Description
KGC	key generation center
MI_i	the i-th medical institution
$U_{i,j}$	the j-th user in the i-th medical institution
$PID_{i,j}$	the pseudo identity of $U_{i,j}$
PP	public parameter of the system
MSK	master secret key of the system
PK_i/SK_i	public/secret key of MI_i
$SK_{i,j}$	secret key of user $U_{i,j}$
$DK_{i,j}/TK_{i,j}$	delegation/transformation key of user $U_{i,j}$
$pw_{i,j}$	password of user $U_{i,j}$
$BGK_{i,j}$	break-glass key of user $U_{i,j}$
$(BGK_{i,j,1}, BGK_{i,j,2})$	auxiliary message of $BGK_{i,j}$
$(\mathbb{A}, \rho, \delta)$	access policy
$attr$	attribute
M	electronic medical file
CT	ciphertext of M
CT'	re-encrypted ciphertext of CT
CT_p	partial ciphertext of CT
pf	proof message of ciphertext CT
κ	security parameter
p	a prime number
$\mathbb{Z}_p^*$	$\{1, \cdots, p-1\}$
$\mathbb{Z}^+$	positive integer
$a \in_R S$	a is randomly chosen from S
$i \in [l]$	$1 \leq i \leq n$
$SEnc/SDec$	secure symmetric encryption/decryption
$\mathcal{K}$	symmetric key space of $SEnc/SDec$
H_1	$H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$
H_2	$H_2 : \{0,1\}^* \rightarrow \mathcal{K}$

The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

Definition 2 (Linear Secret Sharing Scheme (LSSS) [2]). A secret-sharing scheme Π over a set of parties $\mathcal{P}$ is called linear (over $\mathbb{Z}_p$) if

- The shares for each party form a vector over $\mathbb{Z}_p$.
- There exists a matrix M with l rows and n columns called the share-generating matrix for Π. For all $i = 1, \cdots, l$, the ith row of M is labeled by a party $\rho(i)$ (ρ is a function from $\{1, \cdots, l\}$ to $\mathcal{P}$). Set the column vector $v = (s, r_2, \cdots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, \cdots, r_n \in \mathbb{Z}_p$ are randomly chosen, then Mv is the vector of l shares of the secret s according to Π. The share $(Mv)_i$ belongs to party $\rho(i)$.

Every LSSS according to the definition achieves the linear reconstruction property [2]. Suppose that Π is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set and $I \subset \{1, \cdots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exists constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}_{i \in I}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} \omega_i \lambda_i = s$. Furthermore, it is shown in [2] that these constants $\{\omega_i\}_{i \in I}$ can be found in time polynomial in the size of the share-generating matrix M. For unauthorized sets, no such constants exist. In this paper, an LSSS matrix (M, ρ) will be used to express an access policy associated to a ciphertext.

2.2. Bilinear Groups

Let $\mathcal{G}_p$ be an algorithm that on input the security parameter λ, outputs the parameters of a prime order bilinear map as $(p, g, \mathbb{G}, \mathbb{G}_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order p and g is a random generator of $\mathbb{G}$. The mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map. The bilinear map e has three properties: (1) *bilinearity*: $\forall u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(uv)^{ab}$. (2) *non-degeneracy*: $e(g, g) \neq 1$. (3) *computability*: e can be efficiently computed.

2.3. Hardness Problem

Assumption 1 (DBDH: decisional bilinear Diffie-Hellman assumption). Let $\mathbb{G}$ be a bilinear group of prime order p and g be a generator of $\mathbb{G}$. Let $\alpha, \beta, \eta \in \mathbb{Z}_p^*$ be chosen

at random. If an adversary $\mathcal{A}$ is given $\vec{y} = (g, g^{\alpha}, g^{\beta}, g^{\eta})$, it is hard for the attacker $\mathcal{A}$ to distinguish $e(g,g)^{\alpha\beta\eta} \in \mathbb{G}_T$ from an element T that is randomly chosen from $\mathbb{G}_T$.

3. System and Security Model

3.1. System Model

Fig. 1 presents the system model architecture of IoT-based healthcare big data storage system with self-adaptive access control and smart deduplication, which involves the following entities. The characteristic and function of each entity are introduced as follows.

- Key generation center (KGC): KGC is a third trusted entity and responsible to generate the system public parameter. It also creates a master secret key and keeps it confidentially. KGC verifies the medical quality of the medical institutes and generates public/secret keys for them.

- Medical institute (MI): This system can accommodate different medical institutes. A medical institute is responsible to treat the patients, and responsible for managing their patients and medical staffs in its medical domain. A "medical institute" should register to the KGC to obtain the public/secret key pair. Additionally, it assigns the patients and medical staffs a set of attributes to describe their characteristics, and generates attribute secret keys for them. Each medical institute has its own private cloud to enable the storage and computation services on behalf of the users in its management domain, such as patient's emergency contact person list storage, ciphertext re-encryption and break-glass key extraction, etc.

- Data owner (medical-IoT based): The patients always play the role of data owner and are monitored by the medical IoT system. Several tiny wireless sensors are embedded inside or surface-mounted on the skin of patients to formulate a health IoT network. The vital physiology parameters are continuously detected by these sensors and sent to an aggregate node. Then, the patient's electronic healthcare data is included in an electronic medical file. To guarantee the privacy of patient,

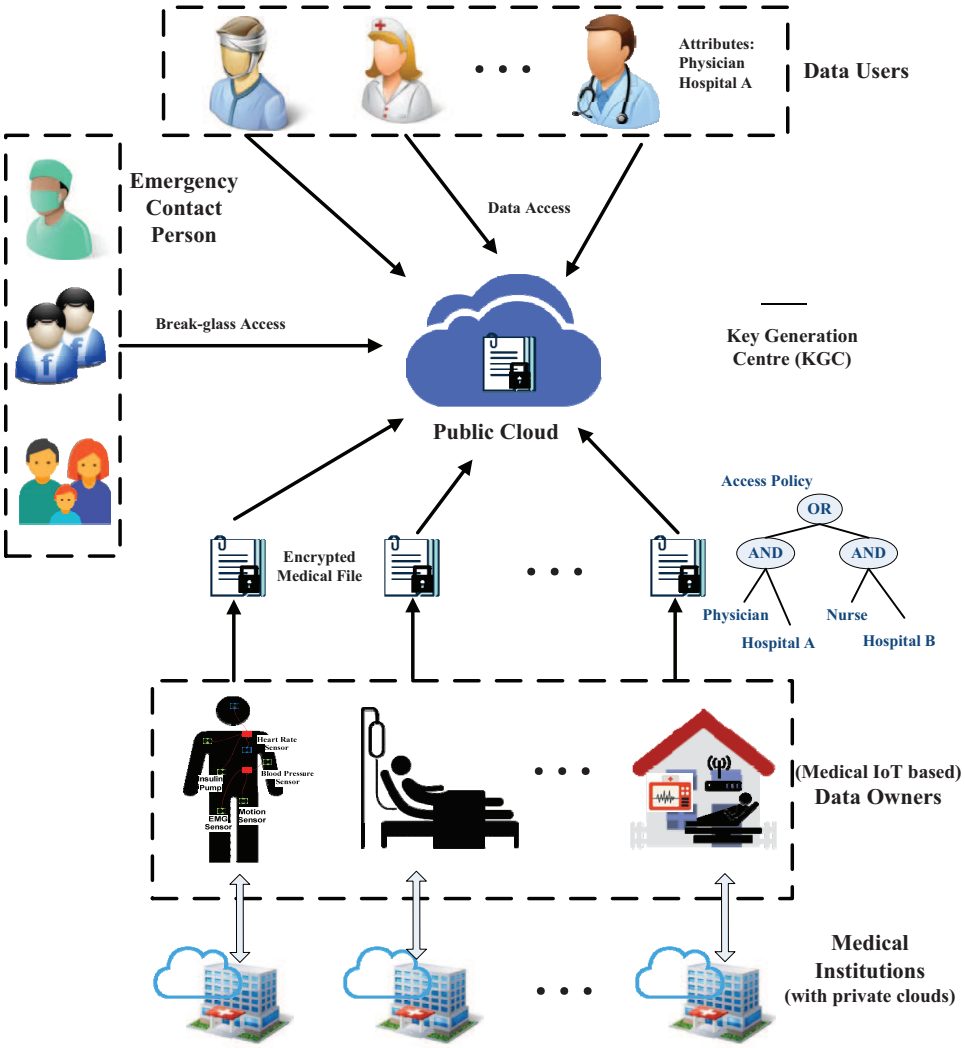


Figure 1: System Model

the medical file is encrypted to a ciphertext, and an access policy is designated for the ciphertext to exert access control such that only authorized users can recover the protected medical files, which are outsourced to the public cloud. To reduce the possibility into emergency situation, the patient pre-sets a password, a break-glass key and a set of emergency contact person (such as his physician-in-charge, family or friends). The break-glass key can decrypt any encrypted medical files of the targeted patient. The patient can tell the emergency contact person (ECP) his password such that the ECP could derive the break-glass key and recover patient's medical files when the patient encounters an emergency situation. The ECP list is managed by the private cloud of the medical institute.

- Public cloud: The public cloud is responsible to store the healthcare big data for different medical institutes and responds on the data access queries. It verifies whether the data user is authorized to access the data according to data user's attributes and the encrypted files' access policy. It also provides partial decryption service to the system users to alleviate the computation burden. In order to eliminate the duplicate copies of encrypted medical files, the public cloud interacts with the medical institute's private cloud to execute deduplication operations to save the storage space.

- Data user: The data user (such as healthcare staffs or patient's friends, relatives) registers to the medical institute to obtain the attribute secret key. Data user sends data access query to the public cloud to get the encrypted medical files, and decrypt them using the attribute secret key.

- Emergency contact person (ECP): The patient pre-shares his password with the ECP. When the patient is in a risky situation, the ECP utilizes the password to derive the break-glass key and decrypts patient's medical files.

3.2. Formal Definition

This privacy-preserving smart IoT-based healthcare big data storage and self-adaptive access control system has thirteen algorithms: global setup algorithm $GlobalSetup$,

medical institute key generation algorithm $KeyGen.MI$, user's attribute key generation algorithm $KeyGen.User$, delegation key generation algorithm $KeyGen.Del$, break-glass key generation algorithm $KeyGen.BGK$, break-glass key extraction algorithm $Extract.BGK$, medical file encryption algorithm Enc, ciphertext validity test algorithm $ValidityTest$, message equality test algorithm $MsgTest$, ciphertext re-encryption algorithm $ReEnc$, partial decryption algorithm $PartialDec$, type-1 decryption algorithm Dec_1 and type-2 decryption algorithm Dec_2.

1. $GlobalSetup(1^\kappa) \rightarrow (PP, MSK)$. The KGC operates the $GlobalSetup$ algorithm. Taken as input the security parameter 1^κ, the global setup algorithm generates the public parameter PP and master secret key MSK for the system. Since PP is an input in all the following algorithms, we omit it to simply the presentation.
2. $KeyGen.MI(MI_i, MSK) \rightarrow (PK_i, SK_i)$. The KGC operates the algorithm $KeyGen.MI$. Taken as input the master secret key MSK and medical institute's identity MI_i, KGC verifies the medical quality of MI_i and generates public/private key pair PK_i/SK_i for MI_i.
3. $KeyGen.User(MI_i, PID_{i,j}, SK_i, \{attr_k\}_{k\in[\varphi]}) \rightarrow SK_{i,j}$. The medical institute MI_i operates the $KeyGen.User$ algorithm. Taken as input the identity MI_i of the medical institute, the user's pseudo identity $PID_{i,j}$, MI_i's secret key SK_i and the user's attributes $\{attr_k\}_{k\in[\varphi]}$, the medical institute MI_i generates the attribute secret key $SK_{i,j}$ for the user.
4. $KeyGen.Del(PID_{i,j}, SK_{i,j}) \rightarrow DK_{i,j}$. The user operates the $KeyGen.Del$ algorithm. Taken as input the user's pseudonym $PID_{i,j}$ and secret key $SK_{i,j}$, the user generates a delegation key $DK_{i,j}$ and sends it to the public cloud.
5. $KeyGen.BGK(PID_{i,j}, pw_{i,j}) \rightarrow (BGK_{i,j,1}, BGK_{i,j,2})$. The patient operates the $KeyGen.BGK$ algorithm. Taken as input the patient's pseudonym $PID_{i,j}$ and the password $pw_{i,j}$, the patient generates a break-glass key $BGK_{i,j}$, a set of emergency contact person, and the auxiliary message $(BGK_{i,j,1}, BGK_{i,j,2})$. The $BGK_{i,j,1}$ is sent to the public cloud. The auxiliary message $BGK_{i,j,2}$ and the emergency contact person list is sent to the private cloud of the medical in-

stitute MI_i.

6. $Extract.BGK(PID_{i,j}, pw_{i,j}, BGK_{i,j,1}, BGK_{i,j,2}) \rightarrow BGK_{i,j}$. This algorithm is interactively executed by the patient's emergency contact person, the public cloud and the medical institute. Taken as input the patient's pseudonym $PID_{i,j}$, the password $pw_{i,j}$ and the auxiliary messages $BGK_{i,j,1}, BGK_{i,j,2}$, the break-glass key extraction algorithm outputs the break-glass key $BGK_{i,j}$.
7. $Enc(M, (\mathbb{A}, \rho, \delta), PK_i, PID_{i,j}, BGK_{i,j}) \rightarrow (CT, pf, TK_{i,j})$. The patient operates the Enc algorithm. Taken as input the medical file M, the access policy $(\mathbb{A}, \rho, \delta)$, the MI_i's public key PK_i, the patient's pseudonym $PID_{i,j}$ and break-glass key $BGK_{i,j}$, the encryption algorithm outputs the encrypted ciphertext CT, a ciphertext proof message pf and a transformation key $TK_{i,j}$. The ciphertext CT and proof message pf are sent to public cloud for remote storage, and the transformation key $TK_{i,j}$ is confidentially sent to the private cloud of the medical institute MI_i.
8. $ValidityTest(CT, pf) \rightarrow 1/0$. The public cloud operates the $ValidityTest$ algorithm. Taken as input the ciphertext CT and the proof pf, the public cloud tests the validity of the ciphertext. If it is valid, the algorithm outputs 1; otherwise, it outputs 0.
9. $MsgTest(pf_1, pf_2) \rightarrow 1/0$. The public cloud operates the $MsgTest$ algorithm. Taken as input the ciphertext proofs pf_1 and pf_2, the public cloud tests whether the two ciphertext are encryptions of the same message. If yes, the algorithm outputs 1; otherwise, it outputs 0.
10. $ReEnc(PDI_{i,j}, TK_{i,j}, CT, (\mathbb{A}', \rho', \delta')) \rightarrow CT'$. The medical institute MI_i operates the $ReEnc$ algorithm. Taken as input the patient's pseudonym $PDI_{i,j}$, the transformation key $TK_{i,j}$, a ciphertext CT and an access policy $(\mathbb{A}', \rho', \delta')$, it outputs another ciphertext CT' which has the same underlying plaintext as CT, but is encrypted under the new access policy $(\mathbb{A}', \rho', \delta')$.
11. $PartialDec(CT, DK_{i,j}) \rightarrow CT_p$. The public cloud operates the $PartialDec$ algorithm. Taken as input the ciphertext CT and the delegation key $DK_{i,j}$, the partial decryption algorithm outputs the partial ciphertext CT_p.
12. $Dec_1(CT_p, SK_{i,j}) \rightarrow M/\perp$. The data user with attribute secret key operates

the Dec_1 algorithm. Taken as input the partial ciphertext $CT_p = (C_M, C_0, C_T)$ and the attribute key $SK_{i,j}$, the Dec_1 recovers M if the partial ciphertext CT_p is correct; otherwise, it outputs $\perp$.

13. $Dec_2(PID_{i,j}, FID, C_M, BGK_{i,j}) \rightarrow M/\perp$. The patient's emergency contact person with break-glass key operates the Dec_2 algorithm. Taken as input the patient's pseudonym $PID_{i,j}$, the file number FID, the ciphertext C_M and the break-glass key $BGK_{i,j}$, the Dec_2 recovers M if the break-glass key $BGK_{i,j}$ is correctly extracted; otherwise, it outputs $\perp$.

3.3. Threat Model

Logically KGC is a fully trusted in the system. The service providers for public cloud server and private cloud server of the medical institute are assumed to be "semi-honest and curious". The service provider of the public cloud is honest to store the encrypted medical big data and execute the pre-defined calculations (such as ciphertext validity test, message equality test and partial decryption operations), but it also curious to get the plaintext of patient's medical message. Moreover, the service provider of the public cloud may be selfish to save its computation resources to return an incorrect transformed ciphertext to the data user. The service provider of the private cloud server of medical institute is honest to store the emergency contact person list for the patients and execute the re-encryption operations using a new access policy in the deduplication process, but is also curious about patients' private medical data. It is assumed that the service provider of the public cloud and the private cloud do not collude with each other. Suppose that the attachers of the system have polynomial time bounded calculation ability. As a result, they cannot solve the hardness problem defined in Section 2.

3.4. Security Model

The system is indistinguishable against chosen plaintext attack (IND-CPA) if there is not polynomial time attacker $\mathcal{A}$ could win the following interactive game with non-negligible advantage.

- Setup: The challenger $\mathcal{C}$ executes $Setup$ algorithm to create the public parameter PP and master secret key MSK. The public parameter PP is sent to the

attacker A and the master secret key MSK is kept secret.

- Query phase 1: The attacker $\mathcal{A}$ adaptively issues the following queries.

  1. $\mathcal{O}_{KeyGen.MI}$: Receiving the key generation query for medical institute MI_i, the challenger $\mathcal{C}$ constructs the public/secret key pair for MI_i by calculating $(PK_i, SK_i) \leftarrow KeyGen.MI$.
  2. $\mathcal{O}_{KeyGen.user}$: Receiving the key generation query for user $u_{i,j}$, the challenger $\mathcal{C}$ constructs the attribute secret key $SK_{i,j}$ by calculating $SK_{i,j} \leftarrow KeyGen.User$.
  3. $\mathcal{O}_{KeyGen.Del}$: Receiving the delegation key generation query for user $u_{i,j}$, the challenger $\mathcal{C}$ constructs the delegation key $DK_{i,j}$ by calculating $DK_{i,j} \leftarrow KeyGen.Del$.
  4. $\mathcal{O}_{KeyGen.BGK}$: Receiving the break-glass key generation query for patient $u_{i,j}$, the challenger $\mathcal{C}$ constructs the break-glass key $BGK_{i,j}$ and its auxiliary message $(BGK_{i,j,1}, BGK_{i,j,2})$ by calculating $(BGK_{i,j,1}, BGK_{i,j,2}) \leftarrow KeyGen.BGK$.
  5. $\mathcal{O}_{Extract.BGK}$: Receiving the break-glass key extraction query for patient $u_{i,j}$ with patient's password $pw_{i,j}$, the challenger $\mathcal{C}$ constructs the break-glass key $BGK_{i,j}$ by calculating $BGK_{i,j} \leftarrow Extract.BGK$.

- Challenge: The adversary $\mathcal{A}$ sends a challenge user $u^*_{i,j}$ (with pseudonym $PID^*_{i,j}$), a challenge access policy $(\mathbb{A}^*, \rho^*, \delta^*)$ and two challenge messages (M^*_0, M^*_1) to $\mathcal{C}$. Then, the challenger $\mathcal{C}$ flips a coin to randomly select $b \in_R \{0, 1\}$. $\mathcal{C}$ constructs ciphertext for M^*_b, which is sent to attacker $\mathcal{A}$.

- Query phase 2: The adversary $\mathcal{A}$ adaptively issues the queries as in phase 1. The requirement is that the attribute secret key or break-glass key for the challenge user $u^*_{i,j}$ (with pseudonym $PID^*_{i,j}$) should not be queried.

- Guess: Adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. If $b' = b$, challenger $\mathcal{C}$ outputs 1 meaning that $\mathcal{A}$ wins the game. Otherwise, $\mathcal{C}$ outputs 0.

4. The proposed system

This system has three distinguished functions: cross-domain data sharing, self-adaptive access control in normal and emergency situations, and smart deduplication.

In order to realize cross-domain data sharing, each medical institute (MI) registers to KGC and obtains the public/secret key pair PK_i/SK_i utilizing $KeyGen.MI$ algorithm. Then, the j-th user (such as patients, doctors and nurses) registers to the i-th MI and obtains its public/secret attribute key pair $PK_{i,j}/SK_{i,j}$ utilizing $KeyGen.User$ algorithm. In the data encryption Enc algorithm, the data owner defines a cross-domain access policy $(\mathbb{A}, \rho, \delta)$, which is embedded in the ciphertext CT. Then, the data users with attributes can satisfy the access policy and execute the decryption Dec_1 algorithm to recover patient's medical record.

The self-adaptive access control is adaptive to the normal and emergency situations, and the normal situation is described above. A password-based break-glass mechanism is designed to realize the access control in emergency situation, which is illustrated mainly in the encryption Enc algorithm, break-glass key generation $KeyGen.BGK$ algorithm, break-glass key extraction $Extract.BGK$ algorithm, and decryption Dec_2 algorithm to be used as follows. Firstly, the patient presets a password and pre-shares it with a set of ECP. $KeyGen.BGK$ algorithm is executed to generate the break-glass key from the password. The auxiliary messages of the break-glass key are also generated in $KeyGen.BGK$ algorithm, which are respectively sent to MI and public cloud. Then, the patient utilizes the break-glass key to encrypt the message in the Enc algorithm. When emergency situation occurs, the ECP of patient interactive with MI and public cloud to extract the break-glass key in the $Extract.BGK$ algorithm. Lastly, ECP utilizes the extracted break-glass key to recover patient's medical files in Dec_2 algorithm.

The secure deduplication effectively discovers the ciphertexts that contain the same plaintext and re-encrypt them into a new ciphertext, which can be accessed by all the original authorized data users. Firstly, public cloud runs ciphertext validity test $ValidityTest$ algorithm to test whether the stored ciphertext is valid. Then, the message equality test $MsgTest$ algorithm is executed to test whether two ciphertexts con-

tain the same plaintext. Lastly, the ciphertext re-encryption $ReEnc$ algorithm can run to generate a new ciphertext with a combined cross-domain access policy.

4.1. System Setup

Taken as input the security parameter 1^κ, the key generation center (KGC) creates the system public parameter PP and the master secret key MSK. The public parameter PP is public in the whole system and MSK is confidentially stored by KGC. PP is a default input in the following algorithms, which is omitted to simply the presentation.

$GlobalSetup(1^\kappa) \rightarrow (PP, MSK)$. KGC operates $GlobalSetup$ algorithm. Taken as input the security parameter 1^κ, KGC randomly selects hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $H_2 : \{0,1\}^* \rightarrow \mathcal{K}$ and cryptographically secure symmetric encryption and decryption pair $SEnc/SDec$ with secret key space $\mathcal{K}$. Then, KGC chooses random numbers $\eta \in_R \mathbb{Z}_p^*$, $g, g_1, g_2, g_3 \in_R \mathbb{G}$ and calculates $Y = e(g_1, g_2)^\eta$. Lastly, KGC sets the public parameter as $PP = (g, g_1, g_2, g_3, Y, H_1, H_2, SEnc/SDec)$ and master secret key as $MSK = \eta$.

Figure 2: System Setup

User $(PID_{i,j})$	Medical Institute (MI_i)	KGC	public cloud
		choose $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$;	
		choose $H_2 : \{0,1\}^* \rightarrow \mathcal{K}$;	
		choose $\eta \in_R \mathbb{Z}_p^*$;	
		choose $g, g_1, g_2, g_3 \in_R \mathbb{G}$;	
		$Y = e(g_1, g_2)^\eta$;	
		$PP = (g, g_1, g_2, g_3, Y, H_1, H_2)$;	
		$MSK = \eta$;	
	$\longleftarrow$	***publish PP***	$\longrightarrow$

4.2. Key Generation for Medical Institute

When a medical institute registers to the system as the i-th medical institute, the KGC checks whether it is a qualified institute. If it is verified, KGC assigns an identity MI_i to the medical institute and generates public key PK_i and secret key SK_i for MI_i. The public key PK_i of MI_i is public in the system and the secret key SK_i is confidentially sent to MI_i via a secure channel.

$KeyGen.MI(MI_i, MSK) \rightarrow (PK_i, SK_i)$. This algorithm is executed by the KGC. Taken as input the master secret key MSK and medical institute's identity MI_i, KGC randomly selects $\alpha_i, \beta_i, \nu_i \in_R \mathbb{Z}_p^*$. Then, KGC calculates the elements of MI_i's public key PK_i as

$$pk_{i,1} = g^{\alpha_i}, pk_{i,2} = g^{\beta_i},$$

and the elements of MI_i's secret key SK_i as

$$K_{i,1} = g_1^{\alpha_i}, K_{i,2} = \beta_i, K_{i,3} = g_2^{\eta} g_3^{\nu_i}, K_{i,4} = g_1^{\nu_i}, K_{i,5} = g_1^{\alpha_i \cdot \nu_i}.$$

Lastly, KGC sets

$$PK_i = (pk_{i,1}, pk_{i,2}), SK_i = (K_{i,1}, K_{i,2}, K_{i,3}, K_{i,4}, K_{i,5}).$$

4.3. Key Generation for User

When a user $U_{i,j}$ registers with medical institute MI_i as the j-th user, the medical institute firstly verifies user's identity. The system user can be a patient, a doctor, a nurse or some other roles. To preserve the privacy of user, MI_i assigns a pseudo identity $PID_{i,j} \in \mathbb{G}$ to the user $U_{i,j}$ and conceals the real identity. According to the user's identity, MI_i assigns a set of attributes $\{attr_k\}_{k \in [\varphi]}$ to describe user's characteristics. Then, MI_i generates the attribute secret key $SK_{i,j}$ for user $PID_{i,j}$.

$KeyGen.User(MI_i, PID_{i,j}, SK_i, \{attr_k\}_{k \in [\varphi]}) \rightarrow SK_{i,j}$. The medical institute MI_i operates the $KeyGen.User$ algorithm. Taken as input the identity MI_i of the medical institute, the user's pseudo identity $PID_{i,j}$, MI_i's secret key SK_i and the user's attributes $\{attr_k\}_{k \in [\varphi]}$, the medical institute MI_i randomly selects $\nu'_{i,j}, t \in_R \mathbb{Z}_p^*$

Figure 3: Key Generation Process for Medical Institute

User $(PID_{i,j})$	Medical Institute (MI_i)	KGC	public cloud
		choose $\alpha_i, \beta_i, \nu_i \in_R \mathbb{Z}_p^*$;	
		$pk_{i,1} = g^{\alpha_i}$;	
		$pk_{i,2} = g^{\beta_i}$;	
		$K_{i,1} = g_1^{\alpha_i}, K_{i,2} = \beta_i$;	
		$K_{i,3} = g_2^{\eta} g_3^{\nu_i}, K_{i,4} = g_1^{\nu_i}$;	
		$K_{i,5} = g_1^{\alpha_i \cdot \nu_i}$;	
		$PK_i = (pk_{i,1}, pk_{i,2})$;	
		$SK_i = (K_{i,1}, K_{i,2}, K_{i,3}, K_{i,4}, K_{i,5})$;	
		$\leftarrow$ ***secretly send*** $\boldsymbol{SK_i}$	
		$\leftarrow$ ***publish*** $\boldsymbol{PK_i}$ $\rightarrow$	

and sets $\nu_{i,j} = \nu_i + \nu'_{i,j}$ for some unknown ν_i. Then, MI_i computes the elements of $PID_{i,j}$'s secret key $SK_{i,j}$ as

$$\begin{aligned}
sk_{i,j,1} &= K_{i,3} \cdot g_3^{\nu'_{i,j}} = g_2^{\eta} g_3^{\nu_{i,j}}, \\
sk_{i,j,2} &= K_{i,4} \cdot g_1^{\nu'_{i,j}} = g_1^{\nu_{i,j}}, \\
sk_{i,j,3} &= g^t, \\
\Gamma_{i,j,k} &= K_{i,5} \cdot K_{i,1}^{\nu'_{i,j}} \cdot g_2^{H_1(PID_{i,j}) \cdot K_{i,2}} \cdot g_3^{H_1(attr_k) \cdot t} \\
&= g_1^{\alpha_i \cdot \nu_{i,j}} \cdot g_2^{H_1(PID_{i,j}) \cdot \beta_i} \cdot g_3^{H_1(attr_k) \cdot t}
\end{aligned}$$

and sets $SK_{i,j} = (sk_{i,j,1}, sk_{i,j,2}, sk_{i,j,3}, \{\Gamma_{i,j,k}\}_{k \in [\varphi]})$.

Figure 4: Key Generation for User

User ($PID_{i,j}$)	Medical Institute (MI_i)	KGC	public cloud
	choose $\nu'_{i,j}, t \in_R \mathbb{Z}_p^*$;		
	set $\nu_{i,j} = \nu_i + \nu'_{i,j}$;		
	$sk_{i,j,1} = K_{i,3} \cdot g_3^{\nu'_{i,j}} = g_2^{\eta} g_3^{\nu_{i,j}}$;		
	$sk_{i,j,2} = K_{i,4} \cdot g_1^{\nu'_{i,j}} = g_1^{\nu_{i,j}}$;		
	$sk_{i,j,3} = g^t$;		
	$\Gamma_{i,j,k} = K_{i,5} \cdot K_{i,1}^{\nu'_{i,j}}$ $\cdot g_2^{H_1(PID_{i,j}) \cdot K_{i,2}}$ $\cdot g_3^{H_1(attr_k) \cdot t}$;		
	$SK_{i,j} = (sk_{i,j,1}, sk_{i,j,2},$ $sk_{i,j,3}, \{\Gamma_{i,j,k}\})$;		
	secretly send $SK_{i,j}$ ⟵		

4.4. Delegation Key Generation

In this phase, the user (with pseudonym $PID_{i,j}$) generates a delegation key $DK_{i,j}$, which is sent to the public cloud. The public cloud utilizes the delegation key to transform the ciphertext such that the user can use a lightweight computation to recover

the medical file. At the same time, the plaintext medical file is unknown to the public cloud.

$KeyGen.Del(PID_{i,j}, SK_{i,j}) \rightarrow DK_{i,j}$. The user operates the $KeyGen.Del$ algorithm. Taken as input the user's pseudonym $PID_{i,j}$ and secret key $SK_{i,j}$, the user chooses random number $\tau \in_R \mathbb{Z}_p^*$ and calculates the elements of the delegation key $DK_{i,j}$ as

$$\begin{aligned}
DK_1 &= (sk_{i,j,1})^\tau = (g_2^\eta g_3^{\nu_{i,j}})^\tau, \\
DK_2 &= (sk_{i,j,2})^\tau = (g_1^{\nu_{i,j}})^\tau, \\
DK_3 &= (sk_{i,j,3})^\tau = (g^t)^\tau, \\
DK_4 &= (g_2^{H_1(PID_{i,j})})^\tau, \\
DK_{5,k} &= (\Gamma_{i,j,k})^\tau = (g_1^{\alpha_i \cdot \nu_{i,j}} \cdot g_2^{H_1(PID_{i,j}) \cdot \beta_i} \cdot g_3^{H_1(attr_k) \cdot t})^\tau.
\end{aligned}$$

Then, user sets the delegation key $DK_{i,j} = (DK_1, DK_2, DK_3, DK_4, \{DK_{5,k}\}_{k \in [\varphi]})$, which is confidentially sent to the public cloud.

Figure 5: Delegated Key Generation

User $(PID_{i,j})$	Medical Institute (MI_i)	KGC	public cloud
choose $\tau \in_R \mathbb{Z}_p^*$;			
$DK_1 = (sk_{i,j,1})^\tau = (g_2^\eta g_3^{\nu_{i,j}})^\tau$;			
$DK_2 = (sk_{i,j,2})^\tau = (g_1^{\nu_{i,j}})^\tau$;			
$DK_3 = (sk_{i,j,3})^\tau = (g^t)^\tau$;			
$DK_4 = (g_2^{H_1(PID_{i,j})})^\tau$;			
$DK_{5,k} = (\Gamma_{i,j,k})^\tau$;			
$DK_{i,j} = (DK_1, DK_2, DK_3, DK_4,$ $\{DK_{5,k}\}_{k \in [\varphi]})$			
	secretly send $DK_{i,j}$ $\longrightarrow$		

4.5. Password based Break-glass Key Generation

In order to provide effective data access when the patient encounters emergency situation (such as suddenly fainted or heart attack), the patient sets a password based break-glass key $BGK_{i,j}$, which can be used to decrypt all the encrypted medical files of the patient. The patient (with pseudonym $PID_{i,j}$) sets a password $pw_{i,j}$ and a break-glass key $BGK_{i,j}$. The patients designates a set of emergency contact person (e.g., his physician-in-charge, family or friends), and secretly tells them the password $pw_{i,j}$. These emergency contact person can deduce the break-glass key using the password $pw_{i,j}$. The emergency contact person list is stored in the private cloud of the patient's medical institute. The patient utilizes the password $pw_{i,j}$ to generate break-glass key $BGK_{i,j}$'s auxiliary messages $(BGK_{i,j,1}, BGK_{i,j,2})$, which are sent to public cloud and MI_i's private cloud, respectively. These auxiliary messages are used to help the emergency contact personals to recover the break-glass key $BGK_{i,j}$ from the password $pw_{i,j}$.

$KeyGen.BGK(PID_{i,j}, pw_{i,j}) \rightarrow (BGK_{i,j,1}, BGK_{i,j,2})$. The patient operates the $KeyGen.BGK$ algorithm. Taken as input the patient's pseudonym $PID_{i,j}$ and the password $pw_{i,j}$, the patient randomly selects $\zeta_1, \zeta_2, \sigma_1, \sigma_2 \in_R \mathbb{Z}_p^*$, $\Psi, \Psi_1 \in_R G$ and sets the break-glass key as $BGK_{i,j} = \Psi$. Then, the patients calculates

$$\begin{aligned} \Psi_2 &= \Psi \cdot (g^{\sigma_1+\sigma_2})^{\varsigma_1} \cdot (\Psi_1)^{-1}, \\ C_{pw_{i,j}} &= (g^{\sigma_1+\sigma_2})^{\varsigma_2} \cdot g_1^{H_1(pw_{i,j})}. \end{aligned}$$

The break-glass key auxiliary messages $(BGK_{i,j,1}, BGK_{i,j,2})$ are calculated as

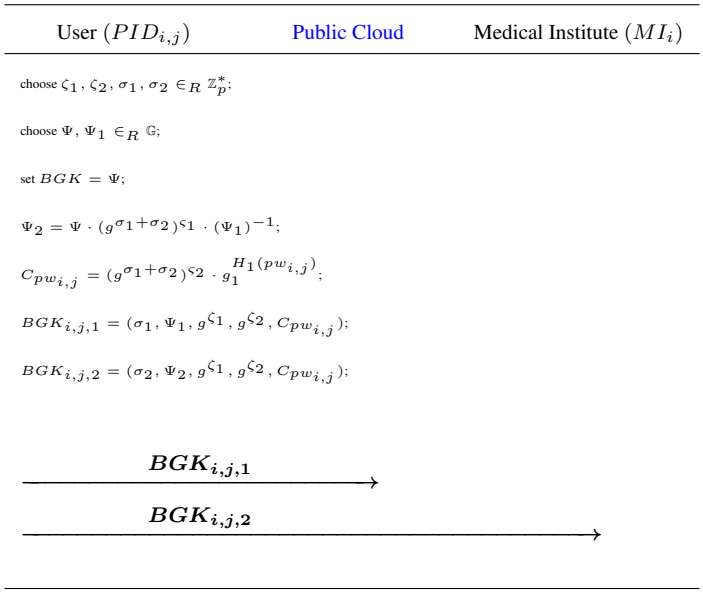
$$\begin{aligned} BGK_{i,j,1} &= (\sigma_1, \Psi_1, g^{\zeta_1}, g^{\zeta_2}, C_{pw_{i,j}}), \\ BGK_{i,j,2} &= (\sigma_2, \Psi_2, g^{\zeta_1}, g^{\zeta_2}, C_{pw_{i,j}}). \end{aligned}$$

Then, $BGK_{i,j,1}$ and $BGK_{i,j,2}$ are secretly sent to public cloud and MI_i, respectively.

4.6. Password based Break-glass Key Extraction

When a patient (with pseudonym $PID_{i,j}$) encounters some emergency situation, it is necessary to quickly access to patient's encrypted medical files to give him first-

Figure 6: Password based Break-glass Key Generation



aid treatments. The medical institute MI_i contacts his designated emergency contact person, who knows patient's password $pw_{i,j}$ corresponding to the break-glass key $BGK_{i,j}$. This emergency contact person interacts with the public cloud and MI_i to recover the break-glass key $BGK_{i,j}$, which can be used to decrypt all of the patient's encrypted medical files. The break-glass key can be extracted using the following algorithm.

$Extract.BGK(PID_{i,j}, pw_{i,j}, BGK_{i,j,1}, BGK_{i,j,2}) \rightarrow BGK_{i,j}$. This algorithm is interactively executed by the patient's emergency contact person, the public cloud and the medical institute. Taken as input the patient's pseudonym $PID_{i,j}$, the password $pw_{i,j}$ and the break-glass key's auxiliary messages $BGK_{i,j,1}, BGK_{i,j,2}$, this algorithm outputs the break-glass key $BGK_{i,j}$. The interactive process is executed as below and also shown in Fig. 7.

- The emergency contact person selects random number $\varrho \in_R \mathbb{Z}_p^*$ and calculates $\Phi = g^{\varrho} \cdot g_1^{H_1(pw_{i,j})}$, which is sent to public cloud and MI_i.
- The public cloud selects random number $a_1 \in_R \mathbb{Z}_p^*$ and calculates $U_1 = (g^{\zeta_2})^{a_1}$,

which is sent to MI_i.

- The medical institute MI_i selects random number $a_2 \in_R \mathbb{Z}_p^*$ and calculates $U_2 = (g^{\zeta_2})^{a_2}$ and transmits it to the public cloud.

- The public cloud computes

$$A_1 = g^{a_1}, W_1 = \Psi_1 \cdot (C_{pw_{i,j}} \cdot \Phi^{-1})^{a_1} \cdot (g^{\zeta_1} \cdot U_1 \cdot U_2)^{-\sigma_1}$$

  and sends (A_1, W_1) to the emergency contact person.

- The medical institute MI_i computes

$$A_2 = g^{a_2}, W_2 = \Psi_2 \cdot (C_{pw_{i,j}} \cdot \Phi^{-1})^{a_2} \cdot (g^{\zeta_1} \cdot U_1 \cdot U_2)^{-\sigma_2}$$

  and sends (A_2, W_2) to the emergency contact person.

- The user recovers the break-glass key by computing

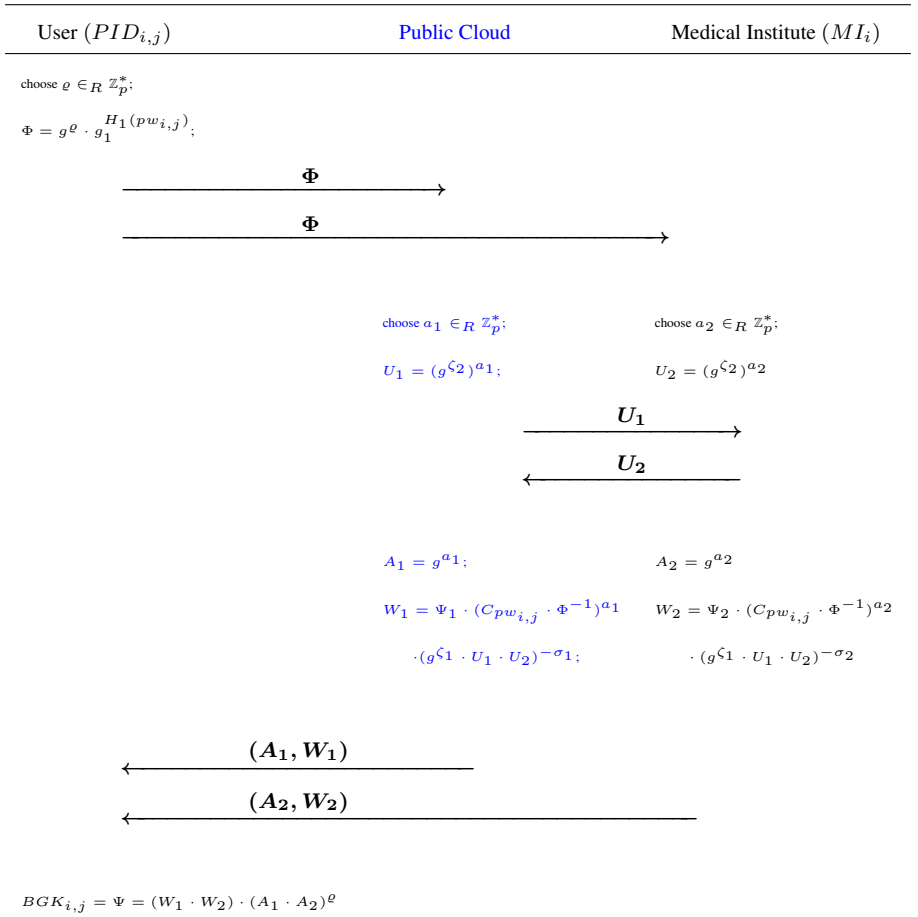
$$BGK_{i,j} = \Psi = (W_1 \cdot W_2) \cdot (A_1 \cdot A_2)^{\varrho}.$$

4.7. Encryption

When the medical record is generated by the medical IoT network, the patient encrypts the message M to the ciphertext and enforce an access policy $(\mathbb{A}, \rho, \delta)$ in the encryption algorithm, which outputs a ciphertext CT, a transformation key $TK_{i,j}$ and a proof message pf. The transformation key $TK_{i,j}$ is utilized in the deduplication algorithm to re-encrypt the a ciphertext using a combined access policy, which will be detailedly introduced in the re-encryption algorithm. The proof message pf helps the public cloud to distinguish the different ciphertext that are the encryption of the same file. The transformation key $TK_{i,j}$ is confidentially sent to MI_i and (CT, pf) are stored by the public cloud.

$Enc(M, (\mathbb{A}, \rho, \delta), PK_i, PID_{i,j}, BGK_{i,j}) \rightarrow (CT, pf, TK_{i,j})$. This algorithm is executed by the patient. The algorithm inputs are the medical file M, the cross-domain access policy $(\mathbb{A}, \rho, \delta)$, the MI_i's public key PK_i, the patient's pseudonym $PID_{i,j}$

Figure 7: Password based Break-glass Key Extraction



and break-glass key $BGK_{i,j}$, where $\mathbb{A} \in \mathbb{Z}_p^{l \times n}$, ρ maps $\mathbb{A}$'s rows to medical institutes and δ maps $\mathbb{A}$'s rows to attributes. Let $\mathbb{A}_x$ be the x-th row of $\mathbb{A}$.

The patient randomly selects $z, v_2, \cdots, v_n, w_2, \cdots, w_n \in_R \mathbb{Z}_p^*$ and sets $\mathbf{v} = (z, v_2, \cdots, v_n)^\top$, $\mathbf{w} = (0, w_2, \cdots, w_n)^\top$. Let $\lambda_x = \langle \mathbb{A}_x, \mathbf{v} \rangle$ and $w_x = \langle \mathbb{A}_x, \mathbf{w} \rangle$, which represent the shares of z and 0 corresponding to row x, respectively. He computes the transformation key $TK_{i,j} = g_3^z$.

The patient randomly selects $t_x \in_R \mathbb{Z}_p^*$ for each row x in $\mathbb{A}$ and chooses $\varpi \in_R \mathbb{Z}^+$. For the electronic medical file M, the patient sets a file number $FID \in \mathbb{G}$ and computes the elements of ciphertext CT as below.

$$
\begin{aligned}
\Upsilon &= H_2(\Psi, PID_{i,j}, FID), \\
C_M &= SEnc(\Upsilon, M||0^{\varpi}), \\
C_{-1} &= g_1^z, \\
C_0 &= \Upsilon \cdot Y^z = \Upsilon \cdot e(g_1, g_2)^{\eta \cdot z}, \\
C_{1,x} &= g_3^{\lambda_x} \cdot (pk_{\rho(x),1})^{t_x} = g_3^{\lambda_x} \cdot g^{\alpha_{\rho(x)} \cdot t_x}, \\
C_{2,x} &= g^{-t_x}, \\
C_{3,x} &= (pk_{\rho(x),2})^{t_x} \cdot g^{w_x} = g^{\beta_{\rho(x)} \cdot t_x} \cdot g^{w_x}, \\
C_{4,x} &= g_3^{H_1(\delta(x)) \cdot t_x},
\end{aligned}
$$

where $M||0^{\varpi}$ denotes ϖ-0s are concatenated after M.

The ciphertext CT is defined as

$$CT = (C_M, C_{-1}, C_0, \{C_{1,x}, C_{2,x}, C_{3,x}, C_{4,x}\}_{x \in [l]}).$$

Then, the patient randomly selects $s, r_1, r_2 \in_R \mathbb{Z}_p^*$ and computes the proof message pf of CT as below.

$$
\begin{aligned}
D_1 &= g^{H_1(M) \cdot s}, \quad D_2 = g^s, \quad D_3 = g^{H_1(M)} \cdot g_1^{H_1(\Upsilon)}, \\
B_1 &= D_2^{r_1}, \quad B_2 = g^{r_1} \cdot g_1^{r_2}, \\
\theta &= H_1(C_M, C_{-1}, C_0, \{C_{1,x}, C_{2,x}, C_{3,x}, C_{4,x}\}_{x \in [l]}, D_1, D_2, D_3, B_1, B_2), \\
D_4 &= r_1 - \theta \cdot H_1(M), \quad D_5 = r_2 - \theta \cdot H_1(\Upsilon).
\end{aligned}
$$

The proof message pf is defined as

$$pf = (D_1, D_2, D_3, D_4, D_5, \theta).$$

Then, the patient outsources $(PID_{i,j}, FID, CT, pf)$ to the public cloud and secretly sends the transformation key $TK_{i,j}$ to the medical institute MI_i.

4.8. Ciphertext Validity Test

In the deduplication process, the public cloud firstly checks whether the stored ciphertext is a valid one. The ciphertext validity text algorithm outputs 1 to indicate that the ciphertext is valid; otherwise, it outputs 0.

$ValidityTest(CT, pf) \rightarrow 1/0$. The public cloud operates the $ValidityTest$ algorithm. Taken as input the ciphertext CT and the proof pf, the public cloud calculates

$$\begin{aligned} B_1' &= D_1^{\theta} \cdot D_2^{D_4}, B_2' = D_3^{\theta} \cdot g^{D_4} \cdot g_1^{D_5}, \\ \theta' &= H_1(C_M, C_{-1}, C_0, \{C_{1,x}, C_{2,x}, C_{3,x}, C_{4,x}\}_{x \in [l]}, D_1, D_2, D_3, B_1', B_2'), \end{aligned}$$

and verifies whether the equation holds $\theta' = \theta$. If it holds, the algorithm outputs 1; otherwise, it outputs 0.

4.9. Message Equality Test

If the two ciphertext are verified to be valid by the validity test algorithm, the message equality test algorithm verifies whether they are the encryption of the same message. If yes, the algorithm outputs 1; otherwise, it outputs 0.

$MsgTest(pf_1, pf_2) \rightarrow 1/0$. The public cloud operates the $MsgTest$ algorithm. Taken as input the ciphertext proofs pf_1 and pf_2, the public cloud parses the proofs as $pf_1 = (D_1, D_2, D_3, D_4, D_5, \theta)$ and $pf_2 = (D_1', D_2', D_3', D_4', D_5', \theta')$. It tests whether the equation $e(D_1, D_2') = e(D_1', D_2)$ holds. If it holds, the algorithm outputs 1 to indicate that the two ciphertext are the encryption of the same message; otherwise, it outputs 0.

4.10. Ciphertext Re-encryption

If a set of ciphertext are verified to contain the same message and belongs to the same data owner, they will be deduplicated. Suppose the ciphertext and the corresponding access policies are $(CT_1, (\mathbb{A}_1, \rho_1, \delta_1)), \cdots, (CT_m, (\mathbb{A}_m, \rho_m, \delta_m))$. The medical institute MI_i firstly combines these access policies as $(\mathbb{A}', \rho', \delta')$, which is a union set of the access policies $((\mathbb{A}_1, \rho_1, \delta_1), \cdots, (\mathbb{A}_m, \rho_m, \delta_m))$. Then, MI_i re-encrypts the ciphertext using the combined cross-domain access policy $(\mathbb{A}', \rho', \delta')$ to generate a new ciphertext CT'. In that way, any pre-defined authorized data user of the ciphertext $(CT_1, \cdots, CT_m)$ can access to the new ciphertext CT'. Denote the ciphertext with the least file number in $(CT_1, \cdots, CT_m)$ as CT. Suppose $CT = CT_1$ and $FID = FID_1$.

$ReEnc(PDI_{i,j}, TK_{i,j}, CT, (\mathbb{A}', \rho', \delta')) \rightarrow CT'$. The medical institute MI_i operates the $ReEnc$ algorithm. The inputs of the re-encryption algorithm are the patient's pseudonym $PDI_{i,j}$, the transformation key $TK_{i,j}$, a ciphertext CT and a combined access policy $(\mathbb{A}', \rho', \delta')$, where $\mathbb{A}' \in \mathbb{Z}_p^{l' \times n'}$, ρ' maps the rows of $\mathbb{A}'$ to medical institutes and δ' maps the rows of $\mathbb{A}'$ to attributes. Let $\mathbb{A}'_x$ be the x-th row of $\mathbb{A}'$, and $\mathbb{A}'_x = (a'_{x,1}, \cdots, a'_{x,n'})$.

The medical institute MI_i randomly selects $\bar{z}, v'_2, \cdots, v'_n, w'_2, \cdots, w'_n \in_R \mathbb{Z}_p^*$ and sets $\mathbf{v}' = (z', v'_2, \cdots, v'_n)^\top$, $\mathbf{w}' = (0, w'_2, \cdots, w'_n)$, where $z' = z + \bar{z}$ for some unknown z. Let $\lambda'_x = \langle \mathbb{A}'_x, \mathbf{v}' \rangle$ and $w'_x = \langle \mathbb{A}'_x, \mathbf{w}' \rangle$, which represent the shares of z' and 0 corresponding to row x, respectively. The medical institute MI_i selects $t'_x \in_R \mathbb{Z}_p^*$ for each row x of $\mathbb{A}'$. The re-encrypted ciphertext

$$CT' = (C'_M, C'_{-1}, C'_0, \{C'_{1,x}, C'_{2,x}, C'_{3,x}, C'_{4,x}\}_{x \in [l]})$$

is constructed as below.

$$\begin{aligned}
C'_M &= C_M, \\
C'_{-1} &= C_{-1} \cdot g_1^{\bar{z}} = g_1^{z'}, \\
C'_0 &= C_0 \cdot Y^{\bar{z}} = \Upsilon \cdot Y^{z'}, \\
C'_{1,x} &= g_3^{\langle \mathbb{A}'_x, \mathbf{v}' \rangle} \cdot (pk_{\rho(x),1})^{t'_x} = g_3^{\lambda'_x} \cdot g^{\alpha_{\rho(x)} \cdot t'_x}, \\
C'_{2,x} &= g^{-t'_x}, \\
C'_{3,x} &= (pk_{\rho(x),2})^{t'_x} \cdot g^{w'_x} = g^{\beta_{\rho(x)} \cdot t'_x} \cdot g^{w'_x}, \\
C'_{4,x} &= g_3^{H_1(\delta(x)) \cdot t'_x}.
\end{aligned}$$

Since $\mathbb{A}'_x = (a'_{x,1}, \cdots, a'_{x,n'})$ and $\lambda'_x = \langle \mathbb{A}'_x, \mathbf{v}' \rangle$, then we have $\lambda'_x = z' a'_{x,1} + \cdots + v'_{n'} a'_{x,n'} = z a'_{x,1} + \bar{z} a'_{x,1} + \cdots + v'_{n'} a'_{x,n'}$. The element $C'_{1,x}$ is actually computed as below using the transformation key $TK_{i,j}$.

$$\begin{aligned}
C'_{1,x} &= g_3^{\langle \mathbb{A}'_x, \mathbf{v}' \rangle} \cdot (pk_{\rho(x),1})^{t'_x} = g_3^{\lambda'_x} \cdot (pk_{\rho(x),1})^{t'_x} \\
&= g_3^{z a'_{x,1} + \bar{z} a'_{x,1} + \cdots + v'_{n'} a'_{x,n'}} \cdot (pk_{\rho(x),1})^{t'_x} \\
&= (g_3^{z})^{a'_{x,1}} \cdot g_1^{\bar{z} a'_{x,1} + \cdots + v'_{n'} a'_{x,n'}} \cdot (pk_{\rho(x),1})^{t'_x} \\
&= (TK_{i,j})^{a'_{x,1}} \cdot g_1^{\bar{z} a'_{x,1} + \cdots + v'_{n'} a'_{x,n'}} \cdot (pk_{\rho(x),1})^{t'_x}.
\end{aligned}$$

It is obvious that the distribution of CT' is consistent with the that in Enc algorithm.

4.11. Partial Decryption

In order to alleviate the computation burden of data user, the public cloud utilizes the delegation key $DK_{i,j}$ to partially decrypt the ciphertext and generates a partial ciphertext CT_p. During the process, the plaintext M is unknown to the public cloud.

$PartialDec(CT, DK_{i,j}) \rightarrow CT_p$. The public cloud operates the $PartialDec$ algorithm. Taken as input the ciphertext CT and the delegation key $DK_{i,j}$, the public cloud computes $c_x \in \mathbb{Z}_p$ such that $\sum_x c_x \cdot A_x = (1, 0, \cdots, 0)$. Then, it calculates

$$\begin{aligned}
C_T &= \frac{\prod_x [e(C_{1,x}, DK_2) \cdot e(C_{2,x}, DK_{5,\delta(x)}) \cdot e(C_{3,x}, DK_4) \cdot e(C_{4,x}, DK_3)]^{c_x}}{e(C_{-1}, DK_1)} \\
&= e(g_1, g_2)^{-\eta \cdot z \cdot \tau},
\end{aligned}$$

and sends the partial ciphertext $CT_p = (C_M, C_0, C_T)$ to data user.

4.12. Decryption with Attribute Key and Verification

The data user with attribute key $SK_{i,j}$ recovers the medical file M using the decryption algorithm Dec_1.

$Dec_1(CT_p, SK_{i,j}) \rightarrow M/\perp$. This algorithm is executed by the data user with attribute secret key. Taken as input the partial ciphertext $CT_p = (C_M, C_0, C_T)$ and the attribute key $SK_{i,j}$, the data user recovers $\Upsilon = C_0 \cdot (C_T)^{\tau^{-1}}$, $M' = SDec(H_2(\Upsilon), C_M)$. If $M' = M||0^{\varpi}$, it indicates that the partial decryption executed by the public cloud is correct and the Dec_1 algorithm outputs the medical file M; otherwise, it outputs $\perp$.

4.13. Decryption with Break-glass Key and Verification

In emergency situation, the patient's emergency contact person extracts the break-glass key $BGK_{i,j}$ using the $Extract.BGK$ algorithm. Then, he decrypts the patients' medical files utilizing the decryption algorithm Dec_2.

$Dec_2(PID_{i,j}, FID, C_M, BGK_{i,j}) \rightarrow M/\perp$. This algorithm is executed by the patient's emergency contact person with break-glass key. Taken as input the patient's pseudonym $PID_{i,j}$, the file number FID, the ciphertext C_M and the break-glass key $BGK_{i,j}$, the emergency contact person recovers $\Upsilon = H_2(BGK_{i,j}, PID_{i,j}, FID)$ and $M' = SDec(H_2(\Upsilon), C_M)$. If $M' = M||0^{\varpi}$, it indicates that the break-glass key $BGK_{i,j}$ is correctly extracted and the Dec_1 algorithm outputs the medical file M; otherwise, it outputs $\perp$.

5. Correctness

5.1. Ciphertext Validity Test Correctness

If the ciphertext is valid with well-formed ciphertext

$$CT = (C_M, C_{-1}, C_0, \{C_{1,x}, C_{2,x}, C_{3,x}, C_{4,x}\}_{x \in [l]})$$

and proof message $pf = (D_1, D_2, D_3, D_4, D_5, \theta)$, the the correctness of ciphertext validity test algorithm can be verified as below.

$$\begin{aligned} B_1' &= D_1^{\theta} \cdot D_2^{D_4} = (g^{H_1(M)\cdot s})^{\theta} \cdot (g^s)^{r_1 - \theta \cdot H_1(M)} \\ &= g^{sr_1} = D_2^{r_1} = B_1, \\ B_2' &= D_3^{\theta} \cdot g^{D_4} \cdot g_1^{D_5} = (g^{H_1(M)} \cdot g_1^{H_1(\Upsilon)})^{\theta} \cdot g^{r_1 - \theta \cdot H_1(M)} \cdot g_1^{r_2 - \theta \cdot H_1(\Upsilon)} \\ &= g^{r_1} \cdot g_1^{r_2} = B_2, \end{aligned}$$

where $\theta = H_1(C_M, C_{-1}, C_0, \{C_{1,x}, C_{2,x}, C_{3,x}, C_{4,x}\}_{x\in[l]}, D_1, D_2, D_3, B_1, B_2)$.

5.2. Message Equality Test Correctness

Suppose the ciphertext is verified valid by the ciphertext validity test algorithm. Then, whether the two ciphertext are encrypted by the same message can be tested, since

$$e(D_1, D_2') = e(g^{H_1(M)\cdot s}, g^{s'}) = e(g^{H_1(M)\cdot s'}, g^s) = e(D_1', D_2).$$

5.3. Password based Break-glass Key Extraction Correctness

The correctness of the password based break-glass key extraction algorithm can be verified as below.

$$\begin{aligned} & (W_1 \cdot W_2) \cdot (A_1 \cdot A_2)^{\varrho} \\ = \; & [\Psi_1 \cdot (C_{pw_{i,j}} \cdot \Phi^{-1})^{a_1} \cdot (g^{\zeta_1} \cdot U_1 \cdot U_2)^{-\sigma_1}] \cdot \\ & [\Psi_2 \cdot (C_{pw_{i,j}} \cdot \Phi^{-1})^{a_2} \cdot (g^{\zeta_1} \cdot U_1 \cdot U_2)^{-\sigma_2}] \cdot (g^{a_1} \cdot g^{a_2})^{\varrho} \\ = \; & (\Psi_1 \cdot \Psi_2) \cdot (C_{pw_{i,j}} \cdot \Phi^{-1})^{a_1 + a_2} \\ & \cdot (g^{\zeta_1} \cdot U_1 \cdot U_2)^{-(\sigma_1 + \sigma_2)} \cdot (g^{a_1} \cdot g^{a_2})^{\varrho} \\ = \; & [\Psi \cdot (g^{\sigma_1 + \sigma_2})^{\varsigma_1}] \\ & \cdot [(g^{\sigma_1 + \sigma_2})^{\varsigma_2} \cdot g_1^{H_1(pw_{i,j})} \cdot (g^{\varrho} \cdot g_1^{H_1(pw_{i,j})})^{-1}]^{a_1 + a_2} \\ & \cdot [g^{\zeta_1} \cdot (g^{\zeta_2})^{a_1 + a_2}]^{-(\sigma_1 + \sigma_2)} \cdot (g^{a_1} \cdot g^{a_2})^{\varrho} \\ = \; & [\Psi \cdot (g^{\sigma_1 + \sigma_2})^{\varsigma_1}] \cdot (g^{(\sigma_1 + \sigma_2)\cdot \varsigma_2} \cdot g^{-\varrho})^{a_1 + a_2} \\ & \cdot [g^{\zeta_1} \cdot (g^{\zeta_2})^{a_1 + a_2}]^{-(\sigma_1 + \sigma_2)} \cdot g^{(a_1 + a_2)\cdot \varrho} \\ = \; & \Psi. \end{aligned}$$

5.4. Decryption Correctness

The correctness of the decryption algorithms Dec_1 and Dec_2 are obvious. We only needs to verify the correctness of partial decryption algorithm.

$$
\begin{aligned}
& e(C_{1,x}, DK_2) \cdot e(C_{2,x}, DK_{5,\delta(x)}) \cdot e(C_{3,x}, DK_4) \cdot e(C_{4,x}, DK_3) \\
= \; & e[g_3^{\lambda_x} \cdot g^{\alpha_{\rho(x)} \cdot t_x}, (g_1^{\nu_{i,j}})^{\tau}] \cdot e[g^{-t_x}, (g_1^{\alpha_{\rho(x)} \cdot \nu_{i,j}} \cdot g_2^{H_1(PID_{i,j}) \cdot \beta_i} \cdot g_3^{H_1(\delta(x)) \cdot t})^{\tau}] \\
& \cdot e[g^{\beta_{\rho(x)} \cdot t_x} \cdot g^{w_x}, (g_2^{H_1(PID_{i,j})})^{\tau}] \cdot e[g_3^{H_1(\delta(x)) \cdot t_x}, (g^t)^{\tau}] \\
= \; & e(g_1, g_3)^{\nu_{i,j} \cdot \lambda_x \cdot \tau} \cdot e(g, g_1)^{\alpha_{\rho(x)} \cdot t_x \cdot \nu_{i,j} \cdot \tau} \\
& \cdot e(g, g_1)^{-\alpha_{\rho(x)} \cdot t_x \cdot \nu_{i,j} \cdot \tau} \cdot e[g^{-t_x}, g_2^{H_1(PID_{i,j}) \cdot \beta_i \cdot \tau}] \cdot e[g^{-t_x}, g_3^{H_1(\delta(x)) \cdot t \cdot \tau}] \\
& \cdot e[g^{\beta_{\rho(x)} \cdot t_x}, g_2^{H_1(PID_{i,j}) \cdot \tau}] \cdot e[g^{w_x}, g_2^{H_1(PID_{i,j}) \cdot \tau}] \cdot e[g_3^{H_1(\delta(x)) \cdot t_x}, g^{t \cdot \tau}] \\
= \; & e(g_1, g_3)^{\nu_{i,j} \cdot \lambda_x \cdot \tau} \cdot e(g_2^{H_1(PID_{i,j})}, g)^{w_x \cdot \tau},
\end{aligned}
$$

$$
\begin{aligned}
& \prod_x [e(C_{1,x}, DK_2) \cdot e(C_{2,x}, DK_{5,\delta(x)}) \cdot e(C_{3,x}, DK_4) \cdot e(C_{4,x}, DK_3)]^{c_x} \\
= \; & \prod_x [e(g_1, g_3)^{\nu_{i,j} \cdot \lambda_x \cdot \tau} \cdot e(g_2^{H_1(PID_{i,j})}, g)^{w_x \cdot \tau}]^{c_x} \\
= \; & e(g_1, g_3)^{\nu_{i,j} \cdot z \cdot \tau},
\end{aligned}
$$

$$
e(C_{-1}, DK_1) = e[g_1^z, (g_2^{\eta} g_3^{\nu_{i,j}})^{\tau}] = e(g_1, g_2)^{\eta \cdot z \cdot \tau} \cdot e(g_1, g_3)^{\nu_{i,j} \cdot z \cdot \tau},
$$

$$
\begin{aligned}
& \frac{\prod_x [e(C_{1,x}, DK_2) \cdot e(C_{2,x}, DK_{5,\delta(x)}) \cdot e(C_{3,x}, DK_4) \cdot e(C_{4,x}, DK_3)]^{c_x}}{e(C_{-1}, DK_1)} \\
= \; & e(g_1, g_2)^{-\eta \cdot z \cdot \tau}.
\end{aligned}
$$

6. Security Analysis

Theorem 1. This system is IND-CPA secure if the DBDH problem is intractable in polynomial time.

Proof: Assume that there exists a probabilistic polynomial time (PPT) attacker $\mathcal{A}$ to undermine the proposed system's security, then there exists a PPT algorithm $\mathcal{C}$ to

solve the DBDH problem. The challenger $\mathcal{C}$ receives the tuple $(g, g^{\alpha}, g^{\beta}, g^{\eta}, T)$ from the decisional bilinear Diffie-Hellman assumption.

- **Setup.** The challenger $\mathcal{C}$ has to construct the public parameters for the system utilizing the DBDH tuple. $\mathcal{C}$ randomly selects $g_3 \in \mathbb{G}$ and sets $g_1 = g^{\alpha}, g_2 = g^{\beta}, Y = T$. $\mathcal{C}$ randomly selects hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $H_2 : \{0,1\}^* \rightarrow \mathcal{K}$ and cryptographically secure symmetric encryption and decryption pair $SEnc/SDec$ with secret key space $\mathcal{K}$. Then, $\mathcal{C}$ sends the public parameter $PP = (g, g_1, g_2, g_3, Y, H_1, H_2, SEnc/SDec)$ to the attacker $\mathcal{A}$.

- Query phase 1: The adversary $\mathcal{A}$ adaptively issues the following queries.

  1. $\mathcal{O}_{KeyGen.MI}$: Receiving a key generation query for medical institute MI_i, the challenger $\mathcal{C}$ randomly chooses $\alpha_i, \beta_i, \nu_i \in_R \mathbb{Z}_p^*$ and compute $pk_{i,1} = g^{\alpha_i}, pk_{i,2} = g^{\beta_i}$ and $K_{i,1} = g_1^{\alpha_i}, K_{i,2} = \beta_i, K_{i,3} = g_2^{\eta} g_3^{\nu_i}, K_{i,4} = g_1^{\nu_i}, K_{i,5} = g_1^{\alpha_i \cdot \nu_i}$. $\mathcal{C}$ sets the public key of MI_i as $PK_i = (pk_{i,1}, pk_{i,2})$ and the secret key of MI_i as $SK_i = (K_{i,1}, K_{i,2}, K_{i,3}, K_{i,4}, K_{i,5})$, which are then sent to the attacker $\mathcal{A}$.
  2. $\mathcal{O}_{KeyGen.User}$: Receiving an attribute key generation query for user $u_{i,j}$ with pseudonym $PID_{i,j}$ and a set of attributes $\{attr_k\}_{k \in [\varphi]}$, the challenger $\mathcal{C}$ randomly selects $\nu_{i,j}, t \in_R \mathbb{Z}_p^* \in_R \mathbb{Z}_p^*$ and compute $sk_{i,j,1} = g_2^{\eta} g_3^{\nu_{i,j}}$, $sk_{i,j,2} = g_1^{\nu_{i,j}}$, $sk_{i,j,3} = g^t$ and $\Gamma_{i,j,k} = g_1^{\alpha_i \cdot \nu_{i,j}} \cdot g_2^{H_1(PID_{i,j}) \cdot \beta_i} \cdot g_3^{H_1(attr_k) \cdot t}$ for $k \in [\varphi]$. Then $\mathcal{C}$ sends the constructed attribute secret key $SK_{i,j} = (sk_{i,j,1}, sk_{i,j,2}, sk_{i,j,3}, \{\Gamma_{i,j,k}\}_{k \in [\varphi]})$ to the attacker $\mathcal{A}$.
  3. $\mathcal{O}_{KeyGen.Del}$: Receiving the delegation key generation query for user $u_{i,j}$ with pseudonym $PID_{i,j}$, the challenger $\mathcal{C}$ firstly construct the attribute secret key $SK_{i,j} = (sk_{i,j,1}, sk_{i,j,2}, sk_{i,j,3}, \{\Gamma_{i,j,k}\}_{k \in [\varphi]})$ using $\mathcal{O}_{KeyGen.User}$. Then, $\mathcal{C}$ chooses random number $\tau \in_R \mathbb{Z}_p^*$ and calculates $DK_1 = (sk_{i,j,1})^{\tau}$, $DK_2 = (sk_{i,j,2})^{\tau}$, $DK_3 = (sk_{i,j,3})^{\tau}$, $DK_4 = (g_2^{H_1(PID_{i,j})})^{\tau}$, $DK_{5,k} = (\Gamma_{i,j,k})^{\tau}$ for $k \in [\varphi]$. $\mathcal{C}$ sends the delegation key

     $$DK_{i,j} = (DK_1, DK_2, DK_3, DK_4, \{DK_{5,k}\}_{k \in [\varphi]})$$

     to the attacker $\mathcal{A}$.

4. $\mathcal{O}_{KeyGen.BGK}$: Receiving the break-glass key generation query for patient $u_{i,j}$, the challenger $\mathcal{C}$ randomly selects a password $pw_{i,j}$, random numbers $\zeta_1, \zeta_2, \sigma_1, \sigma_2 \in_R \mathbb{Z}_p^*$, $\Psi, \Psi_1 \in_R \mathbb{G}$ and sets the break-glass key as $BGK_{i,j} = \Psi$. Then, $\mathcal{C}$ calculates $\Psi_2 = \Psi \cdot (g^{\sigma_1+\sigma_2})^{\varsigma_1} \cdot (\Psi_1)^{-1}$, $C_{pw_{i,j}} = (g^{\sigma_1+\sigma_2})^{\varsigma_2} \cdot g_1^{H_1(pw_{i,j})}$. The break-glass key auxiliary messages $(BGK_{i,j,1}, BGK_{i,j,2})$ are calculated as $BGK_{i,j,1} = (\sigma_1, \Psi_1, g^{\zeta_1}, g^{\zeta_2}, C_{pw_{i,j}})$, $BGK_{i,j,2} = (\sigma_2, \Psi_2, g^{\zeta_1}, g^{\zeta_2}, C_{pw_{i,j}})$. Then, $BGK_{i,j,1}$ and $BGK_{i,j,2}$ are sent to the attacker $\mathcal{A}$.
5. $\mathcal{O}_{Extract.BGK}$: Receiving the break-glass key extraction query for patient $u_{i,j}$ with patient's password $pw_{i,j}$, the challenger $\mathcal{C}$ constructs the break-glass key $BGK_{i,j}$ for patient $u_{i,j}$ using $\mathcal{O}_{KeyGen.BGK}$ and sents it to the attacker $\mathcal{A}$.

- Challenge: The adversary $\mathcal{A}$ sends a challenge user $u_{i,j}^*$ (with pseudonym $PID_{i,j}^*$), a challenge access policy $(\mathbb{A}^*, \rho^*, \delta^*)$ and two challenge messages (M_0^*, M_1^*) to $\mathcal{C}$, where $\mathbb{A}^* \in \mathbb{Z}_p^{l^* \times n^*}$, ρ^* maps the rows of $\mathbb{A}^*$ to medical institutes and δ^* maps the rows of $\mathbb{A}^*$ to attributes. Then, the challenger $\mathcal{C}$ flips a coin to randomly select $b \in_R \{0, 1\}$. Let $\mathbb{A}_x^*$ be the x-th row of $\mathbb{A}^*$.

  The challenger $\mathcal{C}$ randomly chooses $z^*, v_2^*, \cdots, v_n^*, w_2^*, \cdots, w_n^* \in_R \mathbb{Z}_p^*$ and sets $\mathbf{v}^* = (z^*, v_2^*, \cdots, v_n^*)^\top$, $\mathbf{w}^* = (0, w_2^*, \cdots, w_n^*)^\top$. Let $\lambda_x^* = \langle \mathbb{A}_x^*, \mathbf{v}^* \rangle$ and $w_x^* = \langle \mathbb{A}_x^*, \mathbf{w}^* \rangle$, which represent the shares of z^* and 0 corresponding to row x, respectively. $\mathcal{C}$ computes the transformation key $TK_{i,j}^* = g_3^{z^*}$.

  $\mathcal{C}$ randomly selects $t_x^* \in_R \mathbb{Z}_p^*$ for each row x of $\mathbb{A}^*$ and chooses random $\varpi^* \in_R \mathbb{Z}^+$. The challenger $\mathcal{C}$ constructs the break-glass key $BGK_{i,j}^*$ for patient $u_{i,j}^*$ using $\mathcal{O}_{KeyGen.BGK}$. For the electronic medical file M_b^*, the patient sets a file number $FID^* \in \mathbb{G}$ and computes the elements of ciphertext CT^* as below.

$$
\begin{aligned}
\Upsilon^* &= H_2(BGK^*_{i,j}, PID^*_{i,j}, FID^*),\\
C^*_{M^*_b} &= SEnc(\Upsilon^*, M^*_b||0^{\varpi^*}),\\
C^*_{-1} &= g_1^{z^*},\\
C^*_0 &= \Upsilon^* \cdot Y^{z^*} = \Upsilon^* \cdot T^{z^*},\\
C^*_{1,x} &= g_3^{\lambda^*_x} \cdot (pk_{\rho^*(x),1})^{t^*_x} = g_3^{\lambda^*_x} \cdot g^{\alpha_{\rho^*(x)} \cdot t^*_x},\\
C^*_{2,x} &= g^{-t^*_x},\\
C^*_{3,x} &= (pk_{\rho^*(x),2})^{t^*_x} \cdot g^{w^*_x} = g^{\beta_{\rho^*(x)} \cdot t^*_x} \cdot g^{w^*_x},\\
C^*_{4,x} &= g_3^{H_1(\delta^*(x)) \cdot t^*_x},
\end{aligned}
$$

where $M^*_b||0^{\varpi^*}$ denotes ϖ^*-0s are concatenated after M^*_b.

The constructed challenge ciphertext is

$$CT^* = (C^*_{M^*_b}, C^*_{-1}, C^*_0, \{C^*_{1,x}, C^*_{2,x}, C^*_{3,x}, C^*_{4,x}\}_{x \in [l^*]}).$$

Then, the challenger $\mathcal{C}$ randomly selects $s^*, r^*_1, r^*_2 \in_R \mathbb{Z}^*_p$ and computes the proof message pf^* of CT^* as below.

$$
\begin{aligned}
D^*_1 &= g^{H_1(M^*_b) \cdot s^*}, \quad D^*_2 = g^{s^*}, \quad D^*_3 = g^{H_1(M^*_b)} \cdot g_1^{H_1(\Upsilon^*)},\\
B^*_1 &= (D^*_2)^{r^*_1}, \quad B^*_2 = g^{r^*_1} \cdot g_1^{r^*_2},\\
\theta^* &= H_1(C^*_{M^*_b}, C^*_{-1}, C^*_0, \{C^*_{1,x}, C^*_{2,x}, C^*_{3,x}, C^*_{4,x}\}_{x \in [l^*]},\\
&\qquad D^*_1, D^*_2, D^*_3, B^*_1, B^*_2),\\
D^*_4 &= r^*_1 - \theta^* \cdot H_1(M^*_b), \quad D^*_5 = r^*_2 - \theta^* \cdot H_1(\Upsilon).
\end{aligned}
$$

The proof message pf^* is defined as

$$pf^* = (D^*_1, D^*_2, D^*_3, D^*_4, D^*_5, \theta^*).$$

Then, the challenger $\mathcal{C}$ sends $CT^*, pf^*, TK^*_{i,j}$ to the attacker $\mathcal{A}$.

- Query phase 2: The adversary $\mathcal{A}$ adaptively issues the queries as in phase 1 with the restriction that the attribute secret key or break-glass key for the challenge user $u^*_{i,j}$ (with pseudonym $PID^*_{i,j}$) should not be queried.

- Guess: Adversary $\mathcal{A}$ outputs a guess $b_1 \in \{0, 1\}$. If $b' = b$, challenger $\mathcal{C}$ outputs 1 meaning that $T = e(g, g)^{\alpha\beta\eta}$. Otherwise, $\mathcal{C}$ outputs 0 meaning that T is a random element in $\mathbb{G}_T$.

- ***Probability Analysis.*** Suppose $\mathcal{A}$ has advantage ε in attacking decisional DBDH assumption and $\mathcal{C}$ has advantage ε' in winning this game. Then, it is easy to know that $\varepsilon' = \varepsilon$.

7. Performance Analysis

The utility function and efficiency are two important factors to evaluate a system. In this section, we compare the proposed system with other related schemes in terms of functionality, storage and computation costs. Then, these schemes are implemented on an experimental test-bed to measure their performances.

7.1. Comparison

Shown in Table 2, the function of our system is compared with other schemes with break-glass access [6, 7, 24, 25, 35], the schemes with secure deduplication [3, 29, 17, 18, 20, 4, 11], the attribute based encryption schemes [13, 16, 28, 23, 27, 21, 26, 19, 30, 31] and the multi-authority ABE schemes [14, 15, 34, 22]. It is obvious that this proposed system demonstrates versatile useful functions compared with the other existing schemes.

- ***Attribute based encryption***: ABE is an important method to realize fine-grained access control over encrypted data, where user's private keys and the encrypted files are associated with attributes or access policy. When the pre-set attributes satisfies the access policy, the user is authorized to access to the plaintext. However, the schemes in [6, 24, 25, 35, 3, 29, 17, 18, 20, 4, 11] fail to support this kind of access control pattern. The other schemes and our system enables ABE mechanism.

- ***Cross-domain***: In the electronic medical system, there are many different medical institutes and the patients are likely to register and be treated in different hospitals. It is crucial to develop secure sharable medical file system cross diverse

Table 2: Function Comparison

Scheme	**Attribute based encryption**	**Cross domain**	**Break glass access**	**Password based BGK**	**Outsource decryption**	**Verifiable decryption**	**Secure deplication**
[6]	×	×	√	×	×	×	×
[7]	√	×	√	×	×	×	×
[24]	×	×	√	×	×	×	×
[25]	×	√	√	×	×	×	×
[35]	×	×	√	√	×	×	×
[3]	×	×	×	×	×	×	√
[4]	×	×	×	×	×	×	√
[29]	×	×	×	×	×	×	√
[17]	×	×	×	×	×	×	√
[18]	×	×	×	×	×	×	√
[20]	×	×	×	×	×	×	√
[11]	√	×	×	×	×	×	√
[27]	√	×	×	×	×	×	×
[13]	√	×	×	×	√	×	×
[16]	√	×	×	×	√	√	×
[28]	√	×	×	×	√	√	×
[23]	√	×	×	×	√	√	×
[21]	√	×	×	×	×	×	×
[26]	√	×	×	×	×	×	×
[19]	√	×	×	×	×	×	×
[30]	√	×	×	×	×	×	×
[31]	√	×	×	×	×	×	×
[14]	√	√	×	×	×	×	×
[15]	√	√	×	×	×	×	×
[34]	√	√	×	×	√	×	×
[22]	√	√	×	×	×	×	×
Ours	√	√	√	√	√	√	√

medical domains. Our system enables the encrypted medical files to be accessed by the users (such as physicians, nurses or the friends of patients) from different medical domains. The schemes in [14, 15, 34, 22] can also enable cross-domain access control, but the other schemes [6, 7, 24, 25, 35, 3, 29, 17, 18, 20, 4, 11, 13, 16, 28, 23, 27, 21, 26, 19, 30, 31] cannot.

- ***Break-glass access***: In emergency situations, the encrypted medical file owners may be fainted and unable to authorize the access to any doctors when their files are urgently demanded. The break-glass access mechanism provides emergency access to patients medical files if the data owner's authorization is not available. Such access pattern is fatal in the medical application scenario to save patient's life. Unfortunately, very few research work has considered this requirement. The paper [6, 7, 24, 25] discuss this issue but give no concrete construction. The paper in [35] proposes a concrete break-glass method, but does not support attribute based access control. Our system not only enables fine-grained access control, but it also allows break-glass data access for emergency situations.

- ***Password based break-glass key***: Apart of the attribute secret key, this proposed system also enables password-based break-glass key to realize encrypted medical file access. The password based break-glass key can be deduced using a password, which is pre-set by the patient. The emergency contact person who holds this password can recover the medical files through the password based break-glass key on behalf of the patient. Our system and the scheme in [35] supports password based break-glass key, while the other schemes do not.

- ***Outsource decryption***: The outsource decryption methodology allows the public cloud to transform an ABE ciphertext into a short pattern ciphertext using a delegation key provided by the data user, such that the user can decrypt the transformed ciphertext very efficiently than the decryption on the original ciphertext. Our system and the schemes in [13, 16, 28, 23] support such decryption method, while the others do not. However, the schemes in [13, 16, 28, 23] can only support single domain outsource decryption. Our system supports outsource decryption over cross-domain shared encrypted medical files.

- ***Verifiable decryption***: Since the service provider of the public cloud may be selfish to save its computation resource and return a maliciously transformed ciphertext to the data user, it is necessary to provide a method for the user to verify the correctness of the transformation done by the public cloud, which is refereed as "verifiable decryption". This function is supported by our system and the schemes in [16, 28, 23].

- ***Secure deduplication***: In order to save storage space and communication bandwidth, the public cloud executes the secure deduplication algorithms to distinguish the set of ciphertext with the same underlying healthcare record. The requirement is that the plaintext of the medical file is not revealed to the public cloud. The schemes in [3, 29, 17, 18, 20, 4, 11] supports secure deduplication, while the scheme in [11] and our system fully demonstrate secure deduplication over attribute-based encrypted data.

Table 3 compares the storage overhead of our system and the other related schemes. The notations PP, $SK_{i,j}$ and CT in Table 3 denote the size of the public parameter, the user's attribute secret key and the ciphertext, respectively. Let $|U|$ be the size of the universe attribute set U, l the number of rows in matrix $\mathbb{A}$, and $|S|$ the size of user's attribute set S.

- The sizes of the public parameter PP in schemes [11, 13, 26, 31, 14, 15, 34, 22] and this proposed system are constant. However, $|PP|$ in the other schemes [16, 28, 23, 27, 21, 19, 30] linearly increase with the universal attribute set size $|U|$. It not only wastes a lot of storage space, but it also introduces inflexibility to the system. When a new attribute is to be added to the system, the entire system has to be reconstructed. Hence, it is not practical in a large scale secure electronic medical system.

- The sizes of the attribute secret key $SK_{i,j}$ of users in all the schemes grow with the size of the user's attribute set S. In our system, $SK_{i,j}$ comprises $|S| + 3$ elements in group $\mathbb{G}$ and the attribute secret key size is in the medium level

Table 3: Storage Overhead Comparison

Scheme	$\|\boldsymbol{PP}\|$	$\|\boldsymbol{SK_{i,j}}\|$	$\|\boldsymbol{CT}\|$
[11]	$5\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(2\|S\| + 2)\|\mathbb{G}\|$	$(3l + 1)\|\mathbb{G}\| + \|\mathbb{G}_T\|$
[27]	$(2\|U\| + 3)\|\mathbb{G}\|$	$(5\|S\|)\|\mathbb{G}\|$	$(2l + 1)\|\mathbb{G}\| + \|\mathbb{G}_T\|$
[13]	$2\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(\|S\| + 2)\|\mathbb{G}\|$	$(2l + 1)\|\mathbb{G}\| + \|\mathbb{G}_T\|$
[16]	$(\|U\| + 5)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(\|S\| + 2)\|\mathbb{G}\|$	$(4l + 3)\|\mathbb{G}\| + 2\|\mathbb{G}_T\|$
[28]	$(\|U\| + 2)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(\|S\| + 2)\|\mathbb{G}\|$	$(2l + 1)\|\mathbb{G}\| + \|\mathbb{G}_T\|$
[23]	$(\|U\| + 4)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(\|S\| + 2)\|\mathbb{G}\|$	$(2l + 1)\mathbb{G}\| + \|\mathbb{G}_T\|$
[21]	$(\|U\| + 3)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(\|S\| + 3)\|\mathbb{G}\| + \|\mathbb{Z}_p\|$	$(2l + 2)\|\mathbb{G}\| + \|\mathbb{G}_T\|$
[26]	$6\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(2\|S\| + 3)\|\mathbb{G}\| + \|\mathbb{Z}_p\|$	$(3l + 2)\|\mathbb{G}\| + \|\mathbb{G}_T\|$
[19]	$(2\|U\| + 10)\|\mathbb{G}\| + 3\|\mathbb{G}_T\|$	$(3\|S\|)\|\mathbb{G}\|$	$(l + 3)\|\mathbb{G}\| + 2\|\mathbb{G}_T\|$
[30]	$(3\|U\| + 1)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(2\|S\| + 1)\|\mathbb{G}\| + \|\mathbb{Z}_p\|$	$(l + 2)\|\mathbb{G}\|$
[31]	$\|\mathbb{G}\|$	$(3\|S\|)\|\mathbb{G}\|$	$(5l)\|\mathbb{G}\| + \|\mathbb{G}_T\|$
[14]	$3\mathbb{G}\|$	$(\|S\| + 1)\|\mathbb{G}\|$	$(5l)\|\mathbb{G}\| + \|\mathbb{G}_T\|$
[15]	$3\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(\|S\| + 6)\|\mathbb{G}\|$	$(5l)\|\mathbb{G}\| + \|\mathbb{G}_T\|$
[34]	$\|\mathbb{G}\|$	$\|S\| \cdot \|\mathbb{G}\|$	$(3l)\|\mathbb{G}\| + \|\mathbb{G}_T\|$
[22]	$5\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(4\|S\| + 1)\|\mathbb{G}\|$	$(5l)\|\mathbb{G}\| + \|\mathbb{G}_T\|$
Ours	$4\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(\|S\| + 3)\|\mathbb{G}\|$	$(4l + 1)\|\mathbb{G}\| + \|\mathbb{G}_T\|$

among these compared schemes. It is less than the schemes in [11, 27, 21, 26, 19, 30, 31, 22], and larger than the schemes in [13, 16, 28, 23, 14, 34].

- The sizes of the ciphertext in all the schemes linearly increase with the number of rows of the matrix $\mathbb{A}$, which is utilized to define the access policy. In our system, the ciphertext CT comprises $4l + 1$ elements in group $\mathbb{G}$ and one element in group $\mathbb{G}_T$, which is in the medium level among these compared schemes. It is less than the schemes in [16, 31, 14, 15, 22], and larger than the schemes in [11, 13, 28, 23, 27, 21, 26, 19, 30, 34].

Table 4 compares the computation overhead of our system and the other related schemes. The notations $KeyGen.User$, Enc and Dec_1 in Table 3 denote the user's attribute secret key generation algorithm, encryption algorithm and the decryption algorithm using the attribute key respectively. The notations t_p, t_{e_1} and t_{e_2} denote the computation time of the bilinear pairing computation, exponentiation calculation in group $\mathbb{G}$ and in group $\mathbb{G}_T$ respectively.

- In $KeyGen.User$ algorithm, the medical institute MI_i needs $3|S|+3$ exponentiation calculations in group $\mathbb{G}$ to generate user's attribute key, where S denotes the attribute set of the user. The calculation overhead of the user's attribute key generation algorithm of our system is smaller than that in [11, 27, 26, 19, 31, 22] and larger than that in [13, 16, 28, 23, 21, 30, 14, 34].

- In encryption Enc algorithm, the data owner needs $6l + 1$ exponentiation calculations in group $\mathbb{G}$ and one exponentiation calculations in group $\mathbb{G}_T$ to generate the encrypted medical file, where l denotes the number of rows in matrix $\mathbb{A}$. The calculation overhead of the ciphertext generation algorithm of our system is smaller than that in [16, 31, 15] and larger than the other schemes.

- Due to the usage of oursouce decryption mechanism, our system consumes only one exponentiation calculation in group $\mathbb{G}$ to recover the plaintext using user's attribute secret key. The schemes in [13, 16, 28, 23] also have such low computation cost. The scheme in [30] does not include decryption algorithm. The other schemes suffer from high computation overhead in decryption algorithm. For

instance, the scheme in [15] requires as high as $6|S|$ bilinear pairing calculations and $2|S|$ exponentiation calculations in group $\mathbb{G}_T$ to encrypt a message.

Table 4: Computation Overhead Comparison

Scheme	$\boldsymbol{KeyGen.User}$	$\boldsymbol{Enc}$	$\boldsymbol{Dec_1}$
[11]	$(4\|S\|+3)t_{e_1}$	$(5l+4)t_{e_1}+t_{e_2}$	$(3\|S\|+1)t_p$
[27]	$(6\|S\|)t_{e_1}$	$t_p+t_{e_2}+(2l+1)t_{e_1}$	$(3\|S\|)t_p+\|S\|t_{e_1}+2\|S\|t_{e_2}$
[13]	$(\|S\|+2)t_{e_1}$	$t_p+(3l+1)t_{e_1}+t_{e_2}$	t_{e_1}
[16]	$(\|S\|+3)t_{e_1}$	$2t_p+(6l+4)t_{e_1}+2t_{e_2}$	t_{e_1}
[28]	$(\|S\|+3)t_{e_1}$	$t_p+(3l+1)t_{e_1}+t_{e_2}$	t_{e_1}
[23]	$(\|S\|+3)t_{e_1}$	$t_p+(3l+3)t_{e_1}+t_{e_2}$	t_{e_1}
[21]	$(\|S\|+4)t_{e_1}$	$t_p+t_{e_2}+$ $(3l+2)t_{e_1}$	$(2\|S\|+1)t_p+\|S\|t_{e_2}$ $+(\|S\|+1)t_{e_1}$
[26]	$(4\|S\|+4)t_{e_1}$	$t_p+t_{e_2}+$ $(5l+2)t_{e_1}$	$(3\|S\|+1)t_p+\|S\|t_{e_2}$ $+(\|S\|+1)t_{e_1}$
[19]	$4\|S\|t_{e_1}$	$2t_p+2t_{e_2}+$ $(l+5)t_{e_1}$	$4t_p+t_{e_2}+$ $(3l+4)t_{e_1}$
[30]	$(2\|S\|+2)t_{e_1}$	$(2\|S\|+2)t_{e_1}$	$\perp$
[31]	$(4\|S\|)t_{e_1}$	$(7l)\cdot t_{e_1}$	$(2\|S\|)t_p+(6\|S\|)t_{e_1}$
[14]	$(\|S\|+3)t_{e_1}$	$l\cdot t_p+(2l+1)t_{e_1}$	$(2\|S\|+1)t_p$ $+\|S\|t_{e_1}+\|S\|t_{e_2}$
[15]	$(\|S\|+9)t_{e_1}$	$l\cdot t_p+(6l)t_{e_1}+l\cdot t_{e_2}$	$(6\|S\|)t_p+(2\|S\|)t_{e_2}$
[34]	$(2\|S\|)t_{e_1}$	$t_p+(5l)t_{e_1}+t_{e_2}$	$\|S\|\cdot t_{e_1}$
[22]	$(5\|S\|+1)t_{e_1}$	$t_p+(5l)t_{e_1}+t_{e_2}$	$(4\|S\|+1)t_p+t_{e_2}$
Ours	$(3\|S\|+3)t_{e_1}$	$(6l+1)t_{e_1}+t_{e_2}$	t_{e_1}

In Table 5, the performance of each algorithm in our system is presented, which includes both storage and computation overheads.

1. In $GlobalSetup$ algorithm, the public parameter PP consists of four elements in group $\mathbb{G}$ and one element in group $\mathbb{G}_T$, and the master secret key MSK consists

Table 5: Performance of Our System

Algorithm	**Parameter**	**Storage Overhead**	**Computation Overhead**
$GlobalSetup$	PP	$4\|\mathbb{G}\| + \|\mathbb{G}_T\|$	t_p
	MSK	$\|\mathbb{Z}_p\|$	0
$KeyGen.MI$	PK_i	$2\|\mathbb{G}\|$	$2t_{e_1}$
	SK_i	$4\|\mathbb{G}\| + \|\mathbb{Z}_p\|$	$5t_{e_1}$
$KeyGen.User$	$SK_{i,j}$	$(\|S\| + 3)\|\mathbb{G}\|$	$(3\|S\| + 3)t_{e_1}$
$KeyGen.Del$	$DK_{i,j}$	$(\|S\| + 4)\|\mathbb{G}\|$	$(\|S\| + 4)t_{e_1}$
$KeyGen.BGK$	$BGK_{i,j,1}$	$3\|\mathbb{G}\| + \|\mathbb{Z}_p\|$	$5t_{e_1}$
	$BGK_{i,j,2}$	$3\|\mathbb{G}\| + \|\mathbb{Z}_p\|$	
$Extract.BGK$	–		$13t_{e_1}$
Enc	CT	$(4l + 1)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(6l + 1)t_{e_1} + t_{e_2}$
	pf	$3\|\mathbb{G}\| + 3\|\mathbb{Z}_p\|$	t_{e_1}
	$TK_{i,j}$	$\|\mathbb{G}\|$	$7t_{e_1}$
$ValidityTest$	–		$5t_{e_1}$
$MsgTest$	–		$2t_p$
$ReEnc$	CT'	$(4l + 1)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(7l + 1)t_{e_1} + t_{e_2}$
$PartialDec$	CT_p	$2\|\mathbb{G}_T\|$	$(4\|S\| + 1)t_p + \|S\| \cdot t_{e_2}$
Dec_1	–		t_{e_1}
Dec_2	–		$0t_p + 0t_{e_1} + 0t_{e_2}$

of one element in $\mathbb{Z}_p$. The computation overhead of PP is one bilinear pairing operation.

2. In $KeyGen.MI$ algorithm, the medical institute MI_i's public key PK_i consists of two elements in group $\mathbb{G}$, and MI_i's secret key SK_i consists of four elements in group $\mathbb{G}$ and one element in $\mathbb{Z}_p$. The computation overhead of PK_i is two exponentiation calculations in group $\mathbb{G}$ and that of SK_i is five exponentiation calculations in group $\mathbb{G}$.
3. In $KeyGen.User$ algorithm, the user's attribute secret key $SK_{i,j}$ consists of $|S|+3$ elements in group $\mathbb{G}$ with computation overhead $3|S|+3$ exponentiation calculations in group $\mathbb{G}$.
4. In $KeyGen.Del$ algorithm, the delegation key $DK_{i,j}$ consists of $|S|+4$ elements in group $\mathbb{G}$ with computation overhead $|S|+4$ exponentiation calculations in group $\mathbb{G}$.
5. In $KeyGen.BGK$ algorithm, the auxiliary break-glass key messages $BGK_{i,j,1}$ and $BGK_{i,j,2}$ all have three elements in group $\mathbb{G}$ and one element in group $\mathbb{G}_T$. The total computation overhead is five exponentiation calculations in group $\mathbb{G}$.
6. In $Extract.BGK$ algorithm, the computation overhead has five exponentiation calculations in group $\mathbb{G}$. No storage overhead is required in this algorithm.
7. In Enc algorithm, the storage overhead of the ciphertext CT, the proof message pf and the transformation key $TK_{i,j}$ are $(4l+1)|\mathbb{G}|+|\mathbb{G}_T|$, $3|\mathbb{G}|+3|\mathbb{Z}_p|$ and $|\mathbb{G}|$, respectively. The computation overhead of them are $(6l+1)t_{e_1}+t_{e_2}$, t_{e_1} and $7t_{e_1}$, respectively.
8. In $ValidityTest$ algorithm, the computation overhead is five exponentiation calculations in group $\mathbb{G}$. No storage overhead is required in this algorithm.
9. In $MsgText$ algorithm, the computation overhead is two bilinear pairing operations and no storage overhead is required in this algorithm.
10. In $ReEnc$ algorithm, the re-encrypted ciphertext CT' consists of $4l+1$ elements in group $\mathbb{G}$ and one element in group $\mathbb{G}_T$, with computation overhead $(7l+1)t_{e_1}+t_{e_2}$.
11. In $Partial$ algorithm, the partial ciphertext CT_p consists of two elements in group $\mathbb{G}_T$, with computation overhead $(4|S|+1)t_p+|S|\cdot t_{e_2}$.

12. In Dec_1 algorithm, the computation overhead is one exponentiation calculations in group $\mathbb{G}$. No storage overhead is required in this algorithm.
13. In Dec_2 algorithm, no bilinear pairing or exponentiation calculation is required. No storage overhead is required in this algorithm.

7.2. Experimental Analysis

The proposed secure storage system is implemented using the Stanford Pairing Based Cryptography (PBC) library [5], which is a free C library to operate the mathematical calculations underlying the pairing based crypto systems. We use PBC-0.4.7 and Visual Studio 2012 in our implementation. The simulations are conducted on a notebook computer with Intel Core^{TM} i3-2120 CPU @ 3.3 GHz, 4.00 GB RAM running 64-bit Windows 7 operation system.

The proposed cross-domain medical data cloud storage system, together with break-glass access control and secure deduplication, can be simulated over the type A elliptic curve with the expression $E : y^2 = x^3 + x$. The group parameters are $|\mathbb{G}| = 1024$ bits, $|\mathbb{G}_T| = 1024$ bits and $|\mathbb{Z}_p| = 160$ bits. Except for our system, we also implement the schemes in [11, 31, 15, 22] to make a comparison.

Table 6: Simulation of Public Parameter Size (KB)

Ours	**[11]**	**[31]**	**[15]**	**[22]**
0.640	0.768	0.128	0.512	0.768

Tables 6-8 and Figures 8-9 show the storage overhead of our system and the schemes in [11, 31, 15, 22], which includes the public parameter size, user's attribute secret key size and ciphertext size in KB.

- The public parameter sizes $|PP|$ of these schemes are constant, which are shown in Table 6. In our system, the public parameter is 0.64 KB, which is smaller than that in [11, 22] and larger than that in [31, 15].
- Table 7 and Figure 8 show the simulation result of the attribute secret key size. It is easy to find that our system has the best performance among these schemes.

Table 7: Simulation of Attribute Secret key Size (KB)

$\|S\|$	**Ours**	**[11]**	**[31]**	**[15]**	**[22]**
1	0.512	0.512	0.384	0.896	0.640
10	1.664	2.816	3.840	2.048	5.248
20	2.944	5.376	7.680	3.328	10.368
30	4.224	7.936	11.520	4.608	15.488
40	5.504	10.496	15.360	5.888	20.608
50	6.784	13.056	19.200	7.168	25.728
60	8.064	15.616	23.040	8.448	30.848
70	9.344	18.176	26.880	9.728	35.968
80	10.624	20.736	30.720	11.008	41.088
90	11.904	23.296	34.560	12.288	46.208
100	13.184	25.856	38.400	13.568	51.328

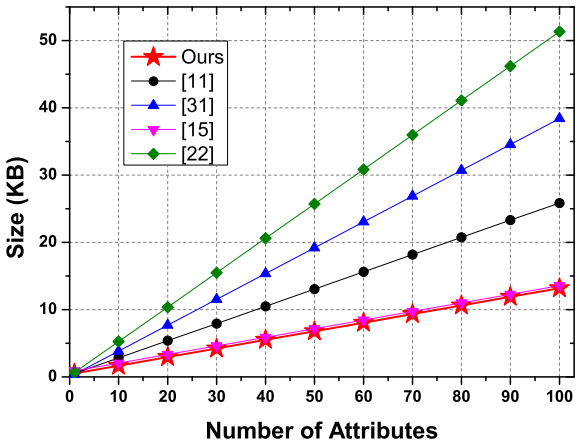


Figure 8: Simulation of Attribute Secret Key Size (KB)

Table 8: Simulation of Ciphertext Size (KB)

l	**Ours**	**[11]**	**[31]**	**[15]**	**[22]**
1	0.768	0.64	0.896	0.896	0.896
10	5.376	4.096	6.656	6.656	6.656
20	10.496	7.936	13.056	13.056	13.056
30	15.616	11.776	19.456	19.456	19.456
40	20.736	15.616	25.856	25.856	25.856
50	25.856	19.456	32.256	32.256	32.256
60	30.976	23.296	38.656	38.656	38.656
70	36.096	27.136	45.056	45.056	45.056
80	41.216	30.976	51.456	51.456	51.456
90	46.336	34.816	57.856	57.856	57.856
100	51.456	38.656	64.256	64.256	64.256

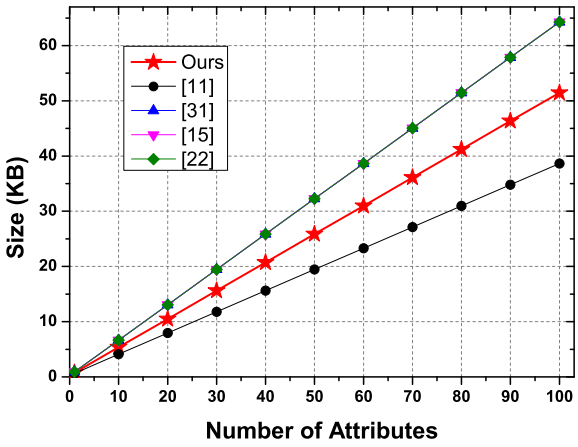


Figure 9: Simulation of Ciphertext Size (KB)

The scheme in [22] has the worst performance, and the schemes in [11, 31, 15] are at the medium level. When the size of the attribute set is 10, the $|SK_{i,j}| = 1.664$ KB in our system and $|SK_{i,j}| = 5.248$ KB in scheme [22]. When the size of the attribute set is 100, the $|SK_{i,j}| = 13.184$ KB in our system, and $|SK_{i,j}| = 51.328$ KB in scheme [22], which is three times of that in our system.

- Table 8 and Figure 9 show the simulation result of the ciphertext size. It seems that there are only three lines for five schemes. In fact, the schemes in [31, 15, 22] have the same performance and the lines representing these schemes overlap. The performance of the ciphertext size of our system is at the medium level. When the row number l of matrix $\mathbb{A}$ is 10, the ciphertext size $|CT| = 5.376$ KB in our system, $|CT| = 4.096$ KB in scheme [11], and $|CT| = 6.646$ KB in schemes [31, 15, 22]. When $l = 100$, the ciphertext size $|CT| = 51.456$ KB in our system, $|CT| = 38.656$ KB in scheme [11], and $|CT| = 64.256$ KB in schemes [31, 15, 22].

Tables 9-11 and Figures 10-12 show the calculation cost of our system and the schemes in [11, 31, 15, 22], which include the key generation time for user's attribute secret key, encryption time and decryption time using attribute secret key.

- Table 9 and Figure 10 show the execution time of the $KeyGen.User$ algorithm. The performance of $KeyGen.User$ algorithm of our system is better than that in schemes [11, 31, 22] but worse than that in scheme [15]. The lines of the schemes [11, 31] are quite close. When the size of the attribute set is 10, the $KeyGen.User$ algorithm's execution time has 0.394 s, 0.397 s, 0.174 s, 0.468 s and 0.354 s in schemes [11, 31, 15, 22] and our system, respectively. When the size of the attribute set is 100, the $KeyGen.User$ algorithm's execution time has 3.695 s, 3.668 s, 0.989 s, 4.594 s and 2.708 s in schemes [11, 31, 15, 22] and our system, respectively.

- Table 10 and Figure 11 show the execution time of the Enc algorithm. The performance of Enc algorithm of our system is better than that in schemes [31, 15] but worse than that in scheme [11, 22]. The lines of the schemes [11, 22]

are quite close. When the row number l of matrix $\mathbb{A}$ is 10, the Enc algorithm's execution time has 0.498 s, 0.642 s, 0.756 s, 0.479 s and 0.562 s in schemes [11, 31, 15, 22] and our system, respectively. When $l = 100$, the Enc algorithm execution times are 4.624 s, 6.418 s, 7.563 s, 4.505 s and 5.513 s in schemes [11, 31, 15, 22] and our system, respectively.

- Table 11 and Figure 12 show the execution time of the Dec_1 algorithm. The performance of Dec_1 algorithm of our system is much better than the other schemes and the execution time remains constant with approximately 0.009 s. When the size of the attribute set is 10, the Dec_1 algorithm's execution times has 0.559 s, 0.911 s, 1.353 s and 0.742 s in schemes [11, 31, 15, 22], respectively. When the size of the attribute set is 100, the Dec_1 algorithm's execution time has 5.426 s, 9.107 s, 11.333 s and 7.231 s respectively in schemes [11, 31, 15, 22].

Table 9: Execution time of $KeyGen.User$ algorithm (s)

$\|S\|$	**Ours**	**[11]**	**[31]**	**[15]**	**[22]**
1	0.055	0.064	0.037	0.092	0.055
10	0.354	0.394	0.397	0.174	0.468
20	0.528	0.791	0.694	0.286	0.986
30	0.853	1.108	1.100	0.338	1.325
40	1.178	1.555	1.467	0.449	1.843
50	1.373	1.801	1.834	0.581	2.371
60	1.678	2.228	2.161	0.603	2.703
70	1.993	2.595	2.567	0.724	3.218
80	2.188	2.902	2.934	0.816	3.727
90	2.503	3.388	3.301	0.938	4.085
100	2.708	3.695	3.668	0.989	4.594

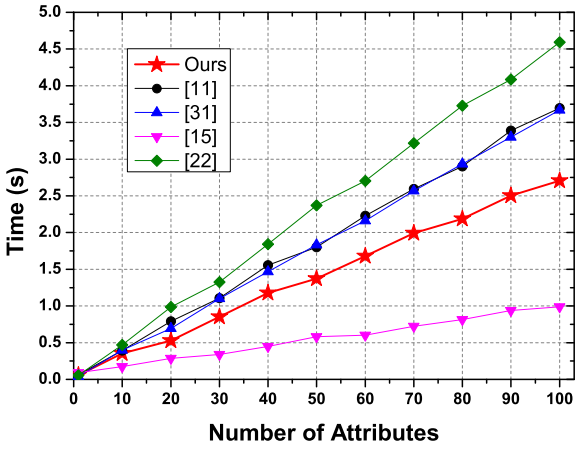


Figure 10: Execution time of $KeyGen.User$ algorithm (s)

Table 10: Execution time of Enc algorithm (s)

$\|S\|$	**Ours**	**[11]**	**[31]**	**[15]**	**[22]**
1	0.067	0.085	0.064	0.076	0.066
10	0.562	0.498	0.642	0.756	0.479
20	1.192	0.956	1.484	1.713	0.838
30	1.662	1.415	1.925	2.269	1.396
40	2.212	1.973	2.767	3.225	1.754
50	2.629	2.332	3.209	3.781	2.313
60	3.313	2.790	3.851	4.338	2.671
70	3.993	3.348	4.693	5.294	3.230
80	4.413	3.707	5.135	6.252	3.588
90	4.803	4.265	5.576	6.706	4.147
100	5.513	4.624	6.418	7.563	4.505

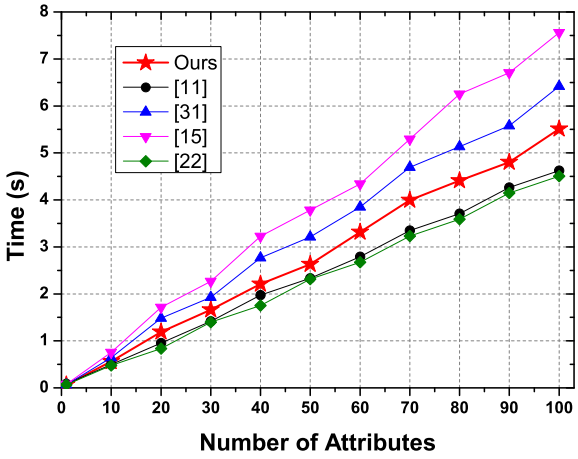


Figure 11: Execution time of Enc algorithm (s)

Table 11: Execution time of Dec_1 algorithm (s)

$\|S\|$	**Ours**	**[11]**	**[31]**	**[15]**	**[22]**
1	0.009	0.072	0.091	0.113	0.093
10	0.009	0.559	0.911	1.353	0.742
20	0.009	1.215	1.821	2.267	1.463
30	0.008	1.582	2.952	3.751	2.314
40	0.009	2.181	3.643	4.533	2.905
50	0.009	2.862	4.233	5.357	3.786
60	0.009	3.263	5.464	6.801	4.217
70	0.008	3.724	6.635	8.233	5.068
80	0.009	4.345	7.285	9.066	5.959
90	0.009	4.701	8.006	10.002	6.386
100	0.009	5.426	9.107	11.333	7.231

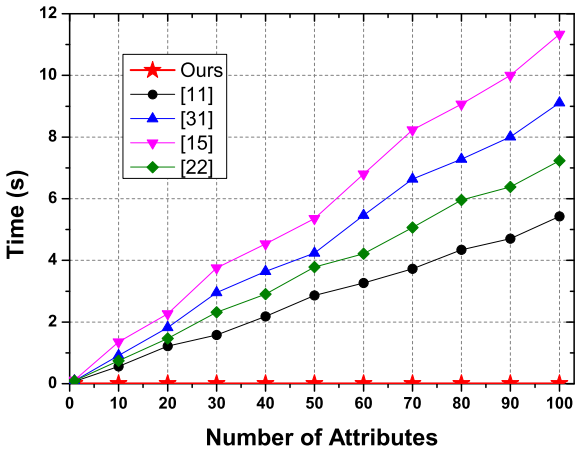


Figure 12: Execution time of Dec_1 algorithm (s)

8. Conclusion

We designed an IoT-based cross-domain medical file cloud storage system with break-glass access control and secure deduplication to facilitate the data management in electronic healthcare system. The secure deduplication demonstrated by our paper can enable (1) the public cloud to distinguish the ciphertexts with the same medical message; and (2) the private cloud of the medical institute to re-encrypt the ciphertext with a combined access policy , so that all the original authorized data users are capable to access to the new ciphertext. Moreover, this system is the first one to provide two suits of access control mechanisms for the medical applications: cross-domain attribute based access in normal situations and password-based break-glass access in emergency situations. These two mechanisms ensure not only the security of patients' encrypted medical files, but they also provide immediate emergency data access to save patients' lives. Compared with other related projects, this system is validated to be secure. The simulation results indicate that this system is efficient and practical.

Acknowledgement

The authors thank the associate editor and reviewers for their constructive and generous feedback. This work is supported by National Natural Science Foundation of China (No. 61402112, 61672159, 61702105, 61472307, 61472309, 61502086); Science and Technology Project of Fujian Education Department (No. JA12028); Open Fund Project of Fujian Provincial Key Laboratory of Information Processing and Intelligent Control (Minjiang University) (No. MJUKF201734); Fujian Major Project of Regional Industry (No. 2014H4015); Major Science and Technology Project of Fujian Province (No. 2015H6013); Technology Innovation Platform Project of Fujian Province (No. 2014H2005), Fujian Collaborative Innovation Center for Big Data Application in Governments, Fujian Engineering Research Center of Big Data Analysis and Processing.

[1] Amandeep Singh Sohal, Rajinder Sandhu, Sandeep K.Sood, and VictorChang. "A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments." Computers & Security (2017). Publish online. DOI: 10.1016/j.cose.2017.08.016.

[2] Beimel, Amos, et al. Secure schemes for secret sharing and key distribution. Technion-Israel Institute of technology, Faculty of computer science, 1996.

[3] Bellare, Mihir, Sriram Keelveedhi, and Thomas Ristenpart. "Message-locked encryption and secure deduplication." Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2013.

[4] Bellare, Mihir, and Sriram Keelveedhi. "Interactive message-locked encryption and secure deduplication." IACR International Workshop on Public Key Cryptography. Springer Berlin Heidelberg, 2015.

[5] Boneh Lynn. The Stanford Pairing Based Crypto Library. [Online]. Available: http://crypto.stanford.edu/pbc, accessed May 7, 2014.

[6] Brucker, Achim D., and Helmut Petritsch. ”Extending access control models with break-glass.” Proceedings of the 14th ACM symposium on Access control models and technologies. ACM, 2009.

[7] Brucker, Achim D., Helmut Petritsch, and Stefan G. Weber. ”Attribute-based encryption with break-glass.” IFIP International Workshop on Information Security Theory and Practices. Springer Berlin Heidelberg, 2010.

[8] Chang, Victor. ”Towards a Big Data system disaster recovery in a Private Cloud.” Ad Hoc Networks 35 (2015): 65-82.

[9] Chang, Victor, Yen-Hung Kuo, and Muthu Ramachandran. ”Cloud computing adoption framework: A security framework for business clouds.” Future Generation Computer Systems 57 (2016): 24-41.

[10] Chang, Victor, and Gary Wills. ”A model to compare cloud and non-cloud storage of Big Data.” Future Generation Computer Systems 57 (2016): 56-76.

[11] Cui, Hui, Robert H. Deng, Yingjiu Li, and Guowei Wu. ”Attribute-Based Storage Supporting Secure Deduplication of Encrypted Data in Cloud.” IEEE Transactions on Big Data (2017).

[12] Gope, Prosanta, and Tzonelih Hwang. ”BSN-Care: a secure IoT-based modern healthcare system using body sensor network.” IEEE Sensors Journal 16.5 (2016): 1368-1376.

[13] Green, Matthew, Susan Hohenberger, and Brent Waters. ”Outsourcing the decryption of abe ciphertexts.” USENIX Security Symposium. Vol. 2011. No. 3. 2011.

[14] Han, Jinguang, et al. ”Privacy-preserving decentralized key-policy attribute-based encryption.” IEEE Transactions on Parallel and Distributed Systems 23.11 (2012): 2150-2162.

[15] Han, Jinguang, et al. ”Improving privacy and security in decentralized ciphertext-policy attribute-based encryption.” IEEE Transactions on Information Forensics and Security 10.3 (2015): 665-678.

[16] Lai, Junzuo, et al. ”Attribute-based encryption with verifiable outsourced decryption.” IEEE Transactions on Information Forensics and Security 8.8 (2013): 1343-1354.

[17] Li, Jin, et al. ”Secure deduplication with efficient and reliable convergent key management.” IEEE transactions on parallel and distributed systems 25.6 (2014): 1615-1625.

[18] Li, Jin, et al. ”A hybrid cloud approach for secure authorized deduplication.” IEEE Transactions on Parallel and Distributed Systems 26.5 (2015): 1206-1216.

[19] Liang, Kaitai, and Willy Susilo. ”Searchable attribute-based mechanism with efficient data sharing for secure cloud storage.” IEEE Transactions on Information Forensics and Security 10.9 (2015): 1981-1992.

[20] Liu, Jian, N. Asokan, and Benny Pinkas. ”Secure deduplication of encrypted data without additional independent servers.” Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, 2015.

[21] Liu, Zhen, Zhenfu Cao, and Duncan S. Wong. ”White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures.” IEEE Transactions on Information Forensics and Security 8.1 (2013): 76-88.

[22] Luo, Entao, Liu Qin, and Wang Guojun . ”Hierarchical Multi-Authority and Attribute-Based Encryption Friend Discovery Scheme in Mobile Social Networks.” IEEE Communications Letters 20.9 (2016): 1772-1775.

[23] Mao, Xianping, et al. ”Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption.” IEEE Transactions on Dependable and Secure Computing 13.5 (2016): 533-546.

[24] Marinovic, Srdjan, et al. ”Rumpole: a flexible break-glass access control model.” Proceedings of the 16th ACM symposium on Access control models and technologies. ACM, 2011.

[25] Maw, Htoo Aung, et al. ”BTG-AC: Break-the-Glass Access Control Model for Medical Data in Wireless Sensor Networks.” IEEE journal of biomedical and health informatics 20.3 (2016): 763-774.

[26] Ning, Jianting, et al. ”White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes.” IEEE Transactions on Information Forensics and Security 10.6 (2015): 1274-1288.

[27] Ostrovsky, Rafail, Amit Sahai, and Brent Waters. ”Attribute-based encryption with non-monotonic access structures.” Proceedings of the 14th ACM conference on Computer and communications security. ACM, 2007.

[28] Qin, Baodong, et al. ”Attribute-based encryption with efficient verifiable outsourced decryption.” IEEE Transactions on Information Forensics and Security 10.7 (2015): 1384-1393.

[29] Stanek, Jan, et al. ”A secure data deduplication scheme for cloud storage.” International Conference on Financial Cryptography and Data Security. Springer Berlin Heidelberg, 2014.

[30] Sun, Wenhai, et al. ”Protecting your right: verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud.” IEEE Transactions on Parallel and Distributed Systems 27.4 (2016): 1187-1198.

[31] Yang, Kan, et al. ”Time-domain attribute-based access control for cloud-based video content sharing: A cryptographic approach.” IEEE Transactions on Multimedia 18.5 (2016): 940-950.

[32] Yang, Yang, and Maode Ma. ”Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds.” IEEE Transactions on Information Forensics and Security 11.4 (2016): 746-759.

[33] Yang, Yang, Xianghan Zheng, and Chunming Tang. ”Lightweight distributed secure data management system for health internet of things.” Journal of Network and Computer Applications 89 (2017): 26-37.

[34] Zhang, Kai, et al. "Adaptively secure multi-authority attribute-based encryption with verifiable outsourced decryption." Science China Information Sciences 59.9 (2016): 99105.

[35] Zhang, Tao, Sherman SM Chow, and Jinyuan Sun. "Password-Controlled Encryption with Accountable Break-Glass Access." Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. ACM, 2016.