9-2020

# Efficient fine-grained data sharing mechanism for electronic medical record systems with mobile devices

Hui MA

Rui ZHANG

Guomin YANG
*Singapore Management University*, gmyang@smu.edu.sg

Zishuai ZONG

Kai HE

*See next page for additional authors*



## Citation

MA, Hui; ZHANG, Rui; YANG, Guomin; ZONG, Zishuai; HE, Kai; and XIAO, Yuting. Efficient fine-grained data sharing mechanism for electronic medical record systems with mobile devices. (2020). *IEEE Transactions on Dependable and Secure Computing*. 17, (5), 1026-1038.
**Available at:** https://ink.library.smu.edu.sg/sis_research/7294

# Efficient Fine-Grained Data Sharing Mechanism for Electronic Medical Record Systems with Mobile Devices

Hui Ma, Rui Zhang, Guomin Yang, *Member, IEEE*, Zishuai Song, Kai He, and Yuting Xiao

**Abstract**—Sharing digital medical records on public cloud storage via mobile devices facilitates patients (doctors) to get (offer) medical treatment of high quality and efficiency. However, challenges such as data privacy protection, flexible data sharing, efficient authority delegation, computation efficiency optimization, are remaining toward achieving practical fine-grained access control in the Electronic Medical Record (EMR) system. In this work, we propose an innovative access control model and a fine-grained data sharing mechanism for EMR, which simultaneously achieves the above-mentioned features and is suitable for resource-constrained mobile devices. In the model, complex computation is outsourced to public cloud servers, leaving almost no complex computation for the private key generator (PKG), sender and receiver. Additionally, the communication cost of the PKG and users is optimized. Moreover, we develop an extensible library called libabe that is compatible with Android devices, and the access control mechanism is actually deployed on realistic environment, including public cloud servers, a laptop and an inexpensive mobile phone with constrained resources. The experimental results indicate that the mechanism is efficient, practical and economical.

**Index Terms**—Data sharing mechanism, attribute based encryption, secure outsourced computation, cloud computing, electronic medical record

## 1 Introduction

As an attractive paradigm for digital information processing, Electronic Medical Record (EMR) benefits patients to obtain medical treatment of high quality and efficiency and enables doctors and patients to conveniently share medical records, e.g., medical history, medication and allergies, radiology images and personal health information etc. To reduce the cost of maintaining specialized data center and achieve data sharing, the EMR systems can outsource medical records to public cloud, where doctors and patients can store, manage and share medical records.

To achieve data sharing, various server-based access control mechanisms have been proposed, e.g., Role-Based Access Control (RBAC) [1], Temporal-RBAC [2], [3] and GEO-RBAC [4], where a trusted access control server is employed to act as a supervisor. However, traditional access control mechanisms may be not suitable for cloud-assisted EMR systems where the records are stored on public cloud, because the cloud and users are not in the same trust domain. As a well-known accident, personal electronic medical information on 26.5 million military veterans, including social security numbers was artificially exposed.[1] Hence, the medical records must be encrypted to protect privacy before outsourcing to the cloud. As a secure access control method that can protect data privacy, the key assignment scheme (KAS) [5], [6], [7], [8], [9], [10] is a symmetric cryptographic primitive that can be used to enforce mandatory access control. In KAS, every security class is assigned a key that can be used to compute the key assigned to any class lower down in the hierarchy. Since security classes in KAS should be set in advance, the access policy should follow the fixed classes and is not flexible enough.

As an innovative cryptographic solution, attribute based encryption (ABE) [11] integrates flexible access control with encryption functionality. The flexibility denotes that every single file can be encrypted with a flexible access policy. Fig. 1 presents the traditional access control model of ABE. In particular, ciphertext-policy ABE (CP-ABE) [12] that is conceptually closer to RBAC has potential to be applied in ERM systems. In order to meet the immediacy and mobility, doctors use mobile devices to create, read and update medical records anywhere, at any time. A doctor is assigned with several attributes based on his/her role, such as "Surgery", "Director", "Male" etc., and uses a mobile device to encrypt patients' medical records associated with access policy, e.g., "Pediatrics" $\wedge$ ("Doctor" $\vee$ "Head-nurse"), before uploading to the EMR cloud. Other


- *H. Ma, Z. Song, K. He, Y. Xiao, are with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China. E-mail: {mahui, songzishuai, hekai, xiaoyuting}@iie.ac.cn.*
- *R. Zhang is with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China, and also with School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: r-zhang@iie.ac.cn.*
- *G. Yang is with the Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia. E-mail: gyang@uow.edu.au.*


*Manuscript received 21 Oct. 2017; revised 7 May 2018; accepted 29 May 2018. Date of publication 7 June 2018; date of current version 1 Sept. 2020. (Corresponding author: Rui Zhang.)*
*Digital Object Identifier no. 10.1109/TDSC.2018.2844814*

1. http://articles.latimes.com/2006/jun/26/health/he-privacy26

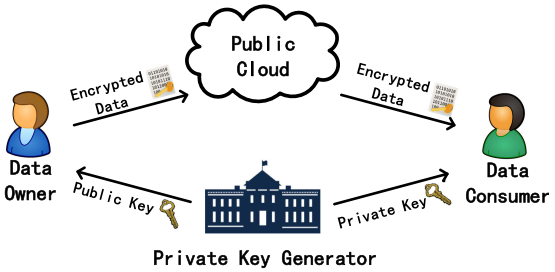



Fig. 1. Traditional access control model of ABE.

doctors who have private keys containing attributes can decrypt encrypted medical records if the attributes satisfy the access policy.

There exist a series of works on ABE for EMR, e.g., multi-authority ABE [13], traceable and revocable multi-authority ABE [14], multi-authority ABE with semi-outsourced decryption [15] etc., which achieve many practical features. However, several serious challenges still remain when ABE is deployed in a large-scale EMR system.

- *Computation Efficiency.* A major drawback that impedes ABE from wide-range deployment is the heavy computation cost, largely the paring and exponentiation operations, which often grow with the complexity of access policy. This is a huge burden for the private key generator (PKG) and users, especially for the resource-constrained mobile devices in EMR. Therefore, the first challenge is *how to simultaneously reduce the computation cost for the PKG and users?*
- *Authority Delegation.* For some emergent temporary consultations, the urgent patients' records need to be rapidly delegated to unauthorized doctors. Therefore, the second challenge is *how to achieve efficient authority delegation without revealing the data privacy in ABE?*
- *Economic Practicality.* For a large healthcare corporation with several thousands of staffs, it is not economic to buy a high-end mobile device (with powerful computation capability) for every staff. Then, the third challenge is *how to design an economical ABE system that can be suitable for resource-constrained mobile devices?*

### 1.1 Our Contribution

Aiming at solving the above challenges, we propose an efficient data sharing mechanism for EMR, which not only achieves data privacy, fine-grained access control and authority delegation simultaneously, but also optimizes the computation efficiency and is suitable for resource-constrained mobile devices. Our contribution is five-fold:

1) *New Access Control Model.* We propose a new access control model for ABE in Fig. 2, which additionally utilizes the powerful outsourced computation capability of public cloud. Compared with the traditional model in Fig. 1, our new model can significantly improve the computation efficiency for PKG and users by adding only one extra public cloud service provider.
2) *High Computation Efficiency.* Based on the new model, the heavy computation of all the ABE algorithms (i.e., key generation, encryption and decryption) is outsourced to public cloud servers, leaving 0 exponentiation for PKG (key generation) and data owners (encryption) and 1 exponentiation for data consumers (decryption). Furthermore, the new model can prevent public cloud servers from learning secret information.
3) *Efficient Authority Delegation.* We propose a fully outsourced ciphertext-policy attribute based proxy re-encryption (FO-CP-AB-PRE) system, which can re-encrypt an ABE ciphertext into a new ciphertext under a new access policy without revealing the plaintext. In the system, it takes almost no complex computation for the doctor to perform one authority delegation operation. Moreover, we propose the security model and give the security proof.
4) *Low Communication Cost.* Outsourced computation definitely brings extra communication cost, e.g., network latency, battery consumption. Nevertheless, the extra communication cost in our system is "imperceptible" for the PKG (users). Because the servers can do the computation once receiving public key, and the PKG (users) can download the results in the spare time (while being charged). They can later rapidly complete operations in real time without waiting for the cloud's response and draining the battery. Hence, our method is suitable for resource-constrained mobile devices.
5) *Performance Evaluation.* We give theoretical comparisons with various practical ABE schemes. Moreover, we develop an extensible library libabe that is compatible with Android devices. We implement a prototype of our mechanism within libabe on two cloud servers, a laptop and a low-end mobile phone. The results indicate high efficiency and economy of our methodology. We believe the prototype makes ABE a step closer to the actual deployment on EMR with mobile devices.

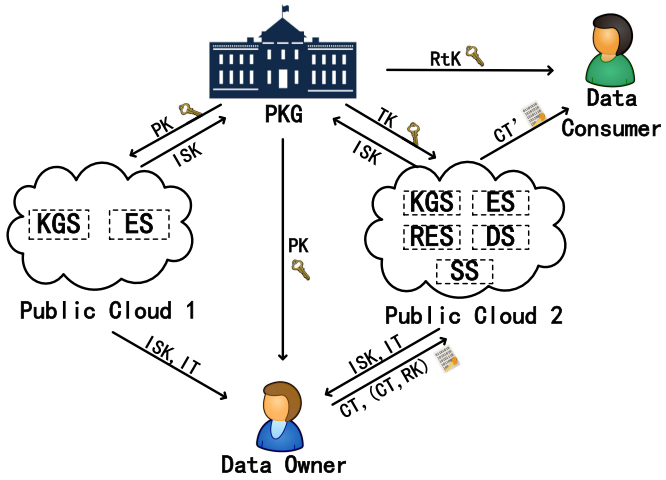



Fig. 2. Access control model of FO-CP-AB-PRE.

### 1.2 Related Work

The key assignment scheme was first considered by Akl and Taylor [5] to achieve cryptographic access control. Recently,

Alderman et al. [6] designed a space-efficient key assignment scheme based on a binary tree and proposed a tree-based cryptographic access control mechanism. Castiglione et al. [7] introduced the cryptographic hierarchical access control for dynamic structures. [16] presented that how to achieve access control in publicly verifiable outsourced computation based on KAS. [8] explored the relations between all security notions for hierarchical key assignment schemes. [9] presented a framework for the cryptographic enforcement of information flow policies. In [10], Castiglione proposed two hierarchical and shared key assignment schemes based on symmetric encryption and public key threshold broadcast encryption separately. Since the security classes in KAS should be set in advance, the access policy must follow the set security classes.

Attribute based encryption can achieve flexible access control over encrypted data. It was first introduced by Sahai and Waters [17], and Goyal et al. [11] formalized two forms of ABE: key-policy ABE (KP-ABE) [11] and ciphertext-policy ABE [12]. In KP-ABE, the secret key is associated with an access policy, while an access policy is assigned to the ciphertext in CP-ABE. A user can decrypt a ciphertext if the set of attributes satisfies the access policy.

However, a major drawback of ABE is the heavy computation cost. Green et al. [18] introduced key blinding technique to construct an ABE scheme with outsourced decryption. Li et al. [19] considered Outsourced ABE (OABE) with outsourced key-issuing and decryption. Online/offline ABE (OOABE) was first considered by Hohenberger and Waters in [20]. Zhang et al. [21] proposed fully outsourced ABE to outsource all the computation. Ma et al. [22] proposed verifiable and exculpable outsourced attribute based encryption which first achieved exculpability in ABE setting. But none of the above work considered authority delegation.

Proxy re-encryption (PRE) first introduced by Blaze et al. [23], can achieve authority delegation. Ateniese et al. [24] proposed a unidirectional PRE scheme. Hanaoka et al. [25] proposed a CCA-secure PRE scheme. Many different PRE schemes were proposed, such as anonymous proxy re-encryption [26], attribute based proxy re-encryption [27]. Recently, Shao et al. [28] proposed online/offline attribute based proxy re-encryption (OOABPRE) which aims at reducing the online computation cost on the mobile device side. The scheme in [28] is a promising and practical solution for high-end mobile devices, but it may not be economical for some scenarios, e.g., for a large healthcare corporation with several thousands of staffs, buying a high-end mobile device for every staff is not a preferred option.

## 2 Preliminary

In this section, we review some notations and definitions.

*Notations.* Let $A(a, b, \ldots) \rightarrow z$ denote the operation of running an algorithm $A$ with inputs $a, b, \ldots$ and output $z$. A function $f$ is negligible if for every $c > 0$ there exists $\lambda_0 > 0$ such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_0$.

**Definition 1 (Bilinear Groups).** *Let $\mathbb{G}, \mathbb{G}_T$ be two multiplicative cyclic groups of prime order $p$. Let $g$ be a generator of $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map with the following properties: 1) Bilinearity: for all $g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p^*$, we have $e(g^a, h^b) = e(g, h)^{ab}$. 2) Nondegeneracy: $e(g, h) \neq 1$ whenever $g, h \neq 1_{\mathbb{G}}$.*

**Definition 2 (Access Structure [29]).** *Let $\{P_1, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, \ldots, P_n\}}$ is monotone for $\forall B$ and $C$, if $B \in \mathbb{A}, B \subseteq C$, then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) of nonempty subsets of $\{P_1, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, \ldots, P_n\}} \backslash \{\emptyset\}$. The sets in $\mathbb{A}$ are called authorized sets, and the sets not in $\mathbb{A}$ are called unauthorized sets.*

ABE utilizes the access structure to achieve access control. Users are represented by some attributes according to their roles, and the access structure $\mathbb{A}$ includes a set of authorized attributes. If a user's attribute set satisfies the access structure, the user can obtain the authorized right.[2]

**Definition 3 (Linear Secret Sharing Schemes (LSSS) [29]).** *A secret sharing scheme $\Pi$ over a set of parties is called linear over $\mathbb{Z}_p$ if 1) The shares of the parties form a vector over $\mathbb{Z}_p$; 2) There exists a matrix $M$ with $l$ rows and $n$ columns called the share-generating matrix for $\Pi$. There exists a function $\rho$ which maps each row of the matrix to an associated party, i.e., for $i = 1, \ldots, l$, the value $\rho(i)$ is the party associated with row $i$. When we consider the column vector $v = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then $Mv$ is the vector of $l$ shares of the secret $s$ according to $\Pi$. The share $(Mv)_i$ belongs to party $\rho(i)$.*

The access structure is described by LSSS, that enjoys the linear reconstruction property: Suppose that $\Pi$ is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subseteq \{1, 2, \ldots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret $s$ according to $\Pi$, then $\sum_{i \in I} \omega_i \lambda_i = s$. For any unauthorized set $S \notin \mathbb{A}$, the secret $s$ should be information theoretically hidden from the parties in $S$.

## 3 System Model and Security Model

In this section, we present the system and security model.

### 3.1 System Model

Compared with the traditional model of ABE in Fig. 1, we adopt two different public cloud servers to achieve secure outsourced computation, such as outsourced key generation/encryption/re-encryption key generation/decryption. Actually, one public cloud server (e.g., public cloud 2) is sufficient for outsourced decryption, but not enough for other operations, because all the secret will be exposed to the unique cloud server. Concretely, once the server that helps data owner generate the intermediate ciphertext ($IT$), obtains the final original ciphertext ($CT$), it can easily recover the secret in $CT$. Consequently, two non-collusive cloud servers are adopted in our system, where a data owner first obtains $IT1, IT2$ from two cloud servers respectively, then generates the final $IT$ by combining $IT1$ and $IT2$. Since two public cloud servers cannot collude with each other, the final combined $IT$ should be information-theoretically hidden from two cloud servers.

The access control model consists of five entities: private key generator, public cloud 1, public cloud 2, data owners

2. We restrict our attention to monotone access structures that do not contain the "not" of an attribute.

TABLE 1
Acronyms Used in This Paper

| Acronym | Description |
|---|---|
| PKG | private key generator |
| KGS | key generation service |
| ES | encryption service |
| SS | storage service |
| RES | re-encryption service |
| DS | decryption service |
| PK | public key |
| MSK | master secret key |
| SK | user private key |
| IT | intermediate ciphertext |
| ISK | intermediate private key |
| CT | original ciphertext |
| CT′ | transformed ciphertext |
| RCT | re-encrypted ciphertext |
| RK | re-encryption key |
| TK | conversion key |
| RtK | retrieval key |

and data consumers. Table 1 lists the acronyms used in this paper. Fig. 2 shows the organization of these entities.

- PKG is responsible to set up the system parameters and distributes all the cryptographic keys $PK$, $MSK$, $SK$s, $TK$s and $RtK$s to other entities.
- Data owner defines access policies and encrypts data under these policies before uploading them to public cloud 2. He/she can also delegate (re-encrypt) the encrypted data to unauthorized data consumers. All the heavy computation in the above operations is done with the help of public cloud 1 and public cloud 2.
- Public cloud 2 equipped with $PK$ and $TK$ is deployed to provide outsourced computation service and cloud data storage service, such as KGS, ES, RES, DS and SS. The description of the services is as follows.
  - KGS generates $ISK$s to help PKG/data owners with key generation/authority delegation.
  - ES generates $IT$s to help data owners with encryption and authority delegation.
  - RES generates $RCT$s to help data owners with authority delegation.
  - DS generates $CT'$ to help data consumers with decryption.
  - SS stores all the encrypted cloud data.
- Public cloud 1 equipped with $PK$ is another different cloud service provider. It is deployed to provide only outsourced computation service, such as KGS and ES.
- Data consumer equipped with a retrieval key $RtK$ can download any encrypted data of his/her interest from public cloud 2 and try to decrypt the ciphertext.

Let $S, S'$ represent two sets of attributes, $(M, \rho)$, $(M', \rho')$ be two access structures. The fully outsourced ciphertext-policy attribute based proxy re-encryption key encapsulation mechanism (FO-CP-AB-PRE-KEM) for access structure space $\mathcal{G}$ consists of 10 algorithms:

- $Setup(\lambda, U) \rightarrow (PK, MSK)$. It is performed by PKG. On input a security parameter $\lambda$ and a universe description $U$, it outputs public parameters $PK$ and a master secret key $MSK$.
- $KeyGen.out(PK, N) \rightarrow ISK$. It is performed by KGS. On input public parameters $PK$ and a number $N$ which assumes the number of attributes, it outputs an intermediate private key $ISK$.
- $KeyGen.pkg(MSK, S, ISK1, ISK2) \rightarrow SK$. It is performed by PKG. On input a master secret key $MSK$, an attribute set $S$ and two intermediate private keys $ISK1$, $ISK2$, it outputs a private key $SK$.
- $KeyGen.ran(SK) \rightarrow (TK, RtK)$. It is performed by PKG. On input a private key $SK$, it outputs a conversion key $TK$ and a corresponding retrieval key $RtK$.
- $Encrypt.out(PK, N') \rightarrow IT$. It is performed by ES. On input public parameters $PK$ and a number $N'$ which assumes a maximum bound of $N'$ rows in LSSS structure, it outputs an intermediate ciphertext $IT$.
- $Encrypt.user(PK, IT1, IT2, (M, \rho)) \rightarrow (CT, key)$. It is performed by data owners. On input public parameters $PK$, two intermediate ciphertexts $IT1$, $IT2$ and an access structure $(M, \rho)$, it outputs a ciphertext $CT$ and an encapsulated key $key$.
- $RKeyGen.user(PK, SK, ISK1, ISK2, IT1, IT2, (M', \rho')) \rightarrow RK$. It is performed by data owners. On input public parameters $PK$, a private key $SK$ associating with an attribute set $S$, two intermediate private keys $ISK1, ISK2$, two intermediate ciphertexts $IT1, IT2$ and an access structure $(M', \rho')$, it outputs a re-encryption key $RK$ associated with the delegation from the attribute set $S$ to the access structure $(M', \rho')$.
- $ReEnc(PK, CT, RK) \rightarrow RCT$. It is performed by RES. On input public parameters $PK$, an original ciphertext $CT$ under access structure $(M, \rho)$ and a re-encryption key $RK$ corresponding to the delegation from the attribute set $S$ to the access structure $(M', \rho')$, if $CT$ is well-formed original ciphertext and $S$ satisfies $(M, \rho)$, it outputs a re-encrypted ciphertext $RCT$, otherwise $\perp$.
- $Decrypt.out(TK, CT/RCT) \rightarrow CT'$. It is performed by DS. On input a conversion key $TK$ and a ciphertext $CT$ (or a re-encrypted ciphertext $RCT$), it outputs a transformed ciphertext $CT'$ or $\perp$.
- $Decrypt.user(RtK, CT') \rightarrow key$. It is performed by data consumers. On input a retrieval key $RtK$ and a transformed ciphertext $CT'$, it outputs an encapsulated key $key$.

*Correctness.* We require that FO-CP-AB-PRE-KEM is correct, i.e., the *Decrypt.out* and *Decrypt.user* algorithms correctly decrypt a ciphertext/re-encrypted ciphertext of an access policy $\mathbb{A}$ with a secret key on an attribute set $S$, when $S$ satisfies the access policy $\mathbb{A}$. Formally, for a fixed universe description $U$ and $\lambda \in \mathbb{N}$, the FO-CP-AB-PRE-KEM correctness property requires that for all $(PK, MSK) \in Setup(\lambda, U)$, all $S, S' \subseteq U$, all $(M, \rho), (M', \rho') \in \mathcal{G}$, all $ISK1, ISK2, ISK3, ISK4, ISK5, ISK6 \in KeyGen.out(PK, N)$, all $SK \in KeyGen.pkg(MSK, S, ISK1, ISK2)$, all $SK' \in KeyGen.pkg(MSK, S', ISK3, ISK4)$, all $(TK, RtK) \in KeyGen.rand(SK)$, all $(TK', RtK') \in KeyGen.rand(SK')$, all $IT1, IT2, IT3, IT4 \in Encrypt.out(PK, N')$, all $(CT, key) \in Encrypt.user(PK, IT1, IT2, (M, \rho))RK \in$, all $RKeyGen.user(PK, SK, ISK5, ISK6, IT3, IT4, (M', \rho'))$, all $RCT \in ReEnc(PK, CT, RK)$, all $CT' \in Decrypt.out(TK, CT)$, all $CT'' \in Decrypt.out(TK', RCT)$, if $S$ satisfies $(M, \rho)$, $Decrypt.user(RtK, CT') \rightarrow key$. If $S$ satisfies $(M, \rho)$ and $S'$ satisfies $(M', \rho')$, $Decrypt.user(RtK', CT'') \rightarrow key$.

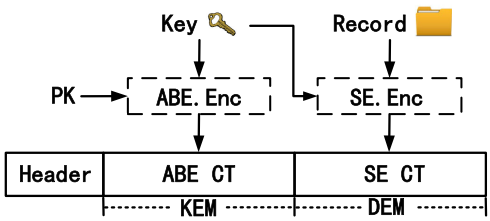



Fig. 3. Encrypted data structure.

*System Initialization.* PKG runs *Setup* to generate $PK$ and $MSK$, then broadcasts $PK$ to all the entities and keeps $MSK$ locally. Data consumers are assigned to several attributes (such as "Surgery", "Director", "Female" etc.) according to their roles in the system. With the help of KGS (running *KeyGen.out*), PKG downloads $ISK$s from two cloud servers and runs *KeyGen.pkg* with an attribute set and $MSK$ to obtain a user private key $SK$, which can be used to decrypt ciphertexts. Then PKG randomizes $SK$ with a blind factor to obtain the convention key $TK$ and the corresponding retrieve key $RtK$ by running *KeyGen.ran*, where $TK$ is a randomized version of $SK$ and is used to do outsourced decryption, $RtK$ is the blind factor and is used to retrieve the plaintext. At last, PKG sends $TK$ to public cloud 2 and sends $RtK$ and $SK$ to the data consumer.

*Data Encryption.* With the help of ES (running *Encrypt.out*), the data owner downloads $IT$s from two cloud servers and encrypts the data in the Key/Data Encapsulation Mechanism (KEM/DEM) setting [30][3] (see Fig. 3). The data owner first defines the access policy, e.g., "Paediatrics" $\wedge$ ("Doctor" $\vee$ "Head-nurse"), then creates an ABE ciphertext $(CT, key)$ with the access policy by running *Encrypt.user*, where $key$ is an encapsulated AES key. Next, the data owner encrypts the data by running the AES algorithm with $key$. At last, the data owner uploads the encrypted data to public cloud 2.

With the help of KGS, ES and RES, the data owner can achieve efficient authority delegation by downloading $ISK$s, $IT$s from two cloud servers, running *RKeyGen.user* to create a re-encryption key $RK$ with the new access policy and uploading $(CT, RK)$ to public cloud 2. RES in public cloud 2 can later run *ReEnc* with the new access policy to generate the re-encrypted ciphertext $RCT$.

*Data Decryption.* Data consumers can download any encrypted data of his/her interest from public cloud 2. Receiving the decryption request, DS in public cloud 2 equipped with $TK$ runs *Decrypt.out* with ABE ciphertext $CT/RCT$ to obtain a transformed ciphertext $CT'$, then sends $CT'$ and the AES ciphertext to the data consumer, who runs *Decrypt.user* with $RtK$ to recover the encapsulated AES key from $CT'$ , and decrypts the AES ciphertext.

### 3.2 Security Model

*Adversarial Model.* In our system, public cloud 1 and public cloud 2 are "honest-but-curious" [19], [31]. More precisely, they will follow the protocol but try to find out as much private information as possible. Most of the data consumers are honest, while few of them are corrupt and will leakage their secret keys in the collusion. On the contrary, PKG and data owner are assumed to be fully trusted. Besides, public cloud 1 and public cloud 2 cannot collude with each other [32], [33]. The non-collusive assumption is reasonable, because the client can demand that two cloud severs cannot reveal users' information by contract.

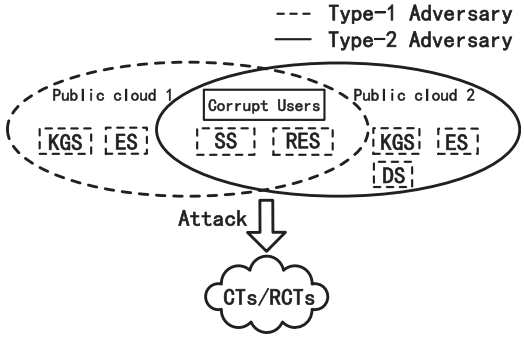



Fig. 4. Two types of adversaries.

The channels that transmit $ISK, IT, TK, RtK$ should be secure and this can be easily implemented by SSL. Because the secret keys $TK, RtK$ are always distributed privately, and the intermediate computation results $ISK, IT$ cannot be accessed by outsiders. If the channels that transmit $ISK, IT$ are public, meaning $ISK, IT$ can be accessed by any entities, the outsourced key generation/encryption/re-encryption generation are not secure. Since we adopt two non-collusive public cloud servers, we should take into account the circumstances that each cloud server colludes with other entities. As depicted in Fig. 4, we consider the following two types of adversaries:

- Type-1 adversary refers to corrupt data consumers colluding with public cloud 1, who can obtain $SK$s of corrupt data consumers, all the $ISK$1s/$IT$1s at public cloud 1, some $RK$s of RES and all the $CT$s/$RCT$s of SS.[4] It intends to decrypt unauthorized $CT$s/$RCT$s.
- Type-2 adversary refers to corrupt data consumers colluding with public cloud 2, who can obtain $SK$s of corrupt data consumers, $ISK$2s/$IT$2s/$RK$s[5]/$TK$s/$CT$s/$RCT$s at public cloud 2. It intends to decrypt unauthorized $CT$s/$RCT$s.

Next, according to the capabilities of two different adversaries and the attack targets ($CT$ or $RCT$), we define the following selective CPA security game.

*Selective CPA Security at Original Ciphertext (CT).* According to two types of adversaries, selective CPA security at CT for FO-CP-AB-PRE-KEM is described as follows.

Selective CPA Security Game at CT for Type-1 Adversary.

*Init.* The adversary $\mathcal{A}$ sends the challenge access policy $\mathbb{A}^*$ to the challenger $\mathcal{C}$.

*Setup.* $\mathcal{C}$ runs *Setup* and gives $PK$ to $\mathcal{A}$.

*Phase 1.* $\mathcal{C}$ initializes two empty tables $T_1$, $T_2$ and an integer counter $j = 0$. $\mathcal{A}$ can adaptively issue the following queries:

3. A key encapsulation mechanism, where the public key (ABE) ciphertext encapsulates a symmetric encryption (SE) key which could be used to encrypt the plaintext. The SE ciphertext is the DEM part.

4. Since the ciphertexts should be public, we consider type-1 adversary with the knowledge of $CT$s, $RK$s, $RCT$s.

5. The $RK$s that accesses by the adversary are limited by some conditions, please refer to the security definition in Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TDSC.2018.2844814.

- *Create*$(S)$: $\mathcal{C}$ first checks whether $S$ satisfies $\mathbb{A}^*$, if so, returns $\perp$. Otherwise, $\mathcal{C}$ sets $j = j + 1$. It runs *KeyGen.out* with $PK$ twice to obtain $ISK1, ISK2$, and runs *KeyGen.PKG*$(MSK, S, ISK1, ISK2)$ to obtain $SK$, then runs *Encrypt.out* with $PK$ to obtain $IT1$. It stores the entry $(j, S, SK, ISK1, IT1)$ in table $T_1$.
- *Corrupt.SK*$(i)$: If the $i$th entry $(i, S, SK)$ is in $T_1$, $\mathcal{C}$ returns $SK$, or returns $\perp$. Note that $S$ in the entry $(i, S, SK)$ cannot satisfy $\mathbb{A}$ that has been queried to $Corrupt.RK$ along with $S'$ satisfying $\mathbb{A}^*$. It captures the case of corrupt users who have their private keys.
- *Corrupt.IT1*$(i)$: If the $i$th entry $(i, S, IT1)$ is in $T_1$, $\mathcal{C}$ returns $IT1$, or returns $\perp$. It captures the case of the encryption service (ES) on public cloud 1.
- *Corrupt.ISK1*$(i)$: If the $i$th entry $(i, S, SK, ISK1)$ is in $T_1$, $\mathcal{C}$ returns $ISK1$, or returns $\perp$. It captures the case of the key generation service (KGS) on public cloud 1.
- *Corrupt.RK*$(i, S', \mathbb{A})$: If the entry $(i, S', SK)$ is in $T_1$, $\mathcal{C}$ obtains $SK$. Otherwise, $\mathcal{C}$ runs *Create(S')* and *Corrupt.SK* to obtain $SK$, then runs *KeyGen.out* and *Encrypt.out* twice respectively to obtain $ISK1, ISK2, IT1, IT2$, and runs *RKeyGen.user* to obtain $RK$, it stores $(S', \mathbb{A}, RK)$ in $T_2$. Note that any $S$ queried to *Corrupt.SK* cannot satisfy $\mathbb{A}$, if $S'$ satisfies $\mathbb{A}^*$. It means that the adversary can obtain re-encryption keys from RES.
- *ReEnc* $(C, S', \mathbb{A})$: $\mathcal{C}$ checks whether the entry $(S', A, RK)$ is in $T_2$. If so, it obtains $RK$. If not, $\mathcal{C}$ runs *Corrupt.RK* to obtain $RK$. Then $\mathcal{C}$ runs *ReEnc* to output the corresponding $RCT$ to $\mathcal{A}$. Note that any $S$ queried to *Corrupt.SK* cannot satisfy $\mathbb{A}$, if $S'$ satisfies $\mathbb{A}^*$. It means that the adversary can generate RCTs with RES.

*Challenge.* $\mathcal{C}$ runs *Encrypt.out* two times to get $IT1^*, IT2^*$ and runs *Encrypt.user* to obtain $(key^*, CT^*)$. $\mathcal{C}$ selects a random bit $b \in \{0, 1\}$. If $b = 0$, it returns $(key^*, CT^*, IT1^*)$ to $\mathcal{A}$. If $b = 1$, it picks a random key $R^*$ in the encapsulated key space and returns $(R^*, CT^*, IT1^*)$.

*Phase 2.* Phase 1 is repeated with the above restrictions.

*Guess.* $\mathcal{A}$ outputs a guess $b'$ of $b$. $\mathcal{A}$ wins the game if $b = b'$.

**Definition 4.** *A FO-CP-AB-PRE-KEM scheme is selective CPA-secure at original ciphertext against type-1 adversary if all probabilistic polynomial time (PPT) type-1 adversaries have at most a negligible advantage in the security game, denote:* $\epsilon_1 = |\Pr[b = b'] - \frac{1}{2}| \leq negl(\lambda)$.

Selective CPA Security Game at CT for Type $-$ 2 Adversary.

*Init*, *Setup*, *Phase 2* and *Guess* are the same as the above game. *Phase 1* and *Challenge* are as follows.

*Phase 1.* $\mathcal{C}$ initializes two empty tables $T_1$, $T_2$ and an integer counter $j = 0$. $\mathcal{A}$ can adaptively issue the following queries:

- *Create*$(S)$: $\mathcal{C}$ first checks whether $S$ satisfies $\mathbb{A}^*$, if so, returns $\perp$. Otherwise, $\mathcal{C}$ sets $j = j + 1$, runs *KeyGen.out* with $PK$ twice to obtain $ISK1, ISK2$, runs *KeyGen.pkg*$(MSK, S, ISK1, ISK2)$ to obtain $SK$, runs *KeyGen.ran* with $SK$ to obtain $TK, RtK$, and runs *Encrypt.out* once with $PK$ to obtain $IT2$. It stores the entry $(j, S, SK, TK, RtK, ISK2, IT2)$ in table $T_1$.
- *Corrupt.SK*$(i)$: Same as the above game.
- *Corrupt.TK*$(i)$: If the $i$th entry $(i, S, TK)$ is in $T_1$, $\mathcal{C}$ returns $TK$, or $\mathcal{C}$ returns $\perp$. It captures the case of the decryption service (DS) on public cloud 2.
- *Corrupt.IT2*$(i)$: Same as the above game except that $IT1$ is replaced by $IT2$. It captures the case of the encryption service on public cloud 2.
- *Corrupt.ISK2*$(i)$: Same as the above game except that $ISK1$ is replaced by $ISK2$. It captures the case of the key generation service on public cloud 2.
- *Corrupt.RK*$(i, S', \mathbb{A})$: Same as the above game.
- *ReEnc*$(C, S', \mathbb{A})$: Same as the above game.

*Challenge.* It is the same as that in the game for type-1 adversary except that $IT1^*$ is replaced by $IT2^*$.

**Definition 5.** *A FO-CP-AB-PRE-KEM scheme is selective CPA-secure at original ciphertext against type-2 adversary if all PPT type-2 adversaries have at most a negligible advantage in the security game, denote* $\epsilon_2 = |\Pr[b = b'] - \frac{1}{2}| \leq negl(\lambda)$.

*Selective CPA Security at Re-Encrypted Ciphertext (RCT).* According to different adversaries, selective CPA security at RCT for FO-CP-AB-PRE-KEM is described as follows.

Selective CPA Security Game at RCT for Type-1 Adversary.

*Init.* The adversary $\mathcal{A}$ sends the challenge access policy $\mathbb{A}^*$ to the challenger $\mathcal{C}$.

*Setup.* $\mathcal{C}$ runs *Setup* and gives $PK$ to $\mathcal{A}$.

*Phase 1.* $\mathcal{C}$ initializes two empty tables $T_1$, $T_2$ and an integer counter $j = 0$. $\mathcal{A}$ can adaptively issue the following queries:

- *Create*$(S)$: $\mathcal{C}$ first checks whether $S$ satisfies $\mathbb{A}^*$, if so, returns $\perp$. Otherwise, $\mathcal{C}$ sets $j = j + 1$. It runs *KeyGen.out* with $PK$ twice to obtain $ISK1, ISK2$, and runs *KeyGen.PKG*$(MSK, S, ISK1, ISK2)$ to obtain $SK$, then runs *Encrypt.out* with $PK$ to obtain $IT1$. It stores the entry $(j, S, SK, ISK1, IT1)$ in table $T_1$.
- *Corrupt.SK*$(i)$: If the $i$th entry $(i, S, SK)$ is in $T_1$, $\mathcal{C}$ returns $SK$, otherwise $\mathcal{C}$ returns $\perp$. Note that $S$ in the entry $(i, S, SK)$ cannot satisfy the policy of the original ciphertext $\mathbb{A}$ that corresponds to the challenge re-encrypted ciphertext $RCT^*$. It captures the case of the corrupt users who have their private keys.
- *Corrupt.IT1*$(i)$: If the $i$th entry $(i, S, IT1)$ is in $T_1$, $\mathcal{C}$ returns $IT1$, or $\mathcal{C}$ returns $\perp$. It captures the case of the encryption service on public cloud 1.
- *Corrupt.ISK1*$(i)$: If the $i$th entry $(i, S, SK, ISK1)$ is in $T_1$, $\mathcal{C}$ returns $ISK1$, or returns $\perp$. It captures the case of the key generation service on public cloud 1.
- *Corrupt.RK*$(i, S', \mathbb{A})$: $\mathcal{C}$ checks whether the entry $(i, S', SK)$ is in table $T_1$. If so, $\mathcal{C}$ obtains $SK$, otherwise, $\mathcal{C}$ runs *Create(S')* and *Corrupt.SK* to obtain $SK$. Then $\mathcal{C}$ runs *KeyGen.out* and *Encrypt.out* twice respectively to obtain $ISK1, ISK2, IT1, IT2$, and runs *RKeyGen.user* to obtain $RK$, it stores $(S', \mathbb{A}, RK)$ in $T_2$. It means that the adversary can obtain $RK$s from RES.
- *ReEnc*$(C, S', \mathbb{A})$: $\mathcal{C}$ checks whether the entry $(S', \mathbb{A}, RK)$ is in table $T_2$. If not, $\mathcal{C}$ runs *Corrupt.RK* to obtain $RK$. Then $\mathcal{C}$ runs *ReEnc* to output the corresponding $RCT$ to $\mathcal{A}$. It means that the adversary can generate $RCT$s with RES.

*Challenge.* $\mathcal{C}$ runs *Encrypt.out* two times to get $IT1^*, IT2^*$ and runs *Encrypt.user* with access policy $\mathbb{A}$ to obtain $(key^*, CT^*)$. Note that any $S$ queried *Corrupt.SK* cannot satisfy $\mathbb{A}$. $\mathcal{C}$ selects a random bit $b \in \{0, 1\}$. If $b = 0$, it sets $K^* = key^*$. If $b = 1$, it picks a random key $R^*$ in the encapsulated key space and sets $K^* = R^*$. $\mathcal{C}$ selects an attribute set $S$ that

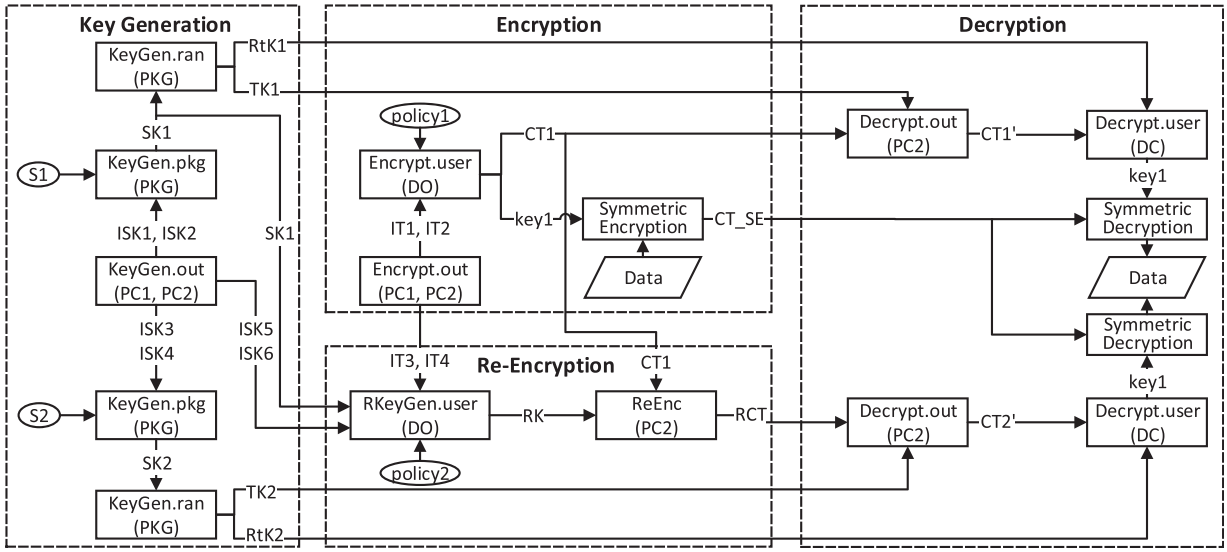



Fig. 5. The flow chart of FO-CP-AB-PRE-KEM. ‡ Let PC1, PC2, DO, DC denote the public cloud 1, the public cloud 2, the data owner, and the data consumer respectively.

satisfies $\mathbb{A}$ and runs *Create(S)* to obtain $SK$, then runs *KeyGen.out*, *Encrypt.out* twice to obtain $I\widehat{SK}1^*, I\widehat{SK}2^*, \widehat{IT1}^*$, $\widehat{IT2}^*$. Next, $\mathcal{C}$ runs *RKeyGen.user*$(PK, SK, I\widehat{SK}1^*, I\widehat{SK}2^*, \widehat{IT1}^*, \widehat{IT2}^*, \mathbb{A}^*)$ to get $RK$. At last, $\mathcal{C}$ runs *ReEnc*$(PK, CT^*, RK)$ to obtain $RCT^*$ and returns $(K^*, IT1^*, \widehat{ISK1}^*, \widehat{IT1}^*, RCT^*)$ to $\mathcal{A}$.

*Phase 2*. Phase 1 is repeated with the above restrictions.

*Guess*. $\mathcal{A}$ outputs a guess $b'$ of $b$. $\mathcal{A}$ wins the game if $b = b'$.

**Definition 6.** *A FO-CP-AB-PRE-KEM scheme is selective CPA-secure at re-encrypted ciphertext against type-1 adversary if all PPT type-1 adversaries have at most a negligible advantage in the security game, denote* $\epsilon_3 = |\Pr[b = b'] - \frac{1}{2}| \leq negl(\lambda)$.

Selective CPA Security Game at RCT for Type $-$ 2 Adversary.

*Init*, *Setup*, *Phase 2* and *Guess* are the same as the above game. *Phase 1* and *Challenge* are as follows.

*Phase 1*. $\mathcal{C}$ initializes two empty tables $T_1$, $T_2$ and an integer counter $j = 0$. $\mathcal{A}$ can adaptively issue the following queries:

- *Create*$(S)$: $\mathcal{C}$ first checks whether $S$ satisfies $\mathbb{A}^*$, if so, returns $\perp$. Otherwise, $\mathcal{C}$ sets $j = j + 1$, runs *KeyGen.out* with $PK$ twice to obtain $ISK1, ISK2$, and runs *KeyGen.pkg*$(MSK, S, ISK1, ISK2)$ to obtain $SK$, and runs *KeyGen.ran* with $SK$ to obtain $TK, RtK$ and runs *Encrypt.out* once with $PK$ to obtain $IT2$. It stores the entry $(j, S, SK, TK, RtK, ISK2, IT2)$ in table $T_1$.
- *Corrupt.SK*$(i)$: Same as the above game.
- *Corrupt.TK*$(i)$: If the $i$th entry $(i, S, TK)$ is in $T_1$, $\mathcal{C}$ returns $TK$, or returns $\perp$. It captures the case of the decryption service on public cloud 2.
- *Corrupt.IT2*$(i)$: Same as the above game except that $IT1$ is replaced by $IT2$. It captures the case of the encryption service on public cloud 2.
- *Corrupt.ISK2*$(i)$: Same as the above game except that $ISK1$ is replaced by $ISK2$. It captures the case of the key generation service on public cloud 2.
- *Corrupt.RK*$(i, S', \mathbb{A})$: Same as the above game.
- *ReEnc*$(C, S', \mathbb{A})$: Same as the above game.

*Challenge*. It is the same as that in the game for type-1 adversary except that the challenge $(K^*, IT1^*, I\widehat{SK}1^*, \widehat{IT1}^*, RCT^*)$ is replaced by $(K^*, IT2^*, I\widehat{SK}2^*, \widehat{IT2}^*, RCT^*)$.

**Definition 7.** *A FO-CP-AB-PRE-KEM scheme is selective CPA-secure at re-encrypted ciphertext against type-2 adversary if all PPT type-2 adversaries have at most a negligible advantage in the security game, denote* $\epsilon_4 = |\Pr[b = b'] - \frac{1}{2}| \leq negl(\lambda)$.

## 4 Fully Outsourced Ciphertext-Policy Attribute Based Proxy Re-Encryption

We present a fully outsourced ciphertext-policy attribute based proxy re-encryption key encapsulation mechanism (FO-CP-AB-PRE-KEM) system and prove its selective CPA security at CT/RCT in the standard model.

### 4.1 The Construction

The construction includes four main functions: Key Generation (i.e., *KeyGen.out*, *KeyGen.pkg*, *KeyGen.ran*), Encryption (i.e., *Encrypt.out*, *Encrypt.user*), Re-Encryption (i.e., *RKeyGen.user*, *ReEnc*) and Decryption (i.e., *Decrypt.out*, *Decrypt.user*). Fig. 5 presents the whole process. Let S1, S2 be two attribute sets, policy1, policy2 be two access policies, where S1 satisfies policy1, and S2 satisfies policy2 but not policy1. The encryption phase utilizes hybrid encryption [30]. The KEM part is CT1 which encapsulates a symmetric secret key key1, and the ciphertext CT_SE that encrypted by the symmetric encryption algorithm with key1 is the DEM part.

Before presenting the details, we give some intuitions of our construction. To achieve secure outsourced key generation/encryption, we adopt two non-collusive cloud servers, then if $ISKs/ITs$ are generated randomly by *KeyGen.out*/*Encrypt.out*, the final combined $ISK/IT$ by the PKG/user should be information-theoretically hidden from two servers. Actually, one re-encryption key generation operation consists of one key generation operation and one encryption operation, thus it can be outsourced with the help of

*KeyGen.out* and *Encrypt.out*. Besides, every user in the system has a dummy attribute $A_{-1}$ which is used to protect the security of *RCT*s. We utilize key blinding technique [18] to achieve outsourced decryption. The FO-CP-AB-PRE-KEM scheme consists of 10 algorithms:

*Setup*$(\lambda, U)$. The algorithm chooses a bilinear map $D = (\mathbb{G}, \mathbb{G}_T, e, p)$, where $p \in \Theta(2^\lambda)$ is the prime order of the groups $\mathbb{G}$ and $\mathbb{G}_T$. The attribute universe is a set of elements in $\mathbb{Z}_p$. It chooses random $g, h, u, v, w \in \mathbb{G}$, picks a random $\alpha \in \mathbb{Z}_p$ and a cryptographic secure hash function $H(\cdot) : \mathbb{G}_T \to \mathbb{G}$. It outputs $PK = (D, g, h, u, v, w, e(g,g)^\alpha, H)$, $MSK = (PK, \alpha)$.

*KeyGen.out*$(PK, N)$. On input public parameters $PK$ and a number $N$ which assumes the number of attributes, the algorithm picks $2N+2$ random $\alpha', r', r'_1, r'_2, \ldots, r'_N, a'_1, a'_2, \ldots, a'_N \in \mathbb{Z}_p$ and computes $K_0 = g^{\alpha'} w^{r'}, K_1 = g^{r'}$. Then for $i = 1$ to $N$, it computes $K_{i,2} = g^{r'_i}$, $K_{i,3} = (u^{a'_i} h)^{r'_i} v^{-r'}$. At last, it outputs $ISK = (PK, \alpha', r', K_0, K_1, \{r'_i, a'_i, K_{i,2}, K_{i,3}\}_{i\in[1,N]})$.

*KeyGen.pkg*$(MSK, S, ISK1, ISK2)$. On input a master secret key $MSK$, an attribute set $S = \{A_{-1}, A_1, A_2, \ldots, A_k\} \subseteq \mathbb{Z}_p$ $(k+1 \le N)$ and two intermediate private keys $ISK1$, $ISK2$:

$$ISK1 = (\alpha', r', K'_0, K'_1, \{r'_i, a'_i, K'_{i,2}, K'_{i,3}\}_{i\in[-1,k]}),$$
$$ISK2 = (\alpha'', r'', K''_0, K''_1, \{r''_i, a''_i, K''_{i,2}, K''_{i,3}\}_{i\in[-1,k]}),$$

the algorithm first recomputes $\widetilde{\alpha}$, $r$, $K_0$ and $K_1$,

$$\widetilde{\alpha} = \alpha' + \alpha'',\ r = r' + r'',\ K_1 = K'_1 \cdot K''_1 = g^{r'} \cdot g^{r''} = g^r,$$
$$K_0 = K'_0 \cdot K''_0 = g^{\alpha'+\alpha''} w^{r'+r''} = g^{\widetilde{\alpha}} w^r.$$

Then for $i = -1$ to $k$, recompute the following parameters:

$$r_i = r'_i + r''_i,\ a_i = \frac{a'_i r'_i + a''_i r''_i}{r_i},\ K_{i,2} = K'_{i,2} \cdot K''_{i,2} = g^{r_i},$$
$$K_{i,3} = K'_{i,3} \cdot K''_{i,3} = \left(u^{\frac{a'_i r'_i + a''_i r''_i}{r_i}} h\right)^{r_i} v^{-r} = (u^{a_i} h)^{r_i} v^{-r}.$$

It obtains the combined $ISK = (\widetilde{\alpha}, r, K_0, K_1, \{r_i, a_i, K_{i,2}, K_{i,3}\}_{i\in[-1,k]})$.

Next it computes $K_4 = \alpha - \widetilde{\alpha}$, then for $i = -1$ to $k$, it computes $K_{i,5} = r_i(A_i - a_i)$. This will correct to the master key and the proper attribute. It outputs the private key $SK = (S, PK, K_0, K_1, K_4, \{K_{i,2}, K_{i,3}, K_{i,5}\}_{i\in[-1,k]})$.

*KeyGen.ran*$(SK)$. On input a private key $SK$, the algorithm chooses a random $\tau \in \mathbb{Z}_p^*$, then computes

$$K'_0 = K_0^{1/\tau} = g^{\widetilde{\alpha}/\tau} w^{r/\tau},\quad K'_1 = K_1^{1/\tau} = g^{r/\tau},$$
$$K'_4 = K_4/\tau = (\alpha - \widetilde{\alpha})/\tau.$$

For $i = -1$ to $k$, it computes

$$K'_{i,2} = K_{i,2}^{1/\tau} = g^{r_i/\tau},\quad K'_{i,3} = K_{i,3}^{1/\tau} = (u^{a_i} h)^{r_i/\tau} v^{-r/\tau},$$
$$K'_{i,5} = K_{i,5}/\tau = r_i(A_i - a_i)/\tau.$$

It outputs the conversion key $TK = (S, PK, K'_0, K'_1, K'_4, \{K'_{i,2}, K'_{i,3}, K'_{i,5}\}_{i\in[-1,k]})$, the retrieval key $RtK = (PK, \tau)$.

*Encrypt.out*$(PK, N')$. On input public parameters $PK$ and a maximum bound of $N'$ rows in LSSS access structure, the algorithm first picks a random $s' \in \mathbb{Z}_p$ and computes $\widehat{C} = e(g,g)^{\alpha s'}, C_0 = g^{s'}$. Then for $j = 1$ to $N'$, choose random $\lambda'_j, x'_j, t'_j \in \mathbb{Z}_p$ and compute

$$C_{j,1} = w^{\lambda'_j} v^{t'_j},\quad C_{j,2} = (u^{x'_j} h)^{-t'_j},\quad C_{j,3} = g^{t'_j}.$$

It outputs the intermediate ciphertext $IT = (s', \hat{C}, C_0, \{\lambda'_j, t'_j, x'_j, C_{j,1}, C_{j,2}, C_{j,3}\}_{j\in[1,N']})$.

*Encrypt.user*$(PK, IT1, IT2, (M, \rho))$. On input the public parameters $PK$, an LSSS access structure $(M, \rho)$, where $M$ is an $l \times n$ matrix and $l \le N'$. Note that $(M, \rho)$ does not contain the attribute $A_{-1}$, and two intermediate ciphertexts

$$IT1 = \left(s', \widehat{C'}, C'_0, \{\lambda'_j, t'_j, x'_j, C'_{j,1}, C'_{j,2}, C'_{j,3}\}_{j\in[1,l]}\right),$$
$$IT2 = \left(s'', \widehat{C''}, C''_0, \{\lambda''_j, t''_j, x''_j, C''_{j,1}, C''_{j,2}, C''_{j,3}\}_{j\in[1,l]}\right).$$

The algorithm first recomputes $s$, $C_0$ and $\widehat{C}$,

$$s = s' + s'',\quad C_0 = C'_0 \cdot C''_0 = g^{s'} \cdot g^{s''} = g^s,$$
$$\widehat{C} = \widehat{C'} \cdot \widehat{C''} = e(g,g)^{\alpha s'} \cdot e(g,g)^{\alpha s''} = e(g,g)^{\alpha s}.$$

Then for $j = 1$ to $l$, recompute the following parameters:

$$\lambda_j = \lambda'_j + \lambda''_j,\quad t_j = t'_j + t''_j,\quad x_j = \frac{x'_j t'_j + x''_j t''_j}{t'_j + t''_j},$$
$$C_{j,1} = C'_{j,1} \cdot C''_{j,1} = \omega^{\lambda'_j} v^{t'_j} \cdot \omega^{\lambda''_j} v^{t''_j} = \omega^{\lambda_j} v^{t_j},$$
$$C_{j,2} = C'_{j,2} \cdot C''_{j,2} = \left(u^{\frac{x'_j t'_j + x''_j t''_j}{t'_j + t''_j}} h\right)^{-t'_j - t''_j} = (u^{x_j} h)^{-t_j},$$
$$C_{j,3} = C'_{j,3} \cdot C''_{j,3} = g^{t'_j} \cdot g^{t''_j} = g^{t_j}.$$

The user obtains the combined intermediate ciphertext $IT = (s, \widehat{C}, C_0, \{\lambda_j, t_j, x_j, C_{j,1}, C_{j,2}, C_{j,3}\}_{j\in[1,l]})$.

Set $key = \widehat{C} = e(g,g)^{\alpha s}$, then pick random $y_2, \ldots, y_n \in Z_p$, set the vector $\overrightarrow{y} = (s, y_2, \ldots, y_n)^T$ and compute a vector of shares of $s$ as $(\hat{\lambda}_1, \ldots, \hat{\lambda}_l)^T = M\overrightarrow{y}$. For $j = 1$ to $l$, compute $C_{j,4} = \hat{\lambda}_j - \lambda_j, C_{j,5} = t_j(x_j - \rho(j))$. This will correct to the shares of $s$ and the proper attribute $\rho(j)$. It outputs the encapsulated key $key$ and the ciphertext $CT = ((M, \rho), C_0, \{C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}, C_{j,5}\}_{j\in[1,l]})$.

*RKeyGen.user*$(PK, SK, ISK1, ISK2, IT1, IT2, (M', \rho'))$. On input public parameters $PK$, a private key $SK = (S, PK, K_0, K_1, K_4, \{K_{i,2}, K_{i,3}, K_{i,5}\}_{i\in[-1,k]})$ associating with an attribute set $S = \{A_{-1}, A_1, A_2, \ldots, A_k\} \subseteq \mathbb{Z}_p$, two intermediate private keys $ISK1$, $ISK2$, two intermediate ciphertexts $IT1$, $IT2$ and an access structure $(M', \rho')$, where $M'$ is an $l' \times n'$ matrix and $l' \le N'$. Note that $(M', \rho')$ must contain the attribute $A_{-1}$. The algorithm combines $ISK1$ and $ISK2$ to

$$ISK = (\widetilde{\alpha}', r', K'_0, K'_1, \{r'_i, a'_i, K'_{i,2}, K'_{i,3}\}_{i\in[1,k]}),$$

as *KeyGen.pkg* does, and combines $IT1$ and $IT2$ to

$$IT = \left(s_r, \widehat{C'}, C'_0, \{\lambda'_j, t'_j, x'_j, C'_{j,1}, C'_{j,2}, C'_{j,3}\}_{j\in[1,l']}\right),$$

as *Encrypt.user* does. Then it computes

$$K_0'' = K_0 \cdot K_0' = g^{\widetilde{\alpha}} w^r \cdot g^{\widetilde{\alpha}'} w^{r'} = g^{\widetilde{\alpha}+\widetilde{\alpha}'} w^{r+r'},$$
$$K_1'' = K_1 \cdot K_1' = g^r \cdot g^{r'} = g^{r+r'},$$
$$K_4'' = K_4 - \widetilde{\alpha}' = \alpha - \widetilde{\alpha} - \widetilde{\alpha}'.$$

For $i = 1$ to $k$, it computes

$$K_{i,2}'' = K_{i,2} \cdot K_{i,2}' = g^{r_i} \cdot g^{r_i'} = g^{r_i + r_i'},$$
$$K_{i,3}'' = K_{i,3} \cdot K_{i,3}' = (u^{a_i} h)^{r_i} v^{-r} \cdot (u^{a_i'} h)^{r_i'} v^{-r'}$$
$$= (u^{\frac{a_i r_i + a_i' r_i'}{r_i + r_i'}} h)^{r_i + r_i'} v^{-r-r'},$$
$$K_{i,5}'' = K_{i,5} + r_i'(A_i - a_i') = r_i(A_i - a_i) + r_i'(A_i - a_i').$$

It sets $key' = \widehat{C'} = e(g,g)^{\alpha \cdot s_r}, rk = H(key') \cdot K_0''$. Then pick random $y_2, \ldots, y_n \in Z_p$, set the vector $\overrightarrow{y} = (s_r, y_2, \ldots, y_n)^T$ and compute a vector of shares of $s_r$ as $(\hat{\lambda}_1, \ldots, \hat{\lambda}_l)^T = M' \overrightarrow{y}$. For $j = 1$ to $l'$, compute $C_{j,4}' = \hat{\lambda}_j - \lambda_j', \ C_{j,5}' = t_j'(x_j' - \rho'(j))$. It outputs the re-encryption key $RK = (S, (M', \rho'), K_1'', K_4'', \{K_{i,2}'', K_{i,3}'', K_{i,5}''\}_{i \in [1,k]}, rk, C_0', \{C_{j,1}', C_{j,2}', C_{j,3}', C_{j,4}', C_{j,5}'\}_{j \in [1,l']})$.

*ReEnc*$(PK, CT, RK)$. On input public parameters $PK$, an original ciphertext $CT = ((M, \rho), C_0, \{C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}, C_{j,5}\}_{j \in [1,l]})$ and a re-encryption key $RK = (S, (M', \rho'), K_1'', \ K_4'', \{K_{i,2}'', K_{i,3}'', K_{i,5}''\}_{i \in [1,k]}, rk, C_0', \{C_{j,1}', C_{j,2}', C_{j,3}', C_{j,4}', C_{j,5}'\}_{j \in [1,l']})$, if $S/A_{-1}$ does not satisfy $(M, \rho)$, the algorithm outputs $\perp$. Otherwise, it calculates $I = \{i : \rho(i) \in S/A_{-1}\}$ and computes the constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that $\sum_{i \in I} \omega_i \cdot M_i = (1, 0, \ldots, 0)$, where $M_i$ is the $i$th row of the matrix $M$. Then compute

$$C' = e(\omega^{\sum_{i \in I} C_{i,4} \omega_i}, K_1'') \cdot \prod_{i \in I} (e(C_{i,1}, K_1'') \cdot e(C_{i,2} \cdot u^{C_{i,5}}, K_{j,2}'') \cdot e(C_{i,3}, K_{j,3}'' \cdot u^{K_{j,5}''}))^{\omega_i} = e(g,g)^{(r+r') \cdot s},$$

where j is the index of the attribute $\rho(i)$ in $S/A_{-1}$ (it depends on $i$). It outputs the re-encrypted ciphertext $RCT = ((M', \rho'), K_4'', C', C_0, rk, C_0', \{C_{j,1}', C_{j,2}', C_{j,3}', C_{j,4}', C_{j,5}'\}_{j \in [1,l']})$.

*Decrypt.out*$(TK, CT/RCT)$. On input an original ciphertext $CT = ((M, \rho), C_0, \{C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}, C_{j,5}\}_{j \in [1,l]})$ (or a re-encrypted ciphertext $RCT = ((M', \rho'), K_4'', \ C', C_0, rk, C_0', \{C_{j,1}', C_{j,2}', C_{j,3}', C_{j,4}', C_{j,5}'\}_{j \in [1,l']})$ ) and a conversion key $TK = (S, PK, K_0', K_1', K_4', \{K_{i,2}', K_{i,3}', K_{i,5}'\}_{i \in [-1,k]})$ for the attribute set $S$, two situations are considered as follows.

- *Case 1: an original ciphertext CT.* If $S/A_{-1}$ does not satisfy the access structure $(M, \rho)$, the algorithm outputs $\perp$. Otherwise, it calculates $I = \{i : \rho(i) \in S/A_{-1}\}$ and computes the constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that $\sum_{i \in I} \omega_i \cdot M_i = (1, 0, \ldots, 0)$, where $M_i$ is the $i$th row of the matrix $M$. Then it recovers the encapsulated key by computing

$$A = \prod_{i \in I} (e(C_{i,1}, K_1') \cdot e(C_{i,2} \cdot u^{C_{i,5}}, K_{j,2}') \cdot e(C_{i,3}, K_{j,3}' \cdot u^{K_{j,5}'}))^{\omega_i},$$
$$key' = \frac{e(C_0, K_0' \cdot g^{K_4'})}{e(\omega^{\sum_{i \in I} C_{i,4} \omega_i}, K_1') \cdot A} = e(g,g)^{\alpha s / \tau},$$

where j is the index of the attribute $\rho(i)$ in $S/A_{-1}$ (it depends on $i$). The transformed ciphertext is $CT' = ((M, \rho), key' = e(g,g)^{\alpha s / \tau})$.

- *Case 2: a re-encrypted ciphertext RCT.* If $S$ does not satisfy the access structure $(M', \rho')$, the algorithm outputs $\perp$. Otherwise, it calculates $I = \{i' : \rho'(i) \in S\}$ and computes the constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that $\sum_{i \in I} \omega_i \cdot M_i' = (1, 0, \ldots, 0)$, where $M_i'$ is the $i$th row of the matrix $M'$. Then it recovers the encapsulated key by computing

$$A = \prod_{i \in I} (e(C_{i,1}', K_1') \cdot e(C_{i,2}' \cdot u^{C_{i,5}'}, K_{j,2}') \cdot e(C_{i,3}', K_{j,3}' \cdot u^{K_{j,5}'}))^{\omega_i},$$
$$key' = \frac{e(C_0', K_0' \cdot g^{K_4'})}{e(\omega^{\sum_{i \in I} C_{i,4}' \omega_i}, K_1') \cdot A} = e(g,g)^{\alpha s_r / \tau},$$

where j is the index of the attribute $\rho(i)'$ in $S$ (it depends on $i$). The transformed ciphertext is $CT' = ((M', \rho'), K_4'', C', C_0, rk, e(g,g)^{\alpha s_r / \tau})$.

*Decrypt.user*$(RtK, CT')$. On input a retrieval key $RtK = \tau$ and a transformed ciphertext $CT'$, if $CT'$ is a transformed ciphertext of an original ciphertext, the algorithm computes $key = key'^{\tau} = e(g,g)^{(\alpha s / \tau) \cdot \tau} = e(g,g)^{\alpha s}$. If $CT'$ is a transformed ciphertext of a re-encrypted ciphertext, it computes

$$K_0'' = rk / H(key'^{\tau}), \quad key = e(C_0, K_0'' \cdot g^{K_4''}) / C' = e(g,g)^{\alpha s}.$$

### 4.2 Security Analysis

**Theorem 1.** *The FO-CP-AB-PRE-KEM scheme in Section 4.1 is selective CPA-secure at original ciphertext under the assumption that the CP-ABE scheme in [35] is selective CPA-secure.*

We postpone the proof to Appendix, available in the online supplemental material due to page limit.

**Theorem 2.** *The FO-CP-AB-PRE-KEM scheme in Section 4.1 is selective CPA-secure at re-encrypted ciphertext under the assumption that the CP-ABE scheme in [35] is selective CPA-secure.*

We postpone the proof to Appendix, available in the online supplemental material due to page limit.

## 5 Performance Evaluations

In this section, we give both theoretical and experimental analyses of the proposed fine-grained data sharing scheme.

### 5.1 Theoretical Analysis

*Computation Cost Comparison.* The computation cost of PKG, data owners and data consumers refers to the execution time of *KeyGen.pkg*, *KeyGen.ran*, *Enc.user*, *RKeyGen.user*, *Dec. user for CT*, and *Dec.user for RCT*. Table 2 compares the

TABLE 2
Computation Cost Comparison

| Schemes | KeyGen.pkg | Enc.user | RKeyGen.user | Dec.user for CT | Dec.user for RCT |
|---|---|---|---|---|---|
| [27] | $(1+2y)$Exp | $(3+l)$Exp | $(4+2y+l)$Exp | 2\|I\|P | 1Exp + $(2\|\mathrm{I}\|+1)$P |
| [34] | $(1+6y)$Exp | $(3+l)$Exp | $(4+2y+l)$Exp | 3\|I\|P | 1Exp + $(1+3\|\mathrm{I}\|)$P |
| [28] | $(3+4y)$Exp | 0Exp | 0Exp | $(2\|\mathrm{I}\|+1)$Exp + $(2+3\|\mathrm{I}\|)$P | $(2\|\mathrm{I}\|+1)$Exp + $(3+3\|\mathrm{I}\|)$P |
| [21] | 0 | 1Exp | × | 1Exp | × |
| [19] | 3Exp | $(3+l)$Exp + 1P | × | 1Exp | × |
| [31] | $(1+3y)$Exp | 4Exp | × | \|I\|Exp + 2\|I\|P | × |
| Ours | 0 | 0 | 0 | 1Exp | 2Exp + 1P |

‡ *Exp and P denote a modular exponentiation and a pairing computation, respectively.* $y$, $l$, *and* I *indicate the number of attributes, the access policy size, and the set that satisfies decryption requirement, respectively.*

number of modular exponentiations and pairing operations in our system with those in [19], [21], [27], [28], [31], [34].

Our scheme achieves outsourced key generation, encryption, re-encryption key generation and decryption simultaneously, only leaves few simple operations to the PKG, data owners and data consumers. In Table 2, the data owner and data consumer need almost no complex operation except 1 exponentiation in *Dec.user for CT*, 1 paring operation and 2 exponentiations in *Dec.user for RCT*. Though the schemes in [27], [34] achieve ABPRE, the computation cost is heavy. The scheme in [28] only optimizes the computation efficiency of encryption and re-encryption, but the mobile device still needs to do plenty of complex offline operations, thus it is a promising and practical solution for high-end mobile devices, but may not be economical for some scenarios. Although the schemes in [19], [21], [31] support outsourced ABE, they do not support authority delegation. Compared with above schemes, our scheme has a considerable advantage in efficiency. We remark that the compared schemes may find different applications.

*Communication Cost Analysis*. Nevertheless, outsourced computation certainly brings extra communication cost, e.g., a user needs to wait for the cloud server's response and downloads the computation results online. In the above process, the user's mobile device will experience the power consumption and the network latency. The power consumption always increases with the transmission size. The network latency can be influenced by many factors, but only the transmission size depends on the scheme. Next, we will compare the transmission size below.

Table 3 compares the transmission size of our scheme with those in [19], [21], [31]. For outsourced key generation and encryption, the transmission size of our scheme is larger than that in [19], [31], but our scheme still behaves better. Because the communication cost of [19], [31] is online, meaning the cloud servers should wait for the essential information (e.g., the plaintext, access policy and attributes) before outsourced computation. While in our system, the cloud servers can generate $IT$s/$ISK$s once they obtain $PK$ instead of waiting for essential information, and the downloading can be executed when the mobile device (PKG) is plugged into a power source (in the spare time). When the key generation/encryption/re-encryption key generation request arrives, the PKG/data owners can rapidly assemble private key/ciphertext/re-encryption key without waiting for the cloud server's response. Thus, the communication cost in our system is offline, which means it is "imperceptible" for the PKG and users. The communication cost of [21] is also offline, but [21] cannot achieve efficient authority delegation. For outsourced decryption in original ciphertexts (CT), all the above schemes utilize key blinding technique [18], thus the communication cost is small and fixed (always the size of an element in $\mathbb{G}_T$). For outsourced decryption in re-encrypted ciphertexts (RCT), the communication cost is also small and fixed (always the size of three elements in $\mathbb{G}$ and two elements in $\mathbb{G}_T$).

In the EMR system, the PKG downloads $ISK$s from two cloud servers (running *KeyGen.out* with $PK$) when it is in the spare time. When the key generation request arrives, the PKG can rapidly assemble and generate private key pair $(SK, TK, RtK)$ by running *KeyGen.pkg* and *KeyGen.ran*. When the doctor is in the spare time (off hours or lunch break), he/she plugs the mobile device into a power source, the device starts to download $IT$s, $ISK$s from two cloud servers (that run *Encrypt.out* and *KeyGen.out*) and stores them in the local storage. When the doctor wants to share/delegate (encrypt/re-encrypt) the medical record, he/she can rapidly perform encryption and re-encryption key generation on move without significantly draining the battery and waiting for the cloud server's response, then send $CT/(CT, RK)$ to public cloud 2. To decrypt encrypted medical records, a doctor requests public cloud 2 the target record, then DS

TABLE 3
Transmission Size Comparison

| Schemes | KeyGen | Encryption | Re-KeyGen | Decryption for CT | Decryption for RCT | Type |
|---|---|---|---|---|---|---|
| [19] | 2y$\|\mathbb{G}\|$ | × | × | 1$\|\mathbb{G}_T\|$ | × | online |
| [31] | × | 2yn$\|\mathbb{G}\|$ | × | × | × | online |
| [21] | $(4\mathrm{j}+4)(\|\mathbb{Z}_p\|+\|\mathbb{G}\|)$ | $(6\mathrm{l}+2)(\|\mathbb{Z}_p\|+\|\mathbb{G}\|)$ | × | 1$\|\mathbb{G}_T\|$ | × | offline |
| Ours | $(4\mathrm{j}+4)(\|\mathbb{Z}_p\|+\|\mathbb{G}\|)$ | $(6\mathrm{l}+2)\|\mathbb{Z}_p\|+(6\mathrm{l}+4)\|\mathbb{G}\|$ | $(4\mathrm{j}+6\mathrm{l}+6)\|\mathbb{Z}_p\|+(4\mathrm{j}+6\mathrm{l}+8)\|\mathbb{G}\|$ | 1$\|\mathbb{G}_T\|$ | 3$\|\mathbb{G}\|$ + 2$\|\mathbb{G}_T\|$ | offline |

‡ $|\mathbb{Z}_p|$, $|\mathbb{G}|$ *and* $|\mathbb{G}_T|$ *denote the size of an element in* $\mathbb{Z}_p$, *an element in* $\mathbb{G}$ *and* $\mathbb{G}_T$, *respectively.* $n$, $l$, $y$ *and* $j$ *indicate the number of cloud servers, the number of rows of the matrix* $M$ *for LSSS, the number of leaf nodes and the number of attributes, respectively.*

TABLE 4
Device Configuration

| Type | Configuration | Role | Algorithm |
|---|---|---|---|
| Alibaba Cloud | Intel Xeon E5-2682 v4@2.4 GHz, 2 GB RAM, 1 Mbps, Ubuntu 16.04 64-bit | public cloud 2 (KGS, ES, RES, SS, DS) | *KeyGen.out*, *Encrypt.out*, *ReEnc*, *Decrypt.out for CT/RCT* |
| Tencent Cloud | Intel Xeon E5-26XX v3@2.3 GHz, 1 GB RAM, 1 Mbps, Ubuntu 16.04 64-bit | public cloud 1 (KGS, ES) | *KeyGen.out*, *Encrypt.out* |
| Laptop (HASEE) | Intel Core i7-4710MQ@2.5 GHz, 8 GB RAM, Kali Linux 2.0 64-bit | PKG, data owner, data consumer | *KeyGen.pkg*, *KeyGen.ran*, *Encrypt.user*, *RKeyGen.user*, *Decrypt.user for CT/RCT* |
| Mobile phone (K-Touch 3) | Snapdragon MSM8926@1.2 GHz, 1 GB RAM, WIFI (IEEE 802.11 n/b/g), Android 4.3 | data owner, data consumer | *Encrypt.user*, *RKeyGen.user*, *Decrypt.user for CT/RCT* |

decrypts the record with $TK$ and sends $CT'$ to the doctor. At last, the doctor decrypts the record.

### 5.2 Experimental Analysis

To evaluate the practical performance, we develop an extensible library libabe, which offers essential APIs for implementing ABE schemes. To be compatible with Android OS, libabe is developed by C language and only dependent on Pairing-Based Cryptography (PBC) library [36] and OpenSSL [37]. Based on libabe, we develop the evaluation program with Java Native Interface (JNI). The curve that we choose is the 224-bit MNT elliptic curve from the PBC library.

We hire two public cloud service providers (Alibaba Cloud and Tencent Cloud) to execute the corresponding algorithms. We utilize a laptop to act as the PKG, and the same laptop and a low-end mobile phone play the part of users. The device configuration is presented in Table 4. To emphasize the economy of our scheme, we choose a low-end mobile phone whose price is about $66 (439CNY).

*Experiment Setting.* We set access policies for $CT$s and $RCT$s in the form of $(S_1 \wedge S_2 \wedge \ldots \wedge S_l)$ to simulate the worst situation. We set 20 distinct access policies with $l$ increasing from 10 to 100, repeat each instance 10 times and take the average value. The time is given in milliseconds.

*Computation Time.* As depicted in Fig. 6, we show *KeyGen.out Time*, *Encrypt.out Time*, *ReEnc Time*, *Decrypt.out for CT/RCT Time*, *KeyGen.pkg Time*, *KeyGen.ran Time*, *Encrypt.user Time*, *RKeyGen.user Time* and *Decrypt.user for CT/RCT Time* of our FO-CP-AB-PRE scheme.

For *KeyGen.out* and *Encrypt.out*, we only evaluate the time cost of one server, because two servers run in parallel. In Figs. 6a and 6e, *KeyGen.out Time* is about 0.12s∼1.18 s while *KeyGen.pkg* is only 0.4∼2.9 ms. In Figs. 6b and 6g, *Encrypt.out Time* is about 0.13∼1.26 s while *Encrypt.user Time* on the laptop is 3.3∼31.7 ms, *Encrypt.user Time* on the mobile phone is 30.1∼295.7 ms. In Fig. 6c, *ReEnc Time* is about 0.21∼2.11 s. In Fig. 6d, since the operations in *Decrypt.out for CT* and *Decrypt.out for RCT* are almost the same, the execution time is about 0.25∼2.25 s. We remark that if we choose higher configuration for the cloud server, the results will be improved significantly.

In Fig. 6i, for the laptop, *Decrypt.user for CT Time* is about 1.4 ms, *Decrypt.user for RCT Time* is about 7.1 ms. For the mobile phone, *Decrypt.user for CT Time* is about 24.1 ms,

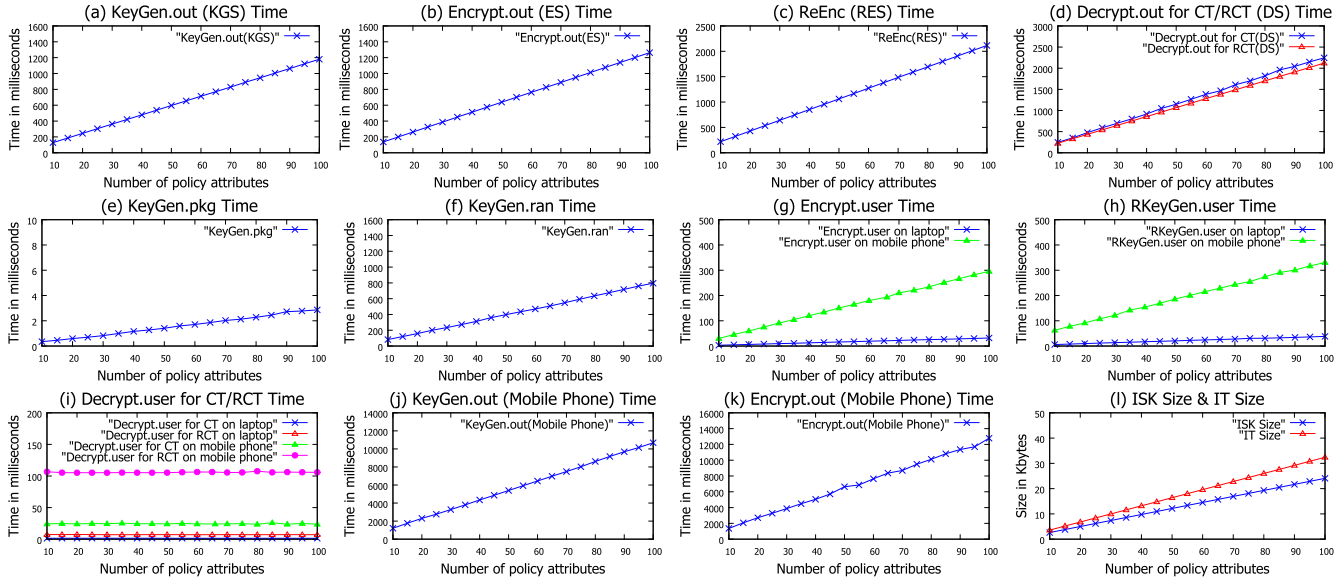



Fig. 6. Experimental results.

*Decrypt.user for RCT Time* is about 105.5 ms. *Decrypt.user for RCT Time* is more than *Decrypt.user for CT Time* because of the extra 1 paring operation and 1 exponentiation. In Fig. 6f, *KeyGen.ran Time* is about 82.4∼795.9 ms. *KeyGen.ran* will be called only when the user's private key pair needs to be generated or updated. In Fig. 6h, *RKeyGen.user Time on laptop* is about 6.1∼38.2 ms, *RKeyGen.user Time on mobile phone* is about 62.5∼330.7 ms.

Since the servers undertake the majority of heavy work, the PKG and users can rapidly complete all the operations. Hence, our technique significantly enhances the efficiency.

*Comparisons with OOABE/OOABPRE.* The primary difference between our scheme and online offline ABE [20]/online offline ABPRE [28] is that the preparation work in [20], [28] is performed by the mobile phone, while that in our scheme is outsourced to public cloud. We conjecture that the scheme in [20], [28] may not be suitable for low-end mobile devices, because the mobile device may take long time to do massive complex computation during the offline phase. To prove our conjecture, we present the following comparative experiment.

In OOABE/OOABPRE, *KeyGen.out Time on mobile phone* is about 1.2∼10.7 s, and *Encrypt.out Time on mobile phone* is about 1.4∼12.8 s. In our scheme, the extra communication cost mainly depends on the transmission size, so we present $ISK$ size and $IT$ size in Fig. 6l. $ISK$ size is about 2.6∼24.1 KB, and $IT$ size is about 3.6∼32.4 KB.

In an EMR system, we assume that a doctor separately encrypts and re-encrypts the medical records 500 times everyday. For the access policy in the form of $(S_1 \wedge S_2 \wedge \ldots \wedge S_{100})$, the mobile phone in OOABE needs about 5 hours of spare time ((10.7 + 12.8 + 12.8)*500/3600 $\approx$ 5 hours) to generate $IT$s and $ISK$s. For the doctors who deal with emergencies, sometimes they may not have 5 hours of spare time. While in our scheme, the mobile phone needs to download 43 MB of $IT$s and $ISK$s from one cloud server ((24.1 + 32.4 + 32.4)*500/1024 $\approx$ 43 MB). We use the mobile phone to evaluate the transmission time from Alibaba Cloud.[6] The mobile phone connects to Internet via Wi-Fi and the network speed is 8 Mbps. The mobile phone spends 374 seconds (about 6 minutes) of spare time to download 43 MB of $IT$s and $ISK$s from Alibaba Cloud via File Transfer Protocol (FTP) (note that the bandwidth of Alibaba Cloud is 1 Mbps). Since the mobile phone has to complete the download from Tencent Cloud, the total time is about 12 minutes. The more money we pay to improve the cloud server bandwidth, the more download time we will save. We remark that OOABE and OOABPRE are promising solutions for high-end mobile devices, and our scheme is an economical solution for low-end mobile devices.

## 6 Conclusion and Future Work

In this work, we propose a fine-grained data sharing mechanism for the EMR system, which not only achieves data privacy, non-interactive fine-grained access control, and authority delegation simultaneously, but also is suitable for low-end mobile devices. Moreover, we develop an extensible library called libabe that is compatible with Android devices, and we implement our mechanism on realistic environment. The experimental results indicate that our scheme is efficient, practical and economical. In the future, we will focus on designing the efficient revocation mechanism.

6. The server's computation time is ignored, because it has superior computation capability and can do the computation once receiving $PK$.

## Acknowledgments

This work was supported in part by National Natural Science Foundation of China (Nos. 61632020, 61472416, 61772520, 61802392), Key Research Project of Zhejiang Province (No. 2017C01062), Fundamental theory and cutting edge technology Research Program of Institute of Information Engineering, CAS (No. Y7Z0321102), Australian Research Council Discovery Early Career Researcher Award (Grant No. DE150101116). The corresponding author is Rui Zhang.

## References

[1] D. F. Ferraiolo, R. S. Sandhu, S. I. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Trans. Inf. Syst. Security*, vol. 4, no. 3, pp. 224–274, 2001.

[2] E. Bertino, P. A. Bonatti, and E. Ferrari, "TRBAC: A temporal role-based access control model," *ACM Trans. Inf. Syst. Security*, vol. 4, no. 3, pp. 191–233, 2001.

[3] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor, "A generalized temporal role-based access control model," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 1, pp. 4–23, Jan. 2005.

[4] E. Bertino, B. Catania, M. L. Damiani, and P. Perlasca, "GEO-RBAC: A spatially aware RBAC," in *Proc. 10th ACM Symp. Access Control Models Technol.*, 2005, pp. 29–37.

[5] S. G. Akl and P. D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," *ACM Trans. Comput. Syst.*, vol. 1, no. 3, pp. 239–248, 1983.

[6] J. Alderman, N. Farley, and J. Crampton, "Tree-based cryptographic access control," in *Proc. 22nd Eur. Symp. Res. Comput. Security*, 2017, pp. 47–64.

[7] A. Castiglione, A. D. Santis, B. Masucci, F. Palmieri, A. Castiglione, and X. Huang, "Cryptographic hierarchical access control for dynamic structures," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 10, pp. 2349–2364, Oct. 2016.

[8] A. Castiglione, A. D. Santis, and B. Masucci, "Key indistinguishability versus strong key indistinguishability for hierarchical key assignment schemes," *IEEE Trans. Depend. Sec. Comput.*, vol. 13, no. 4, pp. 451–460, Jul./Aug. 2016.

[9] J. Alderman, J. Crampton, and N. Farley, "A framework for the cryptographic enforcement of information flow policies," in *Proc. 22nd ACM Symp. Access Control Models Technol.*, 2017, pp. 143–154.

[10] A. Castiglione, A. D. Santis, B. Masucci, F. Palmieri, A. Castiglione, J. Li, and X. Huang, "Hierarchical and shared access control," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 850–865, Apr. 2016.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Security*, 2006, pp. 89–98.

[12] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 321–334.

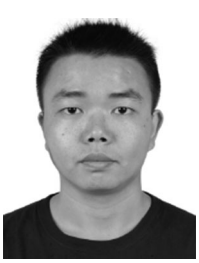
[13] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2013.

[14] J. Zhou, Z. Cao, X. Dong, and X. Lin, "TR-MABE: White-box traceable and revocable multi-authority attribute-based encryption and its applications to multi-level privacy-preserving e-healthcare cloud computing systems," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2398–2406.

[15] I. E. Ghoubach, F. Mrabti, and R. B. Abbou, "Efficient secure and privacy preserving data access control scheme for multi-authority personal health record systems in cloud computing," in *Proc. Int. Conf. Wireless Netw. Mobile Commun.*, 2016, pp. 174–179.

[16] J. Alderman, C. Janson, C. Cid, and J. Crampton, "Access control in publicly verifiable outsourced computation," in *Proc. 10th ACM Symp. Inf. Comput. Commun. Security*, 2015, pp. 657–662.

[17] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. 24th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2005, pp. 457–473.
[18] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. 20th USENIX Security Symp.*, 2011, pp. 34–34.
[19] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2201–2210, Aug. 2014.
[20] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Proc. 17th Int. Conf. Practice Theory Public-Key Cryptography*, 2014, pp. 293–310.
[21] R. Zhang, H. Ma, and Y. Lu, "Fine-grained access control system based on fully outsourced attribute-based encryption," *J. Syst. Softw.*, vol. 125, pp. 344–353, 2017.
[22] H. Ma, R. Zhang, Z. Wan, Y. Lu, and S. Lin, "Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing," *IEEE Trans. Depend. Sec. Comput.*, vol. 14, no. 6, pp. 679–692, Nov./Dec. 2017.
[23] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. Adv. Cryptology*, 1998, pp. 127–144.
[24] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Inf. Syst. Security*, vol. 9, no. 1, pp. 1–30, 2006.
[25] G. Hanaoka, Y. Kawai, N. Kunihiro, T. Matsuda, J. Weng, R. Zhang, and Y. Zhao, "Generic construction of chosen ciphertext secure proxy re-encryption," in *Proc. 12th Conf. Topics Cryptology*, 2012, pp. 349–364.
[26] G. Ateniese, K. Benson, and S. Hohenberger, "Key-private proxy re-encryption," in *Proc. Cryptographers' Track RSA Conf. Topics Cryptology*, 2009, pp. 279–294.
[27] X. Liang, Z. Cao, H. Lin, and J. Shao, "Attribute based proxy re-encryption with delegating capabilities," in *Proc. ACM Symp. Inf. Comput. Commun. Security*, 2009, pp. 276–286.
[28] J. Shao, R. Lu, and X. Lin, "Fine-grained data sharing in cloud computing for mobile devices," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2677–2685.
[29] AmosBeimel, "Secure schemes for secret sharing and key distribution," Ph.D. dissertation, Technion-Israel Inst. Technol., Faculty Comput. Sci., Haifa, Israel, 1996.
[30] V. Shoup, "A proposal for an ISO standard for public key encryption," *IACR Cryptology ePrint Archive*, vol. 2001, 2001, Art. no. 112.
[31] J. Li, C. Jia, J. Li, and X. Chen, "Outsourcing encryption of attribute-based encryption with MapReduce," in *Proc. 14th Int. Conf. Inf. Commun. Security*, 2012, pp. 191–201.
[32] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in *Proc. 2nd Int. Conf. Theory Cryptography*, 2005, pp. 264–282.
[33] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," in *Proc. 17th Eur. Symp. Res. Comput. Security*, 2012, pp. 541–556.
[34] S. Luo, J. Hu, and Z. Chen, "Ciphertext policy attribute-based proxy re-encryption," in *Proc. 12th Int. Conf. Inf. Commun. Security*, 2010, pp. 401–415.
[35] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2013, pp. 463–474.
[36] B. Lynn, "The stanford pairing based crypto library," [Online]. Available: http://crypto.stanford.edu/pbc
[37] "Openssl cryptography and ssl/tls toolkit," [Online]. Available: http://www.openssl.org/



**Hui Ma** received the BE degree in information security from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2012, and the PhD degree in information security from the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, in 2017. Now he is with the Institute of Information Engineering, Chinese Academy of Sciences as an assistant professor. He is working on applied cryptography and the security mechanisms in cloud computing.

**Rui Zhang** received the BE degree from Tsinghua University, and the MS/PhD degrees from the University of Tokyo, respectively. He was a JSPS research fellow before he joined AIST, Japan as a research scientist. Now he is with the Institute of Information Engineering (IIE), Chinese Academy of Sciences as a research professor. His research interests include applied cryptography, network security and information theory.

**Guomin Yang** (M'13) received the PhD degree in computer science from the City University of Hong Kong, Hong Kong, in 2009. He is currently a senior lecturer and a DECRA fellow with the School of Computing and Information Technology, University of Wollongong, Wollongong, NSW, Australia. His current research interests include applied cryptography and network security. He is a member of the IEEE.

**Zishuai Song** received the BE degree in information security from Xidian University, China, in 2017. He is currently working toward the PhD degree in information security with the Institute of Information Engineering, Chinese Academy of Sciences. He is currently involved in the security mechanisms in cloud computing.

**Kai He** received the BE degree in information security from the Huazhong University of Science and Technology, Wuhan, China, in 2014. He is currently working toward the ME degree in information security from the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. He is currently involved in the instantiation of public-key encryption schemes.

**Yuting Xiao** received the BE degree in network engineering from the Sichuan University of College of Computer Science, Chengdu, China, in 2015. She is currently working toward the master's degree in information security with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. Her research interests include cryptography and information security.