Hindawi
Security and Communication Networks
Volume 2020, Article ID 8347213, 20 pages
https://doi.org/10.1155/2020/8347213

Research Article

Secure Outsourced Medical Data against Unexpected Leakage with Flexible Access Control in a Cloud Storage System

Xingguang Zhou,[1,2] Jianwei Liu,[1] Zongyang Zhang ,[1] and Qianhong Wu[1]

[1]*School of Cyber Science and Technology, Beihang University, Beijing, China*
[2]*Civil Aviation Management Institute of China, Beijing, China*

Correspondence should be addressed to Zongyang Zhang; zongyangzhang@buaa.edu.cn

Received 30 August 2019; Revised 1 December 2019; Accepted 8 January 2020; Published 10 February 2020

Guest Editor: Kewei Sha

The application of cloud storage system has been deployed widely in recent years. A lot of electronic medical records (EMRs) are collected and uploaded to the cloud for scalable sharing among the authority users. It is necessary to guarantee the confidentiality of EMRs and the privacy of EMR owners. To achieve this target, we summarize a series of attack behaviors in the cloud storage system and present the security model against many types of unexpected privacy leakage. Privacy of unassailed EMRs is guaranteed in this model, and the influence of privacy leakage is controlled in a certain scope. We also propose a role-based access control scheme to achieve flexible access control on these private EMRs. One can access medical records only if his/her role satisfies the defined access policy, which implies a fine-grained access control. Theoretical and experimental analyses show the efficiency of our scheme in terms of computation and communication.

1. Introduction

Cloud communication has been envisioned as one of the most influential technologies in the medical field. Without being measured face-to-face, medical staff could give guidance to patients in a real-time way, which greatly improves the healthcare quality. For instance, a patient with heart disease history can deploy a medical sensor at home for the purpose of health monitoring. His health data are uploaded to cloud server and used by remote hospitals. The doctors in their duties could download his EMR and prepare the patient for treatment if needed. This method brings convenience for both patients and hospitals. However, once a patient's data are uploaded to the cloud storage system, they lose the physical control over the data and the cloud provider can obtain the access on it. Privacy threats experienced by users of Google Inc., Apple Inc., and Amazon Inc. [1] clearly indicate that cloud is intrinsically insecure from the users' point of view [2]. Most users would like to keep their personal information confidential to outsiders, let alone those patients whose EMRs include a lot of sensitive information. Data confidentiality is one of the important security concerns in the cloud storage system.

A common solution for data confidentiality is to encrypt them using public key encryption [3] before transmitting to the cloud system. We first generate two cryptographic keys: one is public and the other is secret. The public key is distributed by the data owner who is responsible for encryption. The secret key is private and assigned to each recipient in duty for decryption, such as the medical staff who is responsible for a patient.

While public key encryption ensures data confidentiality commendably, we admit that no matter what measures we take, unexpected privacy leakage sometimes happens. There are mainly three potential leakage risks in a typical cloud storage system: vulnerable medical devices, a semitrust cloud server, and the association among EMRs themselves. In the first type of risk, the data collected by medical devices will be firstly handled in a local but relatively open place (such as in a ward or on an ambulance), then it is uploaded to the cloud. The data are easily accessible for unauthorized persons in an emergency. In the second type of risk, since the cloud provider is a semitrusted organization, it honestly follows the rules but does everything possible to spy on the stored files. Patients' information might be leaked by internal staff of the cloud provider. The third type of risk is due to EMRs' internal association for a patient and his family, i.e., a father's heart attack record may reflect a similar heart disease of his son. Leaking one record might infer unassailed ones. Besides the above three potential risks, we also need to consider that there are many malicious adversaries who keep trying to gain access to the cloud storage system, i.e., the communication among the devices (such as body sensors), the cloud and the EMR recipients. There are lots of ways to do this [4]. One way is to break the random-number generator (RNG) [5] and thus gain the randomness used for encryption. Another way is to break into the cloud part. For instance, Albrecht and Paterson [6] introduced a powerful attack that an adversary runs malicious JavaScript in a targeted browser and completely recovers HTTP session cookies or user credentials such as passwords.

All these unexpected events might cause parts of database corrupted and data exposed. The cloud storage system should be resilient in the case of security breach. In other words, once the privacy leakage happens, the leakage effect should be controllable, so that the confidentiality of the "unleaked" information is guaranteed. Therefore, controllability of privacy leakage is another important security concern in the cloud storage system.

Furthermore, we need to consider that the cloud storage system is usually associated with a multiparty communication environment, as Figure 1 describes. The health data are scalably shared among authorized users. From the recipients' point of view, the access control manner needs to be flexible enough to deal with the changes of users' roles and permissions [7].

1.1. Our Contributions. Based on the aforementioned security concerns in the cloud storage system, we propose a leakage controllable scheme to achieve data confidentiality with a flexible access manner. Since our scheme is based on a multiuser access policy, it quite matches the real world's scenario which includes many users in a hierarchical organization. Specific techniques are highlighted as follows.

Privacy Leakage Controllability. As unexpected privacy leakage always happens in various forms, we propose new security models, called role-based access control against unexpected leakage (RBAC-UL) to capture further leakage on the remaining "unleaked" EMRs. RBAC-UL security is achieved if confidentiality of unassailed EMRs is still guaranteed.

Flexible Access Control. We offer an efficient approach to support fine-grained access control for a hierarchical healthcare organization. A user can comprehend an EMR only if his identity satisfies the associated access policy.

Scalable Data Sharing. It is achieved by letting higher-level medical staff delegate access privilege for his subordinates.

Constant Size Ciphertext. Our scheme achieves constant size of encapsulated EMR no matter how many users satisfy the defined access policy.

Rigorous Security Analysis. To ensure that our proposal is qualified enough for the series of security concerns, we present rigorous security analysis. The analysis shows that our proposal achieves a high privacy-preserving capacity, where data confidentiality, leakage controllability, and access control flexibility can be achieved simultaneously in the cloud storage system.

2. Related Work

Privacy-preserving access control in the cloud storage system has received more and more attention recently. Cryptography and authentication methods are utilized in the cloud network to offer secure healthcare services via wireless communications [8]. For the security of EMR, encryption is an efficient and cost-saving choice to guarantee patients' privacy. A lot of prominent schemes have been proposed to achieve this target. The scheme applying identity-based encryption (IBE) [9, 10] presents efficient solutions for the body sensor network. While considering fine-grained sharing of encrypted data, attribute-based encryption (ABE) [11, 12] is promising. This is because ABE provides differential access privileges for a set of users such as healthcare providers and allows flexibility in designating the access privileges of individual users over the encrypted data [13]. An immediate attribute modification method is used to achieve fine-grained user revocation and the outsourced e-health records security [14]. The searchable ABE scheme is a promising technique that can ensure the protection of patients' private information without compromising on performance [15]. When applying ABE schemes in the medical systems, it shows the security and flexibility as the user tries to access the outsourced EMRs. Besides, a role-based access control framework [16, 17] is proposed by using hierarchical identity-based broadcast encryption (HIBBE) [18] without ABE, which ensures the security, scalability, and flexibility for the outsourced EMRs. A secure role-based cloud storage system for encrypted patient-centric health records is achieved in commercial healthcare systems [19]. An auditing revocable privacy-preserving access control scheme for the e-health records shows the efficiency of RBAC in terms of communication and computation [20]. An enhancing medical data security scheme in the cloud using RBAC provides security to the data over an alien environment [21].

Although the aforementioned schemes devote to securing the outsourced EMRs, they are unable to deal with the situation of unexpected privacy leakage, let alone to minimize its effect. In a cloud storage system, the leakage threats mainly include secret credential leakage [22, 23], encapsulation-related randomness leakage [24, 25], internal files, accounts or other records leakage, etc. The target of our paper is to minimize the impact of leakage in the event that these unexpected issues happen. We notice that a lot of schemes have been put forward theoretically against these unexpected leakages, including the public-key encryption

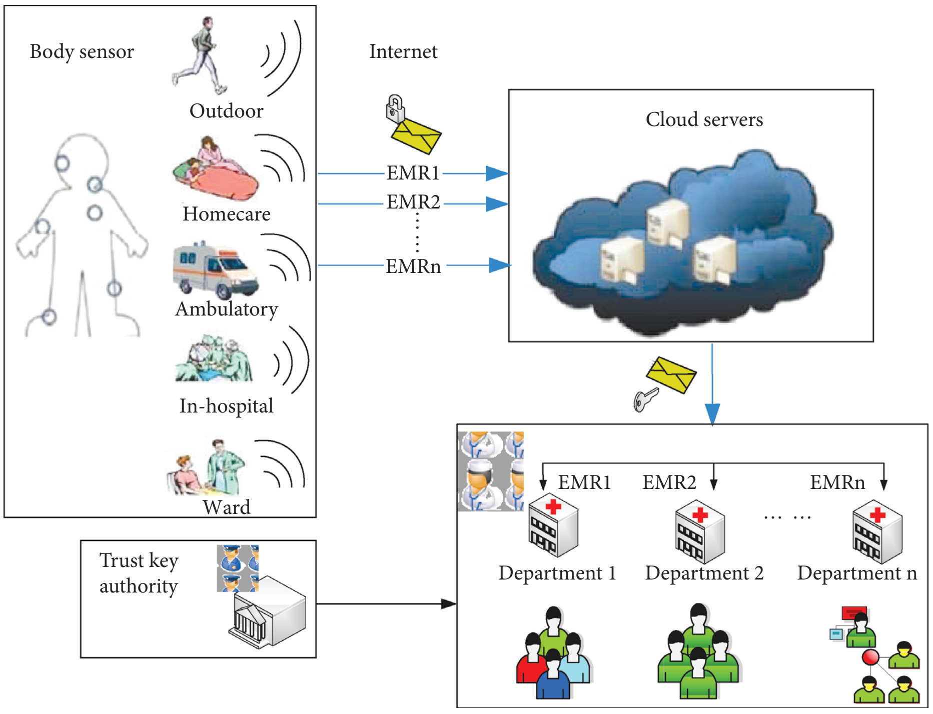


Figure 1: A medical cloud communication environment.

[26–28] and the identity-based encryption schemes [29–31]. They are different from our RBAC-UL mechanism. The former mainly guarantee the confidentiality of the remaining "unleaked" records, while ours not only ensures the confidentiality of "unleaked" data, but also achieves scalable sharing and flexible access control of all the outsourced EMRs.

3. Preliminaries

This section provides some mathematical basis for our proposal. The notations that are used in our scheme are given in Table 1. For ease of description, some of them are borrowed from [16, 17].

Table 1: Notations.

Notation	Description
λ	Security parameter
R	Atom role for medical staff
$\overrightarrow{R}$	Role for medical staff
$S_{\overrightarrow{R}}$	Atom role set for $\overrightarrow{R}$
P	Access policy
S_P	Atom role set for P
MSK	Master secret key
EMR	Electronic medical record
PPT	Probabilistic polynomial time
$\text{SC}_{\overrightarrow{R}}$	Secret credential for a role $\overrightarrow{R}$
En	The encapsulated EMR
$\text{Pref}(\overrightarrow{R})$	Prefix of $\overrightarrow{R}$, defined as $\{(R_1, \ldots, R_{d'}) : d' \le d\}$
$\text{Pref}(P)$	Prefix of P, defined as $\cup_{\overrightarrow{R} \in P} \text{Pref}(\overrightarrow{R})$

3.1. Bilinear Groups. Let $\mathscr{G}$ be a group generation algorithm that takes a security parameter λ as input and outputs the description of a bilinear group $(N, \mathbb{G}, \mathbb{G}_T, e)$. In our case, $\mathscr{G}$ outputs $(N = p_1p_2p_3p_4, G, G_T, e)$ where p_1, p_2, p_3, p_4 are distinct prime factors, $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order $N = p_1p_2p_3p_4$, and $e: \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ is an efficient bilinear map satisfying the two properties: (i) bilinearity: For all $g, h \in \mathbb{G}$ and all $a, b \in \mathbb{Z}_N$, $e(g^a, h^b) = e(g, h)^{ab}$; (ii) non-degeneracy: there exists at least a generator g in $\mathbb{G}$ such that $e(g, g)$ generates $\mathbb{G}_T$. We, respectively, denote the subgroups of order p_1, p_2, p_3, p_4 in $\mathbb{G}$ by $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}$, and $\mathbb{G}_{p_4}$. We use $G_{p_ip_j}$ $(1 \le i, j \le 4)$ to denote the subgroup of order p_ip_j in $\mathbb{G}$. These four subgroups additionally satisfy the orthogonality property, i.e., $\forall h_i \in \mathbb{G}_{p_i}$ and $h_j \in \mathbb{G}_{p_j}$ for $i \ne j$, $e(h_i, h_j) = 1$. This orthogonality property will be a principal tool in our constructions. Composite-order bilinear groups were first introduced in [32].

3.2. Theoretical Assumptions. Our security analysis is based on the following mathematical assumptions.

Assumption 1. Given a group generator $\mathscr{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1p_2p_3p_4, G, G_T, e) \xleftarrow{R} \mathscr{G}, g_1 \xleftarrow{R} G_{p_1}, g_3 \xleftarrow{R} G_{p_3}, g_4 \xleftarrow{R} G_{p_4},$$
$$D = (\mathbb{G}, g_1, g_3, g_4). \tag{1}$$

Assumption 1 determines whether a given element T is randomly chosen from G or from $G_{p_1p_3p_4}$, namely, $T \xleftarrow{R} G$ or $T \xleftarrow{R} G_{p_1p_3p_4}$. The advantage of an algorithm $\mathscr{A}$ that outputs a bit $b \in \{0, 1\}$ in breaking Assumption 1 is defined as

$$\begin{aligned}\mathrm{Adv1}_A(\lambda) = \Big|&\Pr\big[A(D, T\overset{R}{\longleftarrow}G) = 1\big]\\ &- \Pr\big[A\big(D, T\overset{R}{\longleftarrow}G_{p_1p_3p_4}\big) = 1\big]\Big| - \frac{1}{2}.\end{aligned} \tag{2}$$

Definition 1. $\mathcal{G}$ satisfies Assumption 1 if $\mathrm{Adv1}_{\mathscr{A}}(\lambda)$ is a negligible function for any PPT algorithm $\mathscr{A}$.

Assumption 2. Given a group generator $\mathcal{G}$, we define the following distribution:

$$\begin{aligned}&\mathbb{G} = (N = p_1p_2p_3p_4, G, G_T, e)\overset{R}{\longleftarrow}\mathcal{G}, g_1\overset{R}{\longleftarrow}G_{p_1},\\ &g_3\overset{R}{\longleftarrow}G_{p_3}, g_4\overset{R}{\longleftarrow}G_{p_4},\\ &D = (\mathbb{G}, g_1, g_3, g_4).\end{aligned} \tag{3}$$

Assumption 2 determines whether a given element is $T\overset{R}{\longleftarrow}G_{p_1p_2p_4}$ or $T\overset{R}{\longleftarrow}G_{p_1p_4}$. The advantage of an algorithm $\mathscr{A}$ that outputs $b \in \{0, 1\}$ in breaking the Assumption 2 is defined as

$$\begin{aligned}\mathrm{Adv2}_{\mathscr{A}}(\lambda) = \Big|&\Pr\big[\mathscr{A}\big(D, T\overset{R}{\longleftarrow}G_{p_1p_2p_4}\big) = 1\big]\\ &- \Pr\big[\mathscr{A}\big(D, T\overset{R}{\longleftarrow}G_{p_1p_4}\big) = 1\big]\Big| - \frac{1}{2}.\end{aligned} \tag{4}$$

Definition 2. $\mathcal{G}$ satisfies the Assumption 2 if $\mathrm{Adv2}_{\mathscr{A}}(\lambda)$ is a negligible function for any PPT algorithm $\mathscr{A}$.

Assumption 3. Given a group generator $\mathcal{G}$, we define the following distribution:

$$\begin{aligned}&\mathbb{G} = (N = p_1p_2p_3p_4, G, G_T, e)\overset{R}{\longleftarrow}\mathcal{G}, g_1\overset{R}{\longleftarrow}G_{p_1}, g_3\overset{R}{\longleftarrow}G_{p_3},\\ &g_4\overset{R}{\longleftarrow}G_{p_4}, D_{23}\overset{R}{\longleftarrow}G_{p_2p_3}, A_{12}\overset{R}{\longleftarrow}G_{p_1p_2},\\ &D = (\mathbb{G}, g_1, g_3, g_4, D_{23}, A_{12}).\end{aligned} \tag{5}$$

Assumption 3 determines whether a given element is $T\overset{R}{\longleftarrow}G_{p_1p_2p_3}$ or $T\overset{R}{\longleftarrow}G_{p_1p_3}$. The advantage of an algorithm $\mathscr{A}$ that outputs $b \in \{0, 1\}$ in breaking Assumption 3 is defined as

$$\begin{aligned}\mathrm{Adv3}_{\mathscr{A}}(\lambda) = \Big|&\Pr\big[\mathscr{A}\big(D, T\overset{R}{\longleftarrow}G_{p_1p_2p_3}\big) = 1\big]\\ &- \Pr\big[\mathscr{A}\big(D, T\overset{R}{\longleftarrow}G_{p_1p_3}\big) = 1\big]\Big| - \frac{1}{2}.\end{aligned} \tag{6}$$

Definition 3. $\mathcal{G}$ satisfies Assumption 3 if $\mathrm{Adv3}_{\mathscr{A}}(\lambda)$ is a negligible function for any PPT algorithm $\mathscr{A}$.

Assumption 4. Given a group generator $\mathcal{G}$, we define the following distribution:

$$\begin{aligned}&\mathbb{G} = (N = p_1p_2p_3p_4, G, G_T, e)\overset{R}{\longleftarrow}\mathcal{G}, g_2\overset{R}{\longleftarrow}G_{p_2}, g_3\overset{R}{\longleftarrow}G_{p_3},\\ &g_4\overset{R}{\longleftarrow}G_{p_4}, W_{14}\overset{R}{\longleftarrow}G_{p_1p_4}, E_{12}\overset{R}{\longleftarrow}G_{p_1p_2},\\ &D = (\mathbb{G}, g_2, g_3, g_4, W_{14}, E_{12}).\end{aligned} \tag{7}$$

Assumption 4 determines whether a given element is $T\overset{R}{\longleftarrow}G_{p_2p_4}$ or $T\overset{R}{\longleftarrow}G_{p_1p_2p_4}$. The advantage of an algorithm $\mathscr{A}$ that outputs $b \in \{0, 1\}$ in breaking Assumption 4 is defined as

$$\begin{aligned}\mathrm{Adv4}_{\mathscr{A}}(\lambda) = \Big|&\Pr\big[\mathscr{A}\big(D, T\overset{R}{\longleftarrow}G_{p_2p_4}\big) = 1\big]\\ &- \Pr\big[\mathscr{A}\big(D, T\overset{R}{\longleftarrow}G_{p_1p_2p_4}\big) = 1\big]\Big| - \frac{1}{2}.\end{aligned} \tag{8}$$

Definition 4. $\mathcal{G}$ satisfies Assumption 4 if $\mathrm{Adv4}_{\mathscr{A}}(\lambda)$ is a negligible function for any PPT algorithm $\mathscr{A}$.

Assumption 5. Given a group generator $\mathcal{G}$, we define the following distribution:

$$\begin{aligned}&\mathbb{G} = (N = p_1p_2p_3p_4, G, G_T, e)\overset{R}{\longleftarrow}\mathcal{G}, g_1\overset{R}{\longleftarrow}G_{p_1},\\ &g_4\overset{R}{\longleftarrow}G_{p_4}, D_{23}\overset{R}{\longleftarrow}G_{p_2p_3},\\ &D = (\mathbb{G}, g_1, g_4, D_{23}).\end{aligned} \tag{9}$$

Assumption 5 determines whether a given element is $T\overset{R}{\longleftarrow}G$ or $T\overset{R}{\longleftarrow}G_{p_1p_2p_4}$. The advantage of an algorithm $\mathscr{A}$ that outputs $b \in \{0, 1\}$ in breaking Assumption 5 is defined as

$$\begin{aligned}\mathrm{Adv5}_{\mathscr{A}}(\lambda) = \Big|&\Pr[\mathscr{A}(D, T\overset{R}{\longleftarrow}G) = 1]\\ &- \Pr\big[\mathscr{A}\big(D, T\overset{R}{\longleftarrow}G_{p_1p_2p_4}\big) = 1\big]\Big| - \frac{1}{2}.\end{aligned} \tag{10}$$

Definition 5. $\mathcal{G}$ satisfies Assumption 5 if $\mathrm{Adv5}_{\mathscr{A}}(\lambda)$ is a negligible function for any PPT algorithm $\mathscr{A}$.

4. System Model

In Figure 1, we describe a typical medical cloud storage system. It is a multiuser setting environment consisting of four entities: an EMR owner, a cloud server, an EMR recipient, and a trusted key authority (TKA).

The EMR owner is usually a patient who is monitored by different types of medical sensors. His/her medical records are sent via wireless networks. Each record is encrypted and associated with its own access policy and then stored on the cloud server for sharing with the entitled medical staff.

The cloud provides a large number of servers for many organizations. It is honest but curious, i.e., it obeys rules of the cloud system, but could do everything possible to spy on the stored EMRs.

The EMR recipient consists of groups of medical staff who are entitled to read the patients' EMRs and provide services for them. The medical staff are semitrusted. If they are authorized, they do not reveal any information. Otherwise, they might be potential adversaries. The staff at the higher-level is responsible for managing lower-level ones, which derives a tree-like organization. For example, the role of a nurse consisting of ordered atom roles "department psychiatry, chief doctor, head nurse, nurse," is administrated by the head nurse whose role is "department psychiatry, chief doctor, head nurse." The head nurse is administrated by the chief doctor and so on. We group the chief doctor, the head nurse, and the nurse in one access policy, where all of them are responsible for a certain patient. The patient is identified by his name or identity information. Each medical staff can encapsulate the patient's EMR, but only the one whose role satisfies the corresponding access policy can decapsulate it.

The TKA is responsible for generating and distributing system parameters.

A role-based access control scheme consists of the following algorithms:

$(\mathrm{PK}, \mathrm{MSK}) \longleftarrow \mathrm{Setup}(\lambda, n)$: the setup algorithm is run by TKA. It takes as inputs a security parameter λ and a maximal size n of users. It outputs a masker key MSK and a public key PK.

$\mathrm{SC}^{\overrightarrow{R}} \longleftarrow \mathrm{SCGen}(\mathrm{PK}, \mathrm{MSK}, \overrightarrow{R})$: the secret credential generation algorithm of medical staff takes as inputs a public key PK, a master key MSK, and a role $\overrightarrow{R}$ for a medical staff. It outputs a secret credential $\mathrm{SC}^{\overrightarrow{R}}$ for the medical staff with role $\overrightarrow{R}$.

$\mathrm{SC}^{\overrightarrow{R}} \longleftarrow \mathrm{SCDeleg}(\mathrm{PK}, \mathrm{SC}^{\overrightarrow{R'}}, R)$: the secret credential delegation algorithm of medical staff takes as inputs a public key PK, a secret credential $\mathrm{SC}^{\overrightarrow{R'}}$ for a medical staff with role $\overrightarrow{R'}$, and an atom role R. It returns the secret credential $\mathrm{SC}^{\overrightarrow{R}}$ for the medical staff with role $\overrightarrow{R} = (\overrightarrow{R'}, R)$. The medical staff with role $\overrightarrow{R'}$ is the supervisor of the one with role $\overrightarrow{R}$.

$\mathrm{En} \longleftarrow \mathrm{EMREnc}(\mathrm{PK}, P, \mathrm{EMR})$: the EMR encapsulation algorithm takes as inputs a public key PK, an access policy P, and an EMR file EMR. It outputs an encapsulated EMR file En.

$\mathrm{EMR} \longleftarrow \mathrm{EMRDec}(\mathrm{PK}, \overrightarrow{R}, \mathrm{En}, \mathrm{SC}^{\overrightarrow{R}})$: the EMR decapsulation algorithm takes as inputs a public key PK, a medical staff's role R, an encapsulated EMR En, and a secret credential $\mathrm{SC}^{\overrightarrow{R}}$ for the medical staff with role $\overrightarrow{R}$. It outputs an EMR file EMR.

5. Security Requirements

In practice, all entities except TKA are likely to attack the cloud storage system. A dishonest party may try to get useful information from the encapsulated EMRs, which is not authorized to access or derive from the leaked EMRs. In the context of such attack, our scheme is expected to meet the following security requirements.

(i) Data privacy: EMRs need to be obfuscated before being uploaded and securely stored on cloud servers, until an authorized user downloads and deobfuscates them.

(ii) Leakage controllability: when a privacy leakage happens unavoidably, it must be possible to minimize the leakage effect, which means the privacy of nonleaked EMRs should be guaranteed.

5.1. The Adversary Model for RBAC-UL.

The adversary model for RBAC-UL is aimed to satisfy the requirement of *leakage controllability*. Since the content of EMRs may be internally related, partial leakage might expose information on those EMRs. Therefore, it is necessary to clarify what it means for the unleaked EMRs to remain confidential. In the RBAC-UL model, we assume two roles: an adversary and a simulator. The adversary's goal is to collect as much information from the unleaked EMRs as possible, i.e., the corrupted EMRs, the encapsulated EMRs, and the randomness used for encapsulation. The simulator acts like a normal person with neutral characters: he can get the same input as the adversary when a leakage happens, and he has the ability to corrupt the EMR owners to learn their EMRs. Apart from that, the simulator cannot get any further information. We claim that if any adversary cannot obtain more information from the unleaked EMRs than the simulator, then the security of the remaining EMRs is guaranteed.

We formally define the adversary model for RBAC-UL, which is inspired by the work [29]. In the model, the corrputed EMR is encapsulated with a target access policy set P^* containing all the medical staff who are allowed to decapsulate. The adversary is allowed to do the following: (a) it can obtain secret credentials associated with roles $R \notin \mathrm{Pref}(P^*)$, which implies the adversary can collude any medical staff with roles that do not satisfy the target access policy; (b) it can obtain all the targeted encapsulated EMRs, $\overrightarrow{\mathrm{En}} = \{\mathrm{En}_1, \mathrm{En}_2, \ldots, \mathrm{En}_n\}$; (c) it can randomly corrupt any encapsulated EMRs from $\overrightarrow{\mathrm{En}}$ and then obtain their files $\{\mathrm{EMR}_i\}_{i\in[1,n]}$ with the randomness used for encapsulation $\{r_i\}_{i\in[1,n]}$, which implies that the adversary has the ability to corrupt the EMR owners.

We use two security games for further illustration. The first one is played between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$. It describes what an adversary could obtain in the real world. The second one is played between a simulator $\mathscr{S}$ and a challenger $\mathscr{C}$. It describes what a simulator could obtain in an ideal experiment.

The real game $\mathrm{Game}_{\mathrm{real},\mathscr{A}}(\lambda)$ of RBAC-UL:

Setup: the challenger $\mathscr{C}$ runs Setup to obtain the system parameter PK and gives PK to the adversary $\mathscr{A}$.

Challenge: $\mathscr{A}$ outputs a set of EMRs, $\overrightarrow{\mathrm{EMR}} = \{\mathrm{EMR}_1, \mathrm{EMR}_2, \ldots, \mathrm{EMR}_n\}$ on which it wishes to challenge, together with a set of access policy $P^* = \{P_1^*, P_2^*, \ldots, P_n^*\}$ including all the broadcast groups that it wishes to attack. In our case, a broadcast group represents a group of medical staff who are eligible to read a certain kind of EMR. For example, the medical staff in the access policy group P_2^* are authorized to fetch EMR_2, but no authority for other EMRs. Each access policy P_i^* should satisfy that for all the secret credential queries issued in Query Phase 1, there is $\overrightarrow{R} \notin \mathrm{Pref}(P_i^*)$. $\mathscr{C}$ randomly chooses $r[i] \longleftarrow Z_p$ for each EMR_i, where $p = |\mathbb{G}|$ and $i \in \{1, 2, \ldots, n\}$, and computes $\mathrm{En}_i \longleftarrow \mathrm{Enc}(\mathrm{PK}, P_i^*, \mathrm{EMR}_i, r_i)$. Finally, the challenger $\mathscr{C}$ returns the encapsulated EMRs $\overrightarrow{\mathrm{En}} = \{\mathrm{En}_1, \mathrm{En}_2, \ldots, \mathrm{En}_n\}$ to $\mathscr{A}$.

Query Phase 1: $\mathscr{A}$ issues a secret credential query for a medical staff associated with role $\overrightarrow{R}$. The challenger $\mathscr{C}$ generates a secret credential for $\overrightarrow{R}$ and returns it to $\mathscr{A}$.

Challenge: $\mathscr{A}$ outputs a set of EMRs, $\overrightarrow{\mathrm{EMR}} = \{\mathrm{EMR}_1, \mathrm{EMR}_2, \ldots, \mathrm{EMR}_n\}$ on which it wishes to challenge, together with a set of access policy $P^* = \{P_1^*, P_2^*, \ldots, P_n^*\}$ including all the broadcast groups that it wishes to attack. In our case, a broadcast group represents a group of medical staff who are eligible to read a certain kind of EMR. For example, the medical staff in the access policy group P_2^* are authorized to fetch EMR_2, but no authority for other EMRs. Each access policy P_i^* should satisfy that for all the secret credential queries issued in Query Phase 1, there is $\overrightarrow{R} \notin \mathrm{Pref}(P_i^*)$. $\mathscr{C}$ randomly chooses $r[i] \longleftarrow Z_p$ for each EMR_i, where $p = |\mathbb{G}|$ and $i \in \{1, 2, \ldots, n\}$, and computes $\mathrm{En}_i \longleftarrow \mathrm{Enc}(\mathrm{PK}, P_i^*, \mathrm{EMR}_i, r_i)$. Finally, the challenger $\mathscr{C}$ returns the encapsulated EMRs $\overrightarrow{\mathrm{En}} = \{\mathrm{En}_1, \mathrm{En}_2, \ldots, \mathrm{En}_n\}$ to $\mathscr{A}$.

Corrupt: the adversary $\mathscr{A}$ outputs a set $I \subseteq [1, n]$ to $\mathscr{C}$ on which $\mathscr{A}$ wishes to corrupt. $\mathscr{C}$ corrupts the corresponding EMRs to get $(\mathrm{EMR}_i, r_i)_{i \in I}$ and returns them to the adversary.

Query Phase 2: $\mathscr{A}$ issues a secret credential query for the medical staff with role $\overrightarrow{R}$ such that $\overrightarrow{R} \notin \mathrm{Pref}(P^*)$. The challenger responds the same as Query Phase 1.

Output: the adversary $\mathscr{A}$ outputs a bit $\mathrm{Out}_{\mathscr{A}}$.

The simulated game $\mathrm{Game}_{\mathrm{sim},\mathscr{S}}(\lambda)$ of RBAC-UL:

Setup: the simulator $\mathscr{S}$ gets system parameters from the challenger $\mathscr{C}$.

Challenge: $\mathscr{S}$ outputs a set of EMRs, $\overrightarrow{\mathrm{EMR}} = \{\mathrm{EMR}_1, \mathrm{EMR}_2, \ldots, \mathrm{EMR}_n\}$ on which it wishes to challenge, together with a set of access policy $P^* = \{P_1^*, P_2^*, \ldots, P_n^*\}$ including all the broadcast groups that it wishes to attack. The challenger $\mathscr{C}$ gets these inputs, but gives no feedback to the adversary $\mathscr{A}$.

Corrupt: $\mathscr{S}$ outputs a set $I \subseteq [1, n]$ to $\mathscr{C}$. The challenger $\mathscr{C}$ picks up the corresponding $\{\mathrm{EMR}_i\}_{i \in I}$ and returns them to the simulator.

Output: the simulator $\mathscr{S}$ outputs a bit $\mathrm{Out}_{\mathscr{S}}$.

We claim that if for every PPT adversary there exists a PPT simulator who can generate an indistinguishable output without seeing any encapsulated EMR and randomness, then the scheme achieves RBAC-UL security.

Definition 6. The advantage of a RBAC-UL adversary $\mathscr{A}$ against a RBAC scheme Γ with a simulator $\mathscr{S}$ is defined as follows:

$$\mathrm{Adv}_{\Gamma,\mathscr{S},\mathscr{A}}^{RBAC-UL}(\mathscr{A}) = \Big|\Pr\left[\mathrm{Game}_{\mathrm{real},\mathscr{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{Game}_{\mathrm{sim},\mathscr{S}}(\lambda) = 1\right]\Big|. \quad (11)$$

Definition 7 (RBAC-UL Security). Given an RBAC scheme, if no PPT distinguisher $\mathscr{D}$ can distinguish the output from $\mathrm{Game}_{\mathrm{real},\mathscr{A}}$ and $\mathrm{Game}_{\mathrm{sim},\mathscr{S}}$, namely,

$$\left|\Pr[\mathscr{D}(\mathrm{Out}_{\mathscr{A}} = 1)] - \Pr[\mathscr{D}(\mathrm{Out}_{\mathscr{S}} = 1)]\right| = \epsilon, \quad (12)$$

where ϵ is a negligible function in the security parameter, then we claim that the scheme achieves RBAC-UL security.

5.2. The Adversary Model for RBAC-IND.

The adversary model for RBAC-Indistinguishability (RBAC-IND) is aimed to satisfy the requirement of *data privacy*. The indistinguishability can be illustrated as follows: a recipient (the medical staff who are eligible to get access on one EMR) generates a credential pair; a sender (EMR owner) encapsulates one out of two EMRs and send it to the adversary; the RBAC-IND adversary tries to find out which one it was. Here, importantly, the adversary has the authority to issue secret credential query. If the adversary cannot distinguish an encapsulation of challenge EMR from an encapsulation of random message, we claim that our system achieves RBAC-IND security.

The RBAC-IND model is defined by a security game played between a challenger $\mathscr{C}$ and an adversary $\mathscr{A}$. We apply the full security notion [33, 34] to the RBAC-IND model. That means the adversary can adaptively output the access policy that it wishes to attack during the system interaction.

The security game $\mathrm{Game}_{\Gamma,\mathscr{A}}^{\mathrm{RBAC-IND}}(\lambda)$: let $\Gamma = (\mathrm{Setup}, \mathrm{SCGen}, \mathrm{SCDeleg}, \mathrm{EMREnc}, \mathrm{EMRDec})$ be a role-based access control scheme.

Setup: the challenger runs the setup algorithm to obtain a public key PK and gives it to the adversary $\mathscr{A}$.

Query Phase 1: $\mathscr{A}$ issues a secret credential query for the medical staff with role $\overrightarrow{R}$. The challenger generates a secret credential for $\overrightarrow{R}$ and gives it to the adversary.

Challenge: the adversary $\mathscr{A}$ outputs two equal-length EMR files, EMR_0, and EMR_1 on which it wishes to challenge. $\mathscr{A}$ outputs a challenge access policy P^* either. The access policy P^* should satisfy that for all the secret credential queries for $\overrightarrow{R}$ issued in Query Phase, $\overrightarrow{R} \notin \mathrm{Pref}(P^*)$. The challenger flips a random coin $\beta \in \{0, 1\}$ and encapsulates EMR_β under the challenge access policy P^*. Then, it returns the encapsulated EMR to $\mathscr{A}$.

Query Phase 2: $\mathscr{A}$ issues a credential query for the medical staff with role $\overrightarrow{R}$ such that $\overrightarrow{R} \notin \mathrm{Pref}(P^*)$. The challenger responds the same as in Query Phase.

Guess: the adversary $\mathscr{A}$ guesses $\beta' \in \{0,1\}$. It ouputs 1 if $\beta = \beta'$ and 0 otherwise. We say $\mathscr{A}$ succeeds if $\beta = \beta'$.

Definition 8. Given an RBAC scheme Γ, we define the probability advantage for an RBAC-IND adversary $\mathscr{A}$ of winning the security game $\mathrm{Game}_{\Gamma,\mathscr{A}}^{RBAC-IND}(\lambda)$ to be

$$\mathrm{Adv}_{\Gamma,\mathscr{A}}^{\mathrm{RBAC-IND}}(\lambda) = \left|2 \cdot \Pr\left[\mathrm{Game}_{\Gamma,\mathscr{A}}^{\mathrm{RBAC-IND}}(\lambda) = 1\right] - 1\right|. \tag{13}$$

Definition 9 (RBAC-IND security). Given an RBAC scheme Γ, if for each polynomial-time RBAC-IND adversary, its advantage $\mathrm{Adv}_{\Gamma,\mathscr{A}}^{\mathrm{RBAC-IND}}(\lambda)$ is a negligible function in security parameter λ, then the scheme Γ achieves RBAC-IND security.

6. Proposed Solution

Our technical solution leverages hierarchical identity-based encryption (HIBE) [35] and extends it to a multireceiver scenario. In HIBE, an encryptor can only encrypt a single path, which implies either repetitive encryption or constrained access policy for multiple receivers. Our role-based access control solution supports fine-grained access by encapsulating EMRs to any subset of hierarchically organized users, which is based on HIBBE. Furthermore, it resists many types of unexpected leakage, so that the leakage effect can be controlled in a certain scope. The following subsections show how we achieve this target. In Section 6.1, we construct a one-bit RBAC scheme with one-sided public leakage (1SPL) functionality. 1SPL means there exists a public procedure that given the one-bit encapsulation message En^1 of "1" can compute the randomness r under which the encapsulation applied to "1" would generate En^1. En^1 is used to denote the encapsulation message of "1." The idea comes from the notion of one-side public openability [36]. In Section 6.2, we provide security analysis for the proposed one-bit RBAC scheme with 1SPL. In Section 6.4, we provide a reduction showing that if a one-bit RBAC scheme with 1SPL functionality is secure, the normal multibit scheme with RBAC-UL model is secure. Our solution achieves data privacy, leakage controllability, and flexible access control.

6.1. Construction of One-Bit RBAC Scheme with 1SPL

Setup(λ, n). The system setup algorithm is run by TKA. It chooses a bilinear group G of order N, and random elements g_1, g_2, g_3, g_4 from $G_{p_1}, G_{p_2}, G_{p_3}, G_{p_4}$, random exponents $u_{11}, u_{12}, \ldots, u_{1n}, u_4, x_1, x_4, \omega_4 \xleftarrow{R} Z_N$, and computes $U_{1i} \xleftarrow{R} g_1^{u_{1i}}$, $U_4 \xleftarrow{R} g_4^{u_4}$, $X_1 \xleftarrow{R} g_1^{x_1}$, $X_4 \xleftarrow{R} g_4^{x_4}$, $W_4 \xleftarrow{R} g_4^{\omega_4}$, $U_{1i,4} \xleftarrow{R} U_{1i}U_4$, $W_{14} \xleftarrow{R} g_1 W_4$, $X_{14} \xleftarrow{R} X_1 X_4$ for $i \in [1,n]$. It outputs a public key $\mathrm{PK} = \left\{N, \left\{U_{1i,4}\right\}_{i\in[1,n]}, X_{14}, W_{14}, g_4\right\}$ and a master secret key $\mathrm{MSK} = \left\{g_1, g_3, \{U_{1i}\}_{i\in[1,n]}, X_1\right\}$.

SCGen$(\mathrm{PK}, \mathrm{MSK}, \overrightarrow{R})$. This is the secret credential generation algorithm. For the medical staff with role $\overrightarrow{R} = (R_1, \ldots, R_d)$, we denote $I = \left\{i\colon R_i \in S_{\overrightarrow{R}}\right\}$. When a medical staff at the top-level joins a hospital organization, TKA generates a secret credential $\mathrm{SC}^{\overrightarrow{R}}$ for them:

$$\begin{aligned} K_1 &= g_1^{r} g_3^{r_3}, \\ K_2 &= \left(\prod_{i\in I} U_{1i}^{R_i} \cdot X_1\right)^{r} \cdot g_3^{r_3'}, \\ E_j &= \left\{U_{1j}^{r} \cdot g_3^{r_j}\right\}_{j\in[i,n]/I}, \end{aligned} \tag{14}$$

where $r, r_3, r_3', \left\{r_j\right\}_{j\in([1,n]/I)} \xleftarrow{R} Z_N$.

SCDeleg$(\mathrm{PK}, \mathrm{SC}^{\overrightarrow{R'}}, R)$. This is the secret credential delegation algorithm. A junior medical staff with role $\overrightarrow{R} = (\overrightarrow{R'}, R)$ is authenticated by a supervisor with role $\overrightarrow{R'}$. His supervisor delegates a secret credential for them:

$$\begin{aligned} K_1 &= K_1' g_1^{r'} g_3^{\widetilde{r_3}}, \\ K_2 &= K_2'\left(\prod_{i\in I} U_{1i}^{R_i} \cdot X_1\right)^{r'} \cdot (E_i')_{i\,\in\,(I/I')}^{R} \cdot g_3^{\widetilde{r_3'}}, \\ E_j &= \left\{E_j' \cdot U_{1,j}^{r'} \cdot g_3^{r_j'}\right\}_{j\in[1,n]/I} = \left\{U_{1,j}^{r+r'} \cdot g_3^{r_j'+r_j}\right\}_{j\in[1,n]/I}, \end{aligned} \tag{15}$$

where $I' = \left\{i\colon R_i \in S_{\overrightarrow{R'}}\right\}$ and $r', \widetilde{r_3}, \widetilde{r_3'}, \left\{r_j'\right\}_{j\in[i,n]/I} \xleftarrow{R} Z_N$. It can be computed as

$$\begin{aligned} K_1 &= g_1^{\widehat{r}} g_3^{\widehat{r_3}}, \\ K_2 &= \left(\prod_{i\in I} U_{1i}^{R_i} \cdot X_1\right)^{\widehat{r}} \cdot g_3^{\widehat{r_3'}}, \\ E_j &= \left\{U_{1j}^{\widehat{r}} \cdot g_3^{\widehat{r_j}}\right\}_{j\in[i,n]/I}. \end{aligned} \tag{16}$$

The delegated credential is well formed as if it is generated by TKA with SCGen algorithm.

EMREnc$(\mathrm{PK}, P, \mathrm{EMR})$: this is the EMR encapsulation algorithm. For an access policy P, we denote $\mathrm{II} = \{i\colon R_i \in S_P\}$. When a single bit 0 from EMR data needs to be encapsulated under P, a medical staff chooses random $s, t_4, t_4' \xleftarrow{R} Z_N$ and computes $\mathrm{En}_1 = (\prod_{i\in\mathbb{I}} U_{1i,4}^{R_i} \cdot X_{14})^s g_4^{t_4}$, $\mathrm{En}_2 = W_{14} g_4^{t_4'}$. When a single bit 1 needs to be encapsulated, a medical staff sets $\mathrm{En}_1, \mathrm{En}_2 \xleftarrow{R} G$.

$\mathrm{EMRDec}(\mathrm{PK}, \overrightarrow{R}, \mathrm{En}, \mathrm{SC}^{\overrightarrow{R}})$: this is the EMR decapsulation algorithm. The medical staff with role satisfied by an access policy P can use his secret credential to recover all one-bit messages for EMR data. If $e(\mathrm{En}_1, K_1) = e(\mathrm{En}_2, K_2)$, a medical staff returns bit 0. Otherwise, he returns bit 1.

Correctness: We need to verify when input a well-formed encapsulation $\mathrm{En} = (\mathrm{En}_1, \mathrm{En}_2)$ with a valid credential $\mathrm{SC}^{\overrightarrow{R}}$ for 0 bit, whether $e(\mathrm{En}_1, K_1) = e(\mathrm{En}_2, K_2)$ holds.

$$e(\mathrm{En}_1, K_1) = e\left(\left(\prod_j U_{1j,4}^{id_j} \cdot X_{14}\right)^s g_4^{t_4}, g_1^r g_3^r\right),$$
$$e(\mathrm{En}_2, K_2) = e\left(g_1^s \cdot g_4^{\omega_4 s + t_4'}, \left(g_1^{u_{11} \cdot id_1} \cdots g_1^{u_{1j} \cdot id_j} \cdot g_1^{x_1}\right)^r \cdot g_3^{r_3'}\right). \tag{17}$$

Due to the orthogonality property, we get $e(\mathrm{En}_1, K_1) = e(\mathrm{En}_2, K_2) = e(g_1^s, g_1^{r \cdot (\sum_{i=1}^{j} u_{1i} \cdot id_1 + x_1)})$. Therefore, when $\mathrm{En} = (\mathrm{En}_1, \mathrm{En}_2)$ is a well-formed EMR encapsulation, the decapsulation algorithm can correctly recover EMR with a valid credential $\mathrm{SC}^{\overrightarrow{R}}$.

6.2. Security Analysis of One-Bit RBAC Scheme. We prove the security by contradiction. Assume a PPT adversary can break the one-bit RBAC scheme in polynomial time. Then we solve a series of hard-to-solve problems based on subgroup decision assumptions, which are introduced in Section 3.2. Since no PPT algorithm could solve these problems, we reach a contradiction and conclude our proposed scheme is secure.

Theorem 1. *Suppose $\mathbb{G}$ is a group of composite order $N = p_1 p_2 p_3 p_4$, equipped with an efficient bilinear map. Suppose that the Assumption 1–5 hold in $\mathbb{G}$. Then our one-bit RBAC scheme is secure under the formal security model.*

We apply the dual system encapsulation technique [37] to the one-bit RBAC scheme, where the encapsulated message En and the credential SC can take one of two indistinguishable form: normal form and semifunctional form. The correlation between them is shown in Table 2. "√" means is decapsulation allowed and "×" means decapsulation is not allowed. When all the EMR encapsulations and credentials are semifunctional, the adversary obtains no information for the challenge encapsulated EMR since none of the given credential is useful to decapsulate it.

In the next section, we show that no PPT algorithm can distinguish between $\mathrm{Game}_{\mathrm{real}}$ and $\mathrm{Game}_{\mathrm{final}}$. All the components in the encapsulation of $\mathrm{Game}_{\mathrm{final}}$ are random elements, so it does not leak any EMR information. The indistinguishability between those games proves Theorem 1.

Semifunctional ciphertext: a user runs EHRGen to construct a normal ciphertext $(\mathrm{En}_1', \mathrm{En}_1')$. Then they pick up random exponents $x, z_c \in Z_N$ and sets $\mathrm{En}_1 = \mathrm{En}_1' g_2^{x z_c}$, $\mathrm{En}_2 = \mathrm{En}_2' g_2^{x}$.

Table 2: Correlation of normal and semifunctional forms.

	Normal En	Semifunctional En
Normal SC	√	√
Semifunctional SC	√	×

Semifunctional secret credential TKA runs the algorithm SCGenM to generate a normal key $(K_1', K_2', \{E_j\}_{j \in [1,n]/I})$. It chooses random exponents $\gamma, z_k, \{z_j\}_{j \in [1,n]/I} \in Z_N$. The semifunctional key is set as $K_1 = K_1' g_2^{\gamma}, K_2 = K_2' g_2^{\gamma \cdot z_k}, \{E_j = E_j^A g_2^{\gamma z_j}\}$.

It is straightforward that the EHRDecM algorithm correctly outputs EHR when decrypting a semifunctional ciphertext by a semifunctional key since the added elements in G_{p_2} can be cleared due to orthogonality property. However, the blinding factor is multiplied by an additional term $e(g_2, g_2)^{\gamma x (z_k + \sum_{i \in (\text{▯}/I)} z_i R_i - z_c)}$. If $z_c = z_k + \sum_{i \in (\text{▯}/I)} z_i R_i$, decryption still works. In this case, we call the secret credential is nominally semifunctional. We prove Theorem 1 through following games between an adversary and a challenger.

$\mathrm{Game}_{\mathrm{real}}$: this is the real game.

$\mathrm{Game}_{\mathrm{real}'}$: this game is the same as $\mathrm{Game}_{\mathrm{real}}$ except that all the secret credential queries are answered by the secret credential generation algorithm, not by the secret credential delegation algorithm.

$\mathrm{Game}_{\mathrm{res}}$: this game is the same as $\mathrm{Game}_{\mathrm{real}'}$ except that the adversary cannot ask for secret credentials for the roles which are the prefixes of the challenge role modulo p_2. Namely, it is not allowed that, for any queried role $\overrightarrow{R} = (R_1, \ldots, R_d)$, $\exists\, \overrightarrow{R^*} = (R_1^*, R_2^*, \ldots, R_{d'}^*) \in \mathrm{Pref}(P^*)$ with $d' \le d$, s.t. $\forall i \in [1, d'], R_i = R_i^* \bmod p_2$. P^* is the set of challenge access policy.

Game_k: this game is identical with $\mathrm{Game}_{\mathrm{res}}$ except that the EMR encapsulation given to adversary is semifunctional and the first k credentials are semifunctional $(0 \le k \le q)$ for bit 0. We notice that in Game_q, the EMR encapsulation and all credentials are semifunctional.

$\mathrm{Game}_{\mathrm{final}'}$: this game is identical with Game_q except that challenge EMR encapsulation is a semifunctional encapsulation for a random message in subgroup of $\mathbb{G}$ for bit 0, not one of the messages given by the adversary.

$\mathrm{Game}_{\mathrm{final}}$: this game is identical with $\mathrm{Game}_{\mathrm{final}'}$ except that $\mathrm{Game}_{\mathrm{final}}$ replaces the challenge EMR encapsulation of 0 by a pair of random points in the full group $\mathbb{G}$.

6.3. Proof of Theorem 1. In this section, we use six lemmas to prove Theorem 1. Each lemma demonstrates the indistinguishability between the neighbouring games.

Lemma 1. *For any PPT algorithm $\mathscr{A}$, it holds that: $Game_{real}Adv_{\mathscr{A}}(\lambda) = Game_{real_{\prime}}Adv_{\mathscr{A}}(\lambda)$.*

Proof of Lemma 1. We note that the secret credentials are identically distributed whether they are generated by the credential generated algorithm or by the credential delegation algorithm. So, there is no difference between $\mathrm{Game}_{\mathrm{real}}\mathrm{Adv}_{\mathscr{A}}$ and $\mathrm{Game}_{\mathrm{real}'}\mathrm{Adv}_{\mathscr{A}}$ from the adversary's view. □

Lemma 2. *Suppose there exists a PPT algorithm $\mathscr{A}$ such that $|Game_{real'}Adv_{\mathscr{A}}(\lambda) - Game_{res}Adv_{\mathscr{A}}(\lambda)| = \epsilon_1$. Then, we can build a polynomial-time algorithm $\mathscr{B}$ with advantage $\varepsilon_1/3$ in breaking Assumption 1.*

Proof of Lemma 2. If there exists a PPT adversary $\mathscr{A}$ that distinguishes $\mathrm{Game}_{\mathrm{real}'}$ and $\mathrm{Game}_{\mathrm{res}}$ with probability ε_1, by the definition of $\mathrm{Game}_{\mathrm{res}}$, $\mathscr{A}$ knows that it issues a secret credential query for the medical staff with role $\overrightarrow{R} = \{R_1, \ldots, R_d\}$ from others satisfying that $\exists \overrightarrow{R^*} = (R_1^*, R_2^*, \ldots, R_{d'}^*) \in \mathrm{Pref}(P^*)$ with $d' \leq d$, s.t. $\forall i \in [1, d']$, $R_i = R_i^* \bmod p_2$. Then the factor of N can be extracted by computing $\gcd(R_i - R_i^*, N)$, from which we design an algorithm $\mathscr{B}$ breaking Assumption 1 as follows.

$\mathscr{B}$ receives g_1, g_3, g_4, and produces a nontrivial factor of N by computing $r = \gcd(R_i - R_i^*, N)$. We want to use r to generate a point in $\mathbb{G}(Q)$, where $\mathbb{G}(Q)$ denotes the unique subgroup with order $\prod_{j\in Q} p_j$ $(Q \subseteq [4])$. We set $2 \in Q$ but $k \notin Q$ so that $\mathbb{G}(Q)$ can be used to test T with orthogonality. Enumerate the cases for r, N/r, and the resulting k and Q in Table 3. As N/r is the complementary set of r, it covers all the possibilities for the subgroup with different orders.

Due to the rule that $2 \in Q$ but $k \notin Q$, we get from Table 3 at least one choice of k that allows us to use r or N/r to construct a point in $\mathbb{G}(Q)$. $\mathscr{B}$ immediately decides T by orthogonality. For example, if r is p_1, T is chosen from G_{p_1} or $G_{p_1p_2}$. Also we can get elements from $\mathbb{G}(Q)$, i.e., we select $X \xleftarrow{R} G_{p_2p_3p_4}$. Then $\mathscr{B}$ learns whether T has a G_{p_2} component or not by testing if $e(T, X) = 1$. If not, T has a G_{p_2} component. From the point of $\mathscr{A}$'s view, the choice of i is independent, and $\mathscr{B}$ at least has 1/3 chance to pick an i that works.

Compared with $\mathrm{Game}_{\mathrm{res}}$, the challenge encapsulation of a 0 bit is replaced with a semifunctional one in Game_0, meaning its components are multiplied by points in G_{p_2}. As the adversary does not know the factor of $N = p_1p_2p_3p_4$, it cannot determine whether the components of the challenge encapsulation of a 0 are in $G_{p_1p_4}$ or in $G_{p_1p_2p_4}$. Hence, the adversary is not able to know which form the given challenge encapsulation is. □

Lemma 3. *Suppose there exists a PPT algorithm $\mathscr{A}$ such that $|Game_{res}Adv_{\mathscr{A}}(\lambda) - Game_0Adv_{\mathscr{A}}(\lambda)| = \epsilon_2$. We can build a polynomial-time algorithm $\mathscr{B}$ with advantage ϵ_2 in breaking Assumption 2.*

Table 3: Possibilities for $r, N/r, k, Q$.

r	N/r	k	Q
p_1	$p_2p_3p_4$	1	2 3 4
p_2	$p_1p_3p_4$	1 3 4	2
p_3	$p_1p_2p_4$	3	1 2 4
p_4	$p_1p_2p_3$	4	1 2 3
p_1p_2	p_3p_4	3 4	1 2
p_1p_3	p_2p_4	1 3	2 4
p_1p_4	p_2p_3	1 4	2 3

Proof of Lemma 3. The input of algorithm $\mathscr{B}$ is the challenge tuple (g_1, g_3, g_4, T) of Assumption 2. $\mathscr{B}$ has to decide whether T is in $G_{p_1p_4}$ or in $G_{p_1p_2p_4}$.

Setup: on inputs (g_1, g_3, g_4, T), $\mathscr{B}$ picks random exponents $\{a_i\}_{i\in[1,n]}, b, x, y, \omega$ from Z_N, and sets $U_{1i} \xleftarrow{R} g_1^{a_i}$, $X_1 \xleftarrow{R} g_1^{b}, X_4 \xleftarrow{R} g_4^{x}, U_4 \xleftarrow{R} g_4^{y}$, $W_4 \xleftarrow{R} g_4^{\omega}$, $U_{1i,4} \xleftarrow{R} U_{1i}U_4$, $X_{14} \xleftarrow{R} X_1X_4$, $W_{14} \xleftarrow{R} g_1W_4$. It sends public paramter $\{N, U_{1i,4}, X_{14}, W_{14}, g_4\}_{i\in[1,n]}$ to $\mathscr{A}$

Query Phase 1: when the adversary $\mathscr{A}$ issues a secret credential query for a medical staff with role $\overrightarrow{R} = (R_1, \ldots, R_d)$, $\mathscr{B}$ randomly chooses exponents $r, r_3, r_3', \{r_j\}_{j\in([1,n]/I)} \xleftarrow{R} Z_N$ where $I = \{i\colon R_i \in S_{\overrightarrow{R}}\}$ and sets

$$\begin{aligned} K_1 &= g_1^{r} g_3^{r_3}, \\ K_2 &= \left(\prod_{i\in I} U_{1i}^{R_i} \cdot X_1\right)^{r} \cdot g_3^{r_3'}, \\ E_j &= \left\{U_{1j}^{r} \cdot g_3^{r_j}\right\}_{j\in[i,n]/I}. \end{aligned} \tag{18}$$

It has the same distribution as that of the normal secret credential.

Challenge: $\mathscr{A}$ outputs two EMR files EMR_0 and EMR_1 and a challenge access policy P^*. The challenge access policy P^* must satisfy the property that no revealed role in Query Phase 1 was a prefix of its components. $\mathscr{B}$ picks a random coin $\beta \xleftarrow{R} \{0,1\}$ and gives the challenge EMR encapsulation as following. We denote that $\mathrm{II}^* = \{i\colon R_i^* \in S_{P^*}\}$.

(i) $\mathrm{EMR}_\beta = 0$. $\mathscr{B}$ lets $z_c = \sum_{i\in\mathrm{II}^*} a_iR_i^* + b$ and sets $\mathrm{En}_1 \xleftarrow{R} T^{z_c}$, $\mathrm{En}_2 \xleftarrow{R} T$. If $\mathscr{B}$'s challenge bit $\beta = 0$, $T \xleftarrow{R} G_{p_1p_2p_4}$. We write $T = (g_1W_4)^s g_4^{t_4'} g_2^{x_2}$ for random $s, t_4', x_2 \xleftarrow{R} Z_N$ and get

$$\begin{aligned} \mathrm{En}_1 &= \left((g_1W_4)^s g_4^{t_4'} g_2^{x_2}\right)^{z_c} = \left(\prod_{i\in\mathrm{II}^*} U_{1i,4}^{R_i} \cdot X_{14}\right)^{s} g_4^{(s\omega+t_4')\cdot z_c - \left(y\sum_{i\in\mathrm{II}^*} id_i + x\right)s} \cdot g_2^{x_2\cdot z_c}, \\ \mathrm{En}_2 &= (g_1W_4)^s g_4^{t_4'} g_2^{x_2} = W_{14}^s g_4^{t_4'} g_2^{x_2}. \end{aligned} \tag{19}$$

This implicitly sets $t_4 = (s\omega + t_4') \cdot z_c - (y\sum_{i\in\mathbb{I}^*} R_i + x)s \bmod p_4$ and $x_2' = x_2 \cdot z_c \bmod p_2$. The challenge encapsulation is semifunctional formed in $\mathrm{Game}_0^{\mathscr{A}}$. If $\mathscr{B}$'s challenge bit $\beta = 1$, $T \xleftarrow{R} G_{p_1p_4}$. We write $T = (g_1W_4)^s g_4^{t_4'}$ for random $s, t_4' \xleftarrow{R} Z_N$, and get

$$\begin{aligned}\mathrm{En}_1 &= \left((g_1W_4)^s g_4^{t_4'}\right)^{Z_c} = \left(\prod_{i\in\mathrm{II}^*} U_{1i,4}^{R_i} \cdot X_{14}\right)^s \\ &\quad g_4^{(s\omega+t_4')\cdot z_c - \left(y\sum_{i\in\mathrm{II}^*} id_i + x\right)s}, \\ \mathrm{En}_2 &= (g_1W_4)^s g_4^{t_4'} = W_{14}^s g_4^{t_4'}.\end{aligned} \tag{20}$$

This implicitly sets $t_4 = (s\omega + t_4') \cdot z_c - (y\sum_{i\in\mathrm{II}^*} R_i + x)s \bmod p_4$. The challenge EMR encapsulation is normally formed in $\mathrm{Game}_{\mathrm{res}}$.

(ii) $\mathrm{EMR}_\beta = 1$. $\mathscr{B}$ sets $\mathrm{En}_1, \mathrm{En}_2 \xleftarrow{R} \mathbb{G}$.

Query Phase 2: Query Phase 1 is repeated adaptively except that $\overrightarrow{R} \notin \mathrm{Pref}(P^*)$.

Guess: the adversary $\mathscr{A}$ outputs a guess that it is in Game_0 or $\mathrm{Game}_{\mathrm{res}}$. The simulator $\mathscr{B}$ guesses $T \xleftarrow{R} G_{p_1p_4}$ if $\mathscr{A}$ decides it is in $\mathrm{Game}_{\mathrm{res}}$ $(\beta = 1)$. $\mathscr{B}$ outputs $T \xleftarrow{R} G_{p_1p_2p_4}$ if $\mathscr{A}$ decides it is in Game_0 $(\beta = 0)$. If $\mathscr{A}$ has the advantage ε_2 to distinguish $\mathrm{Game}_{\mathrm{res}}$ and Game_0, $\mathscr{B}$ can break Assumption 2 with advantage ϵ_2.

Game_{k-1} and Game_k are two distinguishable games. The way to decide whether the kth queried credential is normal or semifunctional is to decide whether the credential components are in $G_{p_1p_3}$ or in $G_{p_1p_2p_3}$. This is computationally difficult without knowing factor $N = p_1p_2p_3p_4$. □

Lemma 4. *Suppose there exists a PPT algorithm $\mathscr{A}$ such that $|\mathrm{Game}_{k-1}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_k\mathrm{Adv}_{\mathscr{A}}(\lambda)| = \varepsilon_3$. Then, we can build a polynomial-time algorithm $\mathscr{B}$ with advantage ε_3 in breaking Assumption 3.*

Proof of Lemma 4. The input of $\mathscr{B}$ is the challenge tuple $(g_1, g_3, g_4, D_{23}, A_{12}, T)$ of Assumption 3. $\mathscr{B}$ has to decide whether T is in $G_{p_1p_3}$ or in $G_{p_1p_2p_3}$.

Setup: $\mathscr{B}$ receives $g_1, g_3, g_4, D_{23}, A_{12}, T$. It picks random exponents $\{a_1\}_{i\in[1,n]}, b, x, y, \omega$ from Z_N, and sets $U_{1i} \xleftarrow{R} g_1^{a_i}$, $X_1 \xleftarrow{R} g_1^b, X_4 \xleftarrow{R} g_4^x, U_4 \xleftarrow{R} g_4^y$, $W_4 \xleftarrow{R} g_4^\omega, U_{1i,4} \xleftarrow{R} U_{1i}U_4, X_{14} \xleftarrow{R} X_1X_4, W_{14} \xleftarrow{R} g_1$ W_4 and $k \xleftarrow{R} [0, q]$. It sends the public parameters $\{N, U_{1i,4}, X_{14}, W_{14}, g_4\}_{i\in[1,n]}$ to $\mathscr{A}$.

Query Phase 1: when $\mathscr{A}$ requests the ℓth credential for $\overrightarrow{R} = (R_1, \ldots, R_d)$ where $I = \left\{i\colon R_i \in S_{\overrightarrow{R}}\right\}$, we consider three cases: $\ell < k$, $\ell > k$, and $\ell = k$.

(i) When $\ell < k$, $\mathscr{B}$ creates a semifunction credential by picking up random exponents $r, z, z^A, \{z_j\}_{j\in([1,n]/I)}$ from Z_N and setting

$$\begin{aligned}K_1 &= g_1^r D_{23}^z, \\ K_2 &= \left(\prod_{i\in I} U_{1i}^{R_i} \cdot X_1\right)^r \cdot D_{23}^{\acute{z}}, \\ E_j &= \left\{U_{1,j}^r D_{23}^{z_j}\right\}_{j\in([1,n]/I)}.\end{aligned} \tag{21}$$

Consider D_{23} as $g_2^{y_2}g_3^{y_3}$ for random $y_2, y_3 \xleftarrow{R} Z_N$, then

$$\begin{aligned}K_1 &= g_1^r g_2^{y_2z} g_3^{y_3z}, \\ K_2 &= \left(\prod_{i\in I} U_{1i}^{R_i} \cdot X_1\right)^r \cdot g_2^{z'y_2} g_3^{z'y_3}, \\ E_j &= \left\{U_{1,j}^r g_3^{y_3\cdot z_j} g_2^{y_2\cdot z_j}\right\}_{j\in([1,n]/I)}.\end{aligned} \tag{22}$$

This is a properly distributed semifunctional credential.

(ii) When $\ell > k$, $\mathscr{B}$ creates a normal credential by invoking the usual credential generation algorithm.

(iii) When $\ell = k$, $\mathscr{B}$ creates the kth credential. $\mathscr{B}$ lets $z_k = \sum_{i\in I} a_iR_i + b$ and sets

$$\begin{aligned}K_1 &\xleftarrow{R} T, \\ K_2 &\xleftarrow{R} T^{z_k}, \\ E_j &\xleftarrow{R} \{T^{a_j}\}_{j\in([1,n]/I)}.\end{aligned} \tag{23}$$

If $T \xleftarrow{R} G_{p_1p_3}$, $T = g_1^r g_3^{r_3}$ for random $r, r_3 \xleftarrow{R} Z_N$, then

$$\begin{aligned}K_1 &= g_1^r g_3^{r_3}, \\ K_2 &= (g_1^r g_3^{r_3})^{z_k} = \left(\prod_{i\in I} U_{1i}^{R_i} \cdot X_1\right)^r g_3^{r_3\cdot z_k}, \\ E_j &= \left\{(g_1^r g_3^{r_3})^{a_j}\right\}_{j\in([1,n]/I)} = \left\{U_{1,j}^r g_3^{r_3\cdot a_j}\right\}_{j\in([1,n]/I)}.\end{aligned} \tag{24}$$

It has the same distribution as the normal credential. If $T \xleftarrow{R} G_{p_1p_2p_3}$, we write $T = g_1^r g_3^{r_3} g_2^{t_2}$ for random $r, r_3, t_2 \xleftarrow{R} Z_N$, then

$$\begin{aligned}K_1 &= g_1^r g_3^{r_3} g_2^{t_2}, \\ K_2 &= \left(g_1^r g_3^{r_3} g_2^{t_2}\right)^{Z_k} = \left(\prod_{i\in I} U_{1i}^{R_i} \cdot X_1\right)^r g_3^{r_3\cdot z_k} g_2^{t_2\cdot z_k}, \\ E_j &= \left\{\left(g_1^r g_3^{r_3} g_2^{t_2}\right)^{a_j}\right\} = \left\{U_{1,j}^r g_3^{r_3\cdot a_j} g_2^{t_2\cdot a_j}\right\}_{j\in([1,n]/I)}.\end{aligned} \tag{25}$$

It has the same distribution as the semifunctional credential.

Challenge: at some points, $\mathscr{A}$ decides that it obtains enough secret credentials, it outputs two EMR files EMR_0 and EMR_1 and a challenge access policy P^*.

This policy must satisfy that no revealed role in Query Phase 1 was a prefix of its components. $\mathscr{B}$ picks up a random coin $\mathscr{B}\xleftarrow{R}\{0,1\}$ and gives the challenge EMR encapsulation as follows:

(i) $\mathrm{EMR}_{\beta} = 0$. $\mathscr{B}$ picks up $s_4, s_4'\xleftarrow{R}Z_N$ and sets $\mathrm{En}_1\xleftarrow{R}A_{12}^{z_c}g_4^{s_4}, \mathrm{En}_2\xleftarrow{R}A_{12}g_4^{s_4'}$. Consider A_{12} as $g_1^s g_2^{x_2}$ for random $s, x_2\xleftarrow{R}Z_N$, and get

$$\begin{aligned}\mathrm{En}_1 &= \left(\prod_{i\in \mathrm{II}^*} U_{1i}^{R_i^*}\cdot X_{14}\right)^s \cdot g_2^{x_2\cdot z_c} \\ &\quad \cdot g_4^{s_4 - xs - ys\sum_{i\in \mathrm{II}^*} R_i^*}, \\ \mathrm{En}_2 &= g_1^s g_2^{x_2} g_4^{s_4'} = \left(g_1 g_4^{\omega}\right)^s \cdot g_4^{s'-\omega s} \\ &\quad \cdot g_2^{x_2} = g_1^s W_4^s g_4^{s_4'-\omega s} g_2^{x_2} \\ &= W_{14}^s g_4^{t_4'} g_2^{x_2}.\end{aligned} \tag{26}$$

This implicitly sets $t_4 = s_4 - xs - ys\sum_{i\in \mathrm{II}^*} R_i^* \bmod p_4$, $x_2' = x_2\cdot z_c \bmod p_2$ and $t_4' = s\prime - \omega s \bmod p_4$ for $\mathbb{I}^* = \{i\colon R_i^* \in S_{P*}\}$.

(ii) $\mathrm{EMR}_{\beta} = 1$. $\mathscr{B}$ sets $\mathrm{En}_1, \mathrm{En}_2\xleftarrow{R}\mathbb{G}$.

The challenge encapsulation for $\mathrm{EMR}_{\beta} = 0$ is formed as the semifunctional form with $z_c = \sum_{i\in \mathrm{II}^*} a_i R_i^* + b$. Since from $\mathrm{Game}_{\mathrm{res}}$, the role $\overrightarrow{R} = (R_1, \ldots, R_d)$ associated with the kth secret credential is not a prefix of the challenge receiver role $\overrightarrow{R}^* = (R_1^*, \ldots, R_d^*)$ modulo p_2, the variables z_c and z_k are randomly distributed to the adversary $\mathscr{A}$. The relationship between z_c and z_k do not help $\mathscr{A}$ to distinguish the two games.

Query Phase 2: Query Phase 1 is repeated except $\overrightarrow{R} \notin \mathrm{Pref}(P^*)$.

Guess: the adversary $\mathscr{A}$ outputs a guess that it is in Game_{k-1} or Game_k.

$\mathscr{B}$ outputs $T\xleftarrow{R}G_{p_1p_3}$ if $\mathscr{A}$ decides it is in Game_{k-1}, where all components in the kth secret credential by algorithm $\mathscr{B}$ are in $G_{p_1p_3}$. Otherwise, $\mathscr{B}$ outputs $T\xleftarrow{R}G_{p_1p_2p_3}$ if $\mathscr{A}$ decides it is in Game_k, where all components in the kth secret credential by algorithm $\mathscr{B}$ are in $G_{p_1p_2p_3}$. If $\mathscr{A}$ has the advantage ε_3 to distinguish Game_{k-1} and Game_k, $\mathscr{B}$ can break Assumption 3 with advantage ϵ_3. □

Lemma 5. *Suppose there exists a PPT algorithm $\mathscr{A}$ such that $|Game_q Adv_{\mathscr{A}}(\lambda) - Game_{final'} Adv_{\mathscr{A}}(\lambda)| = \varepsilon_4$. Then we can build a polynomial-time algorithm $\mathscr{B}$ with advantage ϵ_4 in breaking Assumption 4.*

Proof of Lemma 5. The input of algorithm $\mathscr{B}$ is the challenge tuple $(g_2, g_3, g_4, W_{14}, E_{12}, T)$ of Assumption 4. Algorithm $\mathscr{B}$ has to answer whether T is in $G_{p_2p_4}$ or in $G_{p_1p_2p_4}$.

Setup: $\mathscr{B}$ first receives $g_2, g_3, g_4, W_{14}, E_{12}, T$. It then picks random exponents $\{a_i\}_{i\in[1,n]}$, b from Z_N, and sets $U_{1i,4}\xleftarrow{R}W_{14}^{a_i}$, $X_{14}\xleftarrow{R}W_{14}^{b}$. It sends these public parameters $\{N, U_{1i,4}, X_{14}, W_{14}, g_4\}_{i\in[1,n]}$ to $\mathscr{A}$.

Query Phase 1: When $\mathscr{A}$ requests the secret credential for the medical staff with role $\overrightarrow{R} = (R_1, \ldots, R_d)$, $\mathscr{B}$ lets $z_k = \sum_{i\in I} a_i R_i + b$, randomly chooses exponents t, y, y', r_3, r_3', $\{r_j\}_{j\in([1,n]/I)}, \{y_j\}_{j\in([1,n]/I)}\xleftarrow{R}Z_N$ where $I = \left\{i\colon R_i \in S_{\overrightarrow{R}}\right\}$, and sets

$$\begin{aligned}K_1 &= E_{12} g_2^{y} g_3^{r_3}, \\ K_2 &= E_{12}^{t\cdot z_k} g_2^{y'} g_3^{r_3'}, \\ E_j &= \left\{E_{12}^{t\cdot a_j} g_3^{r_j} g_2^{y_j}\right\}_{j\in([1,n]/I)}.\end{aligned} \tag{27}$$

Consider $E_{12} = g_1^{e_1} g_2^{e_2}$ for random $e_1, e_2\xleftarrow{R}Z_N$, and get

$$\begin{aligned}K_1 &= g_1^{te_1} g_2^{te_2+y} g_3^{r_3}, \\ K_2 &= \left(\prod_{i\in I} U_{1i}^{R_i}\cdot X_1\right)^{te_1} \cdot g_2^{z_k te_2 + y'} \cdot g_3^{r_3'}, \\ E_j &= \left\{U_{1,j}^{te_1} \cdot g_2^{te_2\cdot a_j + y_j} \cdot g_3^{r_j}\right\}_{j\in([1,n]/I)}.\end{aligned} \tag{28}$$

This implicitly sets $r = te_1$, $t_2 = te_2 + y$, $t_2' = z_k te_2 + y'$, $U_{1i} = g_1^{a_i}$, $X_1 = g_1^{b}$, and $t_j = te_2\cdot a_j + y_j$. The simulated key is distributed as the semifunctional credential.

Challenge: $\mathscr{A}$ outputs two EMR files EMR_0 and EMR_1, and a challenge access policy P^*. This policy must satisfy that no revealed role in Query Phase 1 was a prefix of its components. $\mathscr{B}$ picks a random coin $\beta\xleftarrow{R}\{0,1\}$ and gives the challenge EMR encapsulation as follows. We denote that $\mathrm{II}^* = \{i\colon R_i^* \in S_{P*}\}$.

(i) $\mathrm{EMR}_{\beta} = 0$.$\mathscr{B}$ randomly chooses $s, x_2'\xleftarrow{R}Z_N$ and sets

$$\mathrm{En}_1\xleftarrow{R}\left(\prod_{i\in \mathrm{II}^*} U_{1i,4}^{R_i}\cdot X_{14}\right)^s T^{x_2'}, \mathrm{En}_2\xleftarrow{R}W_{14}^{s}T. \tag{29}$$

(a) If $\mathscr{B}$'s challenge bit is $\beta = 1$, then $T\xleftarrow{R}G_{p_1p_2p_4}$. Hence, the challenge ciphertexts En_1 and En_2 are random in $G_{p_1p_2p_4}$ as in $\mathrm{Game}_{\mathrm{final}'}$.

(b) If $\mathscr{B}$'s challenge bit is $\beta = 0$, then $T\xleftarrow{R}G_{p_2p_4}$. We write $T = g_4^{t_4'} g_2^{x_2}$ for random $t_4', x_2\xleftarrow{R}Z_N$ and get

$$\mathrm{En}_1 = \left(\prod_{i\in \mathbb{I}^*} U_{1i,4}^{R_i}\cdot X_{14}\right)^s \cdot g_4^{t_4' x_2'} g_2^{x_2 x_2'}, \mathrm{En}_2 = W_{14}^s g_4^{t_4'} g_2^{x_2}. \tag{30}$$

(c) This implicitly sets $t_4 = t_4' x_2' \bmod p_4$. The challenge encapsulation is formed as the semifunctional form in Game_q.

(ii) $\mathrm{EMR}_{\beta} = 1$. $\mathscr{B}$ sets $\mathrm{En}_1, \mathrm{En}_2\xleftarrow{R}\mathbb{G}$.

Query Phase 2: Query Phase 1 is repeated adaptively except $\overrightarrow{R} \notin \mathrm{Pref}(P^*)$.

Guess: the adversary $\mathscr{A}$ outputs a guess that it is in Game_q or $\mathrm{Game}_{\mathrm{final}'}$.

The simulator $\mathscr{B}$ guesses $T\xleftarrow{R}G_{p_2p_4}$ if $\mathscr{A}$ decides it is in Game_q ($\beta = 0$). Otherwise, $\mathscr{B}$ outputs $T\xleftarrow{R}G_{p_1p_2p_4}$ ($\beta = 1$). If $\mathscr{A}$ has an advantage ε_4 to distinguish Game_q and $\mathrm{Game}_{\mathrm{final}'}$, $\mathscr{B}$ breaks Assumption 4 with advantage ε_4. Since all the credentials and EMR encapsulations are semifunctional in Game_q, $\mathscr{A}$ cannot get any information about the challenge EMR encapsulation due to none of the given credentials are useful to decapsulate it. Hence, $\mathscr{A}$ cannot find that the challenge EMR encapsulation has been replaced by a random component. This implies the indistinguishability between Game_q and $\mathrm{Game}_{\mathrm{final}'}$. □

Lemma 6. *Suppose there exists a PPT algorithm $\mathscr{A}$ such that $|Game_{final'}Adv_{\mathscr{A}}(\lambda) - Game_{final}Adv_{\mathscr{A}}(\lambda)| = \varepsilon_5$. Then we construct a PPT algorithm $\mathscr{B}$ with advantage ϵ_5 in breaking Assumption 5.*

Proof of Lemma 6. The input of $\mathscr{B}$ is the challenge tuple (g_1, g_4, D_{23}, T) of Assumption 5. $\mathscr{B}$ has to answer T is in $G_{p_1p_2p_4}$ or in G.

Setup: $\mathscr{B}$ first receives g_1, g_4, D_{23}, T. It then picks up random exponents $\{a_i\}_{i\in[1,n]}, b$ from Z_N, and sets $U_{1i}\xleftarrow{R}g_1^{a_i}$, $X_1\xleftarrow{R}g_1^b, X_4\xleftarrow{R}g_4^x$, $U_4\xleftarrow{R}g_4^y$, $W_4\xleftarrow{R}g_4^{\omega}$, $U_{1i,4}\xleftarrow{R}U_{1i}U_4$, $X_{14}\xleftarrow{R}X_1X_4$, $W_{14}\xleftarrow{R}g_1W_4$. It sends the public parameters $\{N, U_{1i,4}, X_{14}, W_{14}, g_4\}_{i\in[1,n]}$ to $\mathscr{A}$.

Query Phase 1: when $\mathscr{A}$ requests a secret credential for a medical staff with role $\overrightarrow{R} = (R_1, \ldots, R_d)$, $\mathscr{B}$ lets $z_k = \sum_{i\in I}a_iR_i + b$, chooses exponents r, d, d', $\{d_j\}_{j\in([1,n]/I)}\xleftarrow{R}Z_N$ where $I = \{i\colon R_i \in S_{\overrightarrow{R}}\}$, and sets

$$\begin{aligned}K_1 &= g_1^rD_{23}^d,\\ K_2 &= \left(\prod_{i\in I}U_{1i}^{R_i}\cdot X_1\right)^rD_{23}^{\dot{d}},\\ E_j &= \left\{U_{1j}^rD_{23}^{d_j}\right\}_{j\in([1,n]/I)}.\end{aligned} \tag{31}$$

We write $D_{23} = g_2^{e_2}g_3^{e_3}$ for random $e_2, e_3\xleftarrow{R}Z_N$ and get

$$\begin{aligned}K_1 &= g_1^rg_2^{e_2d}g_3^{e_3d},\\ K_2 &= \left(\prod_{i\in I}U_{1i}^{R_i}\cdot X_1\right)^rg_2^{e\dot{d}}\cdot g_3^{e_3\dot{d}},\\ E_j &= \left\{U_{1j}^rg_3^{e_3d_j}\cdot g_2^{e_2d_j}\right\}_{j\in i\in([1,n]/I)}.\end{aligned} \tag{32}$$

This implicitly sets $r_3 = e_3d$, $t_2 = e_2d$, $r_3' = e_3d\prime$, $t_2' = e_2d\prime$, $r_j = e_3d_j$, and $t_j = e_2d_j$. The simulated credential is distributed as the semifunctional credential.

Challenge: $\mathscr{A}$ outputs two EMR files EMR_0 and EMR_1, and a challenge access policy P^*. This policy must satisfy that no revealed role in Query Phase 1 was a prefix of its components. $\mathscr{B}$ picks up a random exponent z and a random coin $\beta\xleftarrow{R}\{0,1\}$. It gives the challenge encapsulation: $\mathrm{En}_1\xleftarrow{R}T, \mathrm{En}_2\xleftarrow{R}T^z$. If $\mathscr{B}$'s challenge bit is $\beta = 0$ then $T\xleftarrow{R}G$. Hence, the challenge ciphertexts En_1 and En_2 are random components in $\mathbb{G}$ as in $\mathrm{Game}_{\mathrm{final}}$.

If $\mathscr{B}$'s challenge bit is $\beta = 1$, then $T\xleftarrow{R}G_{p_1p_2p_4}$. Hence, the challenge ciphertexts En_1 and En_2 are random components in $G_{p_1p_2p_4}$ as in $\mathrm{Game}_{\mathrm{final}'}$.

Query Phase 2: repeat Query phase 1 except $\overrightarrow{R} \notin \mathrm{Pref}(P^*)$.

Guess: the adversary $\mathscr{A}$ outputs a guess whether it is in $\mathrm{Game}_{\mathrm{final}'}$ or in $\mathrm{Game}_{\mathrm{final}}$. The simulator $\mathscr{B}$ guesses $T\xleftarrow{R}G_{p_1p_2p_4}$ if $\mathscr{A}$ decides it is in $\mathrm{Game}_{\mathrm{final}'}$ $(\beta = 1)$. Otherwise, $\mathscr{B}$ outputs $T\xleftarrow{R}G$ $(\beta = 0)$. If $\mathscr{A}$ has the advantage ε_5 to distinguish $\mathrm{Game}_{\mathrm{final}'}$ and $\mathrm{Game}_{\mathrm{final}}$, $\mathscr{B}$ can break Assumption 5 with advantage ε_5. $\mathrm{Game}_{\mathrm{final}}$ replaces the challenge encapsulation of 0 by a pair of random points in the full group. From the view of adversary, it cannot find that the challenge EMR encapsulation has been replaced by a random component in the full group or in the subgroup. Hence, it implies the indistinguishability between $\mathrm{Game}_{\mathrm{final}'}$ and $\mathrm{Game}_{\mathrm{final}}$. □

Proof of Theorem 1. If a group generator algorithm $\mathbb{G}$ satisfies Assumption i with advantage $\varepsilon_i'(1\leq i\leq 5)$, then Lemmas 0–5 show that there is no polynomial time adversary to distinguish $\mathrm{Game}_{\mathrm{real}}$ and $\mathrm{Game}_{\mathrm{final}}$ with advantage $|\mathrm{Game}_{\mathrm{real}}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_{\mathrm{final}}\mathrm{Adv}_{\mathscr{A}}(\lambda)|$, which can be expanded as follows:

$$\begin{aligned}&\left|\mathrm{Game}_{\mathrm{real}}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_{\mathrm{final}}\mathrm{Adv}_{\mathscr{A}}(\lambda)\right|\\ &\quad\leq\left|\mathrm{Game}_{\mathrm{real}}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_{\mathrm{real}'}\mathrm{Adv}_{\mathscr{A}}(\lambda)\right| + \left|\mathrm{Game}_{\mathrm{real}'}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_{\mathrm{res}}\mathrm{Adv}_{\mathscr{A}}(\lambda)\right|\\ &\qquad+\left|\mathrm{Game}_{\mathrm{res}}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_0\mathrm{Adv}_{\mathscr{A}}(\lambda)\right| + \left|\mathrm{Game}_0\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_1\mathrm{Adv}_{\mathscr{A}}(\lambda)\right|\\ &\qquad+\left|\mathrm{Game}_1\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_2\mathrm{Adv}_{\mathscr{A}}(\lambda)\right| + \cdots + \left|\mathrm{Game}_{\mathrm{q}-1}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_{\mathrm{q}}\mathrm{Adv}_{\mathscr{A}}(\lambda)\right|\\ &\qquad\cdot\left|\mathrm{Game}_{\mathrm{q}}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_{\mathrm{final}'}\mathrm{Adv}_{\mathscr{A}}(\lambda)\right| + \left|\mathrm{Game}_{\mathrm{final}'}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_{\mathrm{final}}\mathrm{Adv}_{\mathscr{A}}(\lambda)\right|\\ &\quad= 3\varepsilon_1' + \varepsilon_2' + \varepsilon_3' + \varepsilon_4' + \varepsilon_5'.\end{aligned} \tag{33}$$

All the components in $\text{Game}_{\text{final}}$ are random elements in $\mathbb{G}$, and the messages are hidden from the adversary. Therefore, if the group $\mathbb{G}$ with composite order $N = p_1p_2p_3p_4$ satisfies Assumption 1–5 with advantage $\varepsilon_1', \varepsilon_2', \varepsilon_3', \varepsilon_4', \varepsilon_5'$ respectively, then our one-bit RBAC is secure with advantage $3\varepsilon_1' + \varepsilon_2' + \varepsilon_3' + \varepsilon_4' + \varepsilon_5'$. □

6.4. From One-Bit RBAC with 1SPL to Multibit RBAC-UL. We provide security analysis for the RBAC-UL model. The key point is to reduce RBAC-UL security from a secured one-bit RBAC with 1SPL functionality. We use a specific 1SPL algorithm "LeakToOne" which exposes the randomness as if it is randomly chosen for bit 1, and fails with probability δ when it cannot find out the randomness to 1. In the security analysis, we assume that all the roles in the access policy set or its subset are ordered from high-level staff to the lower level one.

Theorem 2. *Let Γ be a one-bit RBAC scheme, and Γ^{ℓ} be the ℓ-bit RBAC scheme built from it. Let k be the number of leaked EMRs and δ be the failing probability of LeakToOne. Suppose there exists an RBAC-UL adversary $\mathscr{A}$, RBAC-UL simulator $\mathscr{S}$, and RBAC adversary $\mathscr{B}$. If Γ is secure with $Adv_{\Gamma,\mathscr{B}}^{RBAC-IND}(\lambda)$, then Γ^{ℓ} is secure with $Adv_{\Gamma^{\ell},\mathscr{S},\mathscr{A}}^{RBAC-UL}(\lambda) \le k\ell \cdot Adv_{\Gamma,\mathscr{B}}^{RBAC-IND}(\lambda) + k\ell \cdot \delta$.*

We prove it by a series of game transitions.

(i) $\text{Game}_{\text{real}}^{\mathscr{A}}$: this is the real game.

(ii) $\text{Game}_{\text{real1}}^{\mathscr{A}}$: this game reselects the randomness for the "0" bit encapsulation at Corrupt phase.

(iii) $\text{Game}_{\text{real2}}^{\mathscr{A}}$: this game runs LeakToOne algorithm for the "0" bit encapsulation at Corrupt phase. If LeakToOne fails, it reselects the randomness as $\text{Game}_{\text{real1}}^{\mathscr{A}}$ does.

(iv) $\text{Game}_{\text{real3}}^{\mathscr{A}}$: compared with $\text{Game}_{\text{real2}}^{\mathscr{A}}$, it does nothing if LeakToOne algorithm fails.

(v) $\text{Game}_{v}^{\mathscr{A}}$: the first v bits from the challenge EMRs are replaced by bit "0" and then encapsulated. The remaining bits are encapsulated normally. At Corrupt phase, if it needs to open an encapsulation component to a "0" bit, it directly gives $\mathscr{A}$ the randomness it used when creating the encapsulation. If it needs to open an encapsulation to a "1" bit, it runs LeakToOne to find the randomness.

(vi) $\text{Game}_{k\ell}^{\mathscr{A}}$: all the bits from challenge EMRs are replaced by "0" and all the "0" bits are encapsulated.

(vii) $\text{Game}_{\text{sim}}^{\mathscr{S}}$: this game is run by the simulator $\mathscr{S}$.

In the next subsection, we show that no PPT algorithm can distinguish between $\text{Game}_{\text{real}}^{\mathscr{A}}$ and $\textit{Game}_{\text{real1}}^{\mathscr{A}}$ and between $\text{Game}_{\text{real1}}^{\mathscr{A}}$ and $\text{Game}_{\text{real2}}^{\mathscr{A}}$. Then we demonstrate that, if any execution of LeakToOne fails at most δ, no PPT algorithm has advantage $k\ell\delta$ to distinguish between $\text{Game}_{\text{real2}}^{\mathscr{A}}$ and $\text{Game}_{\text{real3}}^{\mathscr{A}}$. Following that, if no adversary $\mathscr{B}$ has the advantage $\text{Adv}_{\Gamma,\mathscr{B}}^{\text{RBAC-IND}}(\lambda)$ to break one-bit RBAC scheme, then no algorithm has the advantage $\text{Adv}_{\Gamma,\mathscr{B}}^{\text{RBAC-IND}}(\lambda)$ to distinguish between $\text{Game}_{v}^{\mathscr{A}}$ and $\text{Game}_{v+1}^{\mathscr{A}}$, so that no algorithm has the advantage $k\ell \cdot \text{Adv}_{\Gamma,\mathscr{B}}^{\text{RBAC-IND}}(\lambda)$ to distinguish between $\text{Game}_{\text{real3}}^{\mathscr{A}}$ and $\text{Game}_{k\ell}^{\mathscr{A}}$. From the above deductions, we get $|\Pr[\text{Game}_{\text{real}}^{\mathscr{A}}(\lambda) = 1] - \Pr[\text{Game}_{k\ell}^{\mathscr{A}}(\lambda) = 1]| \le k\ell \cdot \text{Adv}_{\Gamma,\mathscr{B}}^{\text{RBAC-IND}}(\lambda) + k\ell \cdot \delta$. We also show that the simulator $\mathscr{S}$ runs identically to the $\text{Game}_{k\ell}^{\mathscr{A}}$, which means the $\text{Game}_{k\ell}^{\mathscr{A}}$ is distinguishable from the $\text{Game}_{\text{sim}}^{\mathscr{S}}$. Finally, we get $\text{Adv}_{\Gamma,\mathscr{S},\mathscr{A}}^{\text{RBAC-UL}}(\lambda) = |\Pr[\text{Game}_{\text{real}}^{\mathscr{A}}(\lambda) = 1] - \Pr[\text{Game}_{\text{sim}}^{\mathscr{S}}(\lambda) = 1]|$ which is defined in Definition 6.

6.5. Proof of Theorem 2. Let $\mathscr{A}$ be a RBAC-UL adversary against Γ^{ℓ}. We can construct a simulator $\mathscr{S}$ that runs Setup to generate PK and MSK. It runs $\mathscr{A}$ to answer the following queries. (1) When $\mathscr{A}$ outputs the set of EMRs on which it wishes to challenge, $\mathscr{S}$ then generates a set of EMR encapsulations where each encryption is an encryption of the all-zero message 0^{ℓ} and returns them to $\mathscr{A}$. (2) When $\mathscr{A}$ decides to corrupt some of these EMR encapsulations, the simulator $\mathscr{S}$ queries its own Corrupt procedure, learns the EMRs it needs to corrupt, and opens them bit-by-bit. If it needs to open an encapsulation component to a 0, it directly gives $\mathscr{A}$ the randomness it used when creating the encapsulation. Otherwise, $\mathscr{S}$ needs to open an encapsulation to a 1, it runs LeakToOne algorithm to find the randomness. (3) When $\mathscr{A}$ issues a secret credential query, $\mathscr{S}$ simply uses MSK to answer correctly. Through the above ways, the simulator $\mathscr{S}$ can generate the same output as $\mathscr{A}$. We use $\text{Game}_{\text{real}}$ and Game_{sim} to describe the games that $\mathscr{A}$ and $\mathscr{S}$ runs, respectively. Based on Definition 6, the target of Theorem 2 is to prove

$$\left|\Pr\left[\text{Game}_{\text{real}}^{\mathscr{A}}(\lambda) = 1\right] - \Pr\left[\text{Game}_{\text{sim}}^{\mathscr{S}}(\lambda) = 1\right]\right| \le k\ell \cdot \text{Adv}_{\Gamma,\mathscr{B}}^{\text{RBAC-IND}}(\lambda) + k\ell \cdot \delta. \quad (34)$$

Lemma 7. *For any PPT algorithm $\mathscr{A}$, $Game_{real}^{\mathscr{A}}Adv_{\mathscr{A}}(\lambda) = Game_{real1}^{\mathscr{A}}Adv_{\mathscr{A}}(\lambda)$.*

Proof of Lemma 7. Since the randomness in $\text{Game}_{\text{real}}^{\mathscr{A}}$ and $\text{Game}_{\text{real1}}^{\mathscr{A}}$ are uniformly and independently chosen from Z_p, they are identically distributed from the view of $\mathscr{A}$. □

Lemma 8. *For any PPT algorithm $\mathscr{A}$, $Game_{real1}^{\mathscr{A}}Adv_{\mathscr{A}}(\lambda) = Game_{real2}^{\mathscr{A}}Adv_{\mathscr{A}}(\lambda)$.*

Proof of Lemma 8. If LeakToOne does not fail, its output is identically distributed as that in $\text{Game}_{\text{real1}}^{\mathscr{A}}$ from the view of $\mathscr{A}$'s. If LeakToOne does fail, $\text{Game}_{\text{real2}}^{\mathscr{A}}$ does the same operations as that in $\text{Game}_{\text{real1}}^{\mathscr{A}}$. □

Lemma 9. *Suppose that any execution of LeakToOne fails at most δ. For any PPT algorithm $\mathscr{A}$, the following holds:*

$$\left|\text{Game}_{\text{real2}}^{\mathscr{A}}\text{Adv}_{\mathscr{A}}(\lambda) - \text{Game}_{\text{real3}}^{\mathscr{A}}\text{Adv}_{\mathscr{A}}(\lambda)\right| \le k\ell \cdot \delta. \quad (35)$$

Proof of Lemma 9. Since there are at most $k\ell$ bits that have to be opened by LeakToOne algorithm in phase Corrupt, the worst event is that all the $k\ell$ bits are failed to be opened and

$\mathrm{Game}^{\mathscr{A}}_{\mathrm{real3}}$ does not make any response to the failure. The probability for the worst event is $k\ell \cdot \delta$. □

Lemma 10. *If a PPT adversary $\mathscr{B}$ can break the one-bit RBAC-IND scheme with $Adv^{RBAC-IND}_{\Gamma,\mathscr{B}}(\lambda)$, then there exists a PPT algorithm $\mathscr{A}$ so that*

$$\left|\mathrm{Game}^{\mathscr{A}}_{v}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}^{\mathscr{A}}_{v+1}\mathrm{Adv}_{\mathscr{A}}(\lambda)\right| = Adv^{\mathrm{RBAC-IND}}_{\Gamma,\mathscr{B}}(\lambda). \tag{36}$$

Proof of Lemma 10. Suppose $\mathrm{Ev}^{1}_{v+1}(\mathrm{Game}^{\mathscr{A}}_{v})$ is the event that in the execution of $\mathrm{Game}^{\mathscr{A}}_{v}$, the $(v+1)$st bits sampled from the set of EMR are 1.

Let $\mathrm{Game}^{\mathscr{A}}_{v}|\mathrm{Ev}^{1}_{v+1}(\mathrm{Game}^{\mathscr{A}}_{v})$ denote that $\mathrm{Game}^{\mathscr{A}}_{v}$ is run in the condition that the event $\mathrm{Ev}^{1}_{v+1}(\mathrm{Game}^{\mathscr{A}}_{v})$ happens. Ev^{0} is the complementary event for Ev^{1}. Notice that in the event that the $(v+1)$st bit sampled in $\mathrm{Game}^{\mathscr{A}}_{v}$ is 0, the game $\mathrm{Game}^{\mathscr{A}}_{v+1}$ and $\mathrm{Game}^{\mathscr{A}}_{v}$ are identical, because $\mathrm{Game}^{\mathscr{A}}_{v+1}$ ignores the actual bit and encapsulates it as a bit 0 based on its definition, and $\mathrm{Game}^{\mathscr{A}}_{v}$ encapsulates the actual $(v+1)$st bit which is 0. Thus, $\mathrm{Ev}^{0}_{v+1}(\mathrm{Game}_{v+1}) = \mathrm{Ev}^{0}_{v+1}(\mathrm{Game}_{v})$. Next, we compute $|\mathrm{Game}^{\mathscr{A}}_{v}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}^{\mathscr{A}}_{v+1}\mathrm{Adv}_{\mathscr{A}}(\lambda)|$ as follows (we use G to represent Game):

$$\begin{aligned}
&\left|\mathrm{Game}^{\mathscr{A}}_{v}\mathrm{Adv}_{\mathscr{A}} - \mathrm{Game}^{\mathscr{A}}_{v+1}\mathrm{Adv}_{\mathscr{A}}\right| \\
&= \Big|\left(\Pr\left[G^{\mathscr{A}}_{v} \,\&\, \mathrm{Ev}^{0}_{v+1}\left(G^{\mathscr{A}}_{v}\right)\right] + \Pr\left[G^{\mathscr{A}}_{v} \,\&\, \mathrm{Ev}^{1}_{v+1}\left(G^{\mathscr{A}}_{v}\right)\right]\right) \\
&\quad -\left(\Pr\left[G^{\mathscr{A}}_{v+1} \,\&\, \mathrm{Ev}^{0}_{v+1}\left(G^{\mathscr{A}}_{v+1}\right)\right] + \Pr\left[G^{\mathscr{A}}_{v+1} \,\&\, \mathrm{Ev}^{1}_{v+1}\left(G^{\mathscr{A}}_{v+1}\right)\right]\right)\Big| \\
&= \Big|\Big(\Pr\left[\mathrm{Ev}^{0}_{v+1}\left(G^{\mathscr{A}}_{v}\right)\right] \cdot \Pr\left[G^{\mathscr{A}}_{v} \,\middle|\, \mathrm{Ev}^{0}_{v+1}\left(G^{\mathscr{A}}_{v}\right)\right] - \Big|\Pr\left[\mathrm{Ev}^{0}_{v+1}\left(G^{\mathscr{A}}_{v+1}\right)\right] \cdot \Pr\left[G^{\mathscr{A}}_{v} \,\middle|\, \mathrm{Ev}^{0}_{v+1}\left(G^{\mathscr{A}}_{v+1}\right)\right]\Big) \\
&\quad +\Big(\Pr\left[\mathrm{Ev}^{1}_{v+1}\left(G^{\mathscr{A}}_{v}\right)\right] \cdot \Pr\left[G^{\mathscr{A}}_{v} \,\middle|\, \mathrm{Ev}^{1}_{v+1}\left(G^{\mathscr{A}}_{v}\right)\right] - \Pr\left[\mathrm{Ev}^{1}_{v+1}\left(G^{\mathscr{A}}_{v+1}\right)\right] \cdot \Pr\left[G^{\mathscr{A}}_{v+1} \,\middle|\, \mathrm{Ev}^{1}_{v+1}\left(G^{\mathscr{A}}_{v+1}\right)\right]\Big)\Big| \\
&= \Big|\Pr\left[\mathrm{Ev}^{0}_{v+1}\left(G^{\mathscr{A}}_{v}\right)\right] \cdot \Big(\Pr\left[G^{\mathscr{A}}_{v} \,\middle|\, \mathrm{Ev}^{0}_{v+1}\left(G^{\mathscr{A}}_{v}\right)\right] - \Pr\left[G^{\mathscr{A}}_{v+1} \,\middle|\, \mathrm{Ev}^{0}_{v+1}\left(G^{\mathscr{A}}_{v+1}\right)\right]\Big) \\
&\quad +\Big(\Pr\left[\mathrm{Ev}^{1}_{v+1}\left(G^{\mathscr{A}}_{v}\right)\right] \cdot \Pr\left[G^{\mathscr{A}}_{v} \,\middle|\, \mathrm{Ev}^{1}_{v+1}\left(G^{\mathscr{A}}_{v}\right)\right] - \Pr\left[\mathrm{Ev}^{1}_{v+1}\left(G^{\mathscr{A}}_{v+1}\right)\right] \cdot \Pr\left[G^{\mathscr{A}}_{v+1} \,\middle|\, \mathrm{Ev}^{1}_{v+1}\left(G^{\mathscr{A}}_{v+1}\right)\right]\Big)\Big|.
\end{aligned} \tag{37}$$

Since in the event Ev^{0}_{v+1} both $\mathrm{Game}^{\mathscr{A}}_{v+1}$ and $\mathrm{Game}^{\mathscr{A}}_{v}$ are identical, the first item in the above formula is 0. It means that

$$\begin{aligned}
\left|\mathrm{Game}^{\mathscr{A}}_{v}\mathrm{Adv}_{\mathscr{A}} - \mathrm{Game}^{\mathscr{A}}_{v+1}\mathrm{Adv}_{\mathscr{A}}\right| = \Big|&\Pr\left[\mathrm{Ev}^{1}_{v+1}\left(\mathrm{Game}^{\mathscr{A}}_{v}\right)\right] \cdot \Pr\left[\mathrm{Game}^{\mathscr{A}}_{v} \,\middle|\, \mathrm{Ev}^{1}_{v+1}\left(\mathrm{Game}^{\mathscr{A}}_{v}\right)\right] \\
&-\Pr\left[\mathrm{Ev}^{1}_{v+1}\left(\mathrm{Game}^{\mathscr{A}}_{v+1}\right)\right] \cdot \Pr\left[\mathrm{Game}^{\mathscr{A}}_{v+1} \,\middle|\, \mathrm{Ev}^{1}_{v+1}\left(\mathrm{Game}^{\mathscr{A}}_{v+1}\right)\right]\Big|.
\end{aligned} \tag{38}$$

Next, we consider the adversary $\mathscr{B}$ against the one-bit RBAC scheme with 1SPL. $\mathscr{B}$ runs $\mathscr{A}$ while simulating its RBAC-IND environment as in either $\mathrm{Game}^{\mathscr{A}}_{v}|\mathrm{Ev}^{1}_{v+1}(\mathrm{Game}^{\mathscr{A}}_{v})$ or $\mathrm{Game}^{\mathscr{A}}_{v+1}|\mathrm{Ev}^{1}_{v+1}(\mathrm{Game}^{\mathscr{A}}_{v+1})$. Note that $\mathrm{Challenge}_{\mathrm{B}}$ is used to denote that the adversary $\mathscr{B}$ runs the Challenge phase in the RBAC-IND experiment.

Setup: $\mathscr{B}$ generates a public key PK by running the system setup algorithm, and it sends PK to $\mathscr{A}$.

Query Phase 1: $\mathscr{A}$ issues a secret credential query for the medical staff associated with role $\overrightarrow{R}$. $\mathscr{B}$ creates the secret credential by running the credential generation algorithm and return the secret credential to $\mathscr{A}$.

Challenge: the adversary $\mathscr{A}$ outputs a set of EMR files $\overrightarrow{\mathrm{EMR}} = \{M_1, M_2, \ldots, M_k\}$ and a challenge access policy set $P^* = \{P^*_1, P^*_2, \ldots, P^*_k\}$ to $\mathscr{B}$. Each challenge access policy P^*_i in the set should satisfy that for all the secret credential queries for $\overrightarrow{R}$ issued in Query Phase 1, $\overrightarrow{R} \notin \mathrm{Pref}(P^*_i)$. We note that each EMR file EMR_i for $i \in [1,k]$ constitutes of ℓ bits since we let the Γ^{ℓ} be a ℓ-bit RBAC scheme. $\mathscr{B}$ randomly chooses $i \xleftarrow{R} [1,\ell]$ and $j \xleftarrow{R} [1,k]$. For each $i \in [1,\ell]$ and $j \in [1,k]$, we consider three cases:

(i) When $(i-1)\cdot\ell + j \le v$, $\mathrm{EMR}[i][j]$ is ignored and replaced by 0. $\mathscr{B}$ randomly chooses $r[i][j] \xleftarrow{R} Z_p$, computes

$\mathrm{En}[i][j] \xleftarrow{R} \mathrm{EMREnc}(\mathrm{PK}, P_i^*, 0, r[i][j])$, and returns it to $\mathscr{A}$.

(ii) When $(i-1)\cdot\ell + j = v+1$, $\mathscr{B}$ encrypts the $v+1$st bit in two different conditions. If $\mathrm{EMR}[i][j] = 1$, $\mathscr{B}$ runs the $\mathrm{Challenge}_{\mathscr{B}}$ algorithm against the RBAC-IND scheme. Specifically speaking, $\mathscr{B}$ flips a random coin to decide whether it encapsulates 0 or 1 under the challenge access policy P_i^*. Then $\mathscr{B}$ returns the cihpertext to $\mathscr{A}$. If $\mathrm{EMR}[i][j] = 0$, EMR $[i][j]$ is encapsulated normally. $\mathscr{B}$ randomly chooses $r[i][j] \xleftarrow{R} Z_p$, executes $\mathrm{En}[i][j] \xleftarrow{R}$ EMREnc (PK, $P_i^*, \mathrm{EMR}[i][j], r[i][j]$) and returns it to $\mathscr{A}$.

(iii) When $(i-1)\cdot\ell + j > v+1$, $\mathrm{EMR}[i][j]$ is encapsulated normally. $\mathscr{B}$ randomly chooses $r[i][j] \xleftarrow{R} Z_p$, computes $\mathrm{En}[i][j] \xleftarrow{R}$ EMREnc ($PK, P_i^*, \mathrm{EMR}[i][j], r[i][j]$), and returns it to $\mathscr{A}$.

Corrupt: $\mathscr{A}$ outputs a set $I \subseteq [1,n]$ and then $\mathscr{B}$ learns $\overrightarrow{\mathrm{EMR}}[I]$. For each index $i \in I$ and each $j \in [1,\ell]$, $\mathscr{B}$ generates the randomness as follows:

(i) If $\mathrm{EMR}[i][j] = 0$, $\mathscr{B}$ returns the actual randomness it used to generated En_i.

(ii) If $\mathrm{EMR}[i][j] = 1$, $\mathscr{B}$ runs LeakToOne algorithm to get randomness under which the EMREnc algorithm applied to 1 would produce $\mathrm{En}_{[i][j]}$, namely,

$$r[i][j] \xleftarrow{R} \text{Leak to one}\left(\mathrm{PK}, P_i^*, \mathrm{En}_{[i][j]}\right). \tag{39}$$

Finally, $\mathscr{B}$ returns $(M[i][j], r[i][j])_{i\in[1,k] j\in[1,\ell]}$ to $\mathscr{A}$.

Query Phase 2: Query Phase 1 is repeated adaptively except that $\overrightarrow{R} \notin \mathrm{Pref}(P_i^*)$.

Output: when $\mathscr{A}$ halts with out, $\mathscr{B}$ halts and outputs $(\overrightarrow{\mathrm{EMR}}, P^*, I, \mathrm{out})$.

The adversary $\mathscr{B}$ only runs $\mathrm{Challenge}_{\mathscr{B}}$ in the event Ev_{v+1}^1 ($\mathrm{EMR}[i][j] = 1$) in the Challenge phase, such that all of its advantage comes from this case. It means that $\Pr[\mathrm{Ev}_{v+1}^1(\mathrm{Game}_v)] = 1$ and $\Pr[\mathrm{Ev}_{v+1}^1(\mathrm{Game}_{v+1})] = 1$. It is important to notice that in the event Ev_{v+1}^1, if $\mathscr{B}$ decides to encapsulate 1, it simulates its environment as in playing $\mathrm{Game}_v^{\mathscr{A}}$ with $\mathscr{A}$. If $\mathscr{B}$ decides to encapsulate 0, it simulates the environment as in $\mathrm{Game}_{v+1}^{\mathscr{A}}$. Therefore, the advantage of $\mathscr{A}$ to distinguish $\mathrm{Game}_v^{\mathscr{A}} | \mathrm{Ev}_{v+1}^1(\mathrm{Game}_v^{\mathscr{A}})$ and $\mathrm{Game}_{v+1}^{\mathscr{A}} | \mathrm{Ev}_{v+1}^1(\mathrm{Game}_{v+1}^{\mathscr{A}})$ depends on the advantage of $\mathscr{B}$ to distinguish that the challenge EMR-encapsulation is for 1 or 0. It is easy to see that

$$\mathrm{Adv}_{\Gamma,\mathscr{B}}^{\mathrm{RBAC-IND}}(\lambda) = \left| \Pr\left[\mathrm{Game}_v^{\mathscr{A}} \mid \mathrm{Ev}_{v+1}^1\left(\mathrm{Game}_v^{\mathscr{A}}\right)\right] - \Pr\left[\mathrm{Game}_{v+1}^{\mathscr{A}} \mid \mathrm{Ev}_{v+1}^1\left(\mathrm{Game}_{v+1}^{\mathscr{A}}\right)\right]\right|. \tag{40}$$

Combined with equation (38), we get

$$\left|\mathrm{Game}_v^{\mathscr{A}}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_{v+1}^{\mathscr{A}}\mathrm{Adv}_{\mathscr{A}}(\lambda)\right| = \mathrm{Adv}_{\Gamma,\mathscr{B}}^{\mathrm{RBAC-IND}}(\lambda). \tag{41}$$

Furthermore, since

$$\left|\mathrm{Game}_0^{\mathscr{A}}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_{k\ell}^{\mathscr{A}}\mathrm{Adv}_{\mathscr{A}}(\lambda)\right| = \sum_{v=0}^{k\ell-1}\left(\left|\mathrm{Game}_v^{\mathscr{A}}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_{v+1}^{\mathscr{A}}\mathrm{Adv}_{\mathscr{A}}(\lambda)\right|\right), \tag{42}$$

we conclude

$$\left|\mathrm{Game}_0^{\mathscr{A}}\mathrm{Adv}_{\mathscr{A}}(\lambda) - \mathrm{Game}_{k\ell}^{\mathscr{A}}\mathrm{Adv}_{\mathscr{A}}(\lambda)\right| = k\ell \cdot \mathrm{Adv}_{\Gamma,\mathscr{B}}^{\mathrm{RBAC-IND}}(\lambda). \tag{43}$$

□

Lemma 11. *For any PPT algorithm* $\mathscr{A}$,

$$\mathrm{Game}_{k\ell}^{\mathscr{A}}\mathrm{Adv}_{\mathscr{A}}(\lambda) = \mathrm{Game}_{\mathrm{sim}}^{\mathscr{S}}\mathrm{Adv}_{\mathscr{S}}(\lambda). \tag{44}$$

Proof of Lemma 11. First, we compare two games $\mathrm{Game}_{k\ell}^{\mathscr{A}}$ and $\mathrm{Game}_{\mathrm{sim}}^{\mathscr{S}}$.

$\mathrm{Game}_{k\ell}^{\mathscr{A}}$ works as follows:

Setup: $\mathscr{A}$ receives a public key from the challenger $\mathscr{C}$.

Query Phase 1: $\mathscr{A}$ issues a secret credential query for the medical staff associated with role $\overrightarrow{R}$. The challenger $\mathscr{C}$ creates the secret credential by running the credential generation algorithm and returns the secret credential to $\mathscr{A}$.

Challenge: the adversary $\mathscr{A}$ outputs a set of EMR files $\overrightarrow{\mathrm{EMR}} = \{\mathrm{EMR}_1\mathrm{EMR}_2, \ldots, \mathrm{EMR}_k\}$ and the challenge access policy set $P^* = \{P_1^*, P_2^*, \ldots, P_k^*\}$ to the challenger $\mathscr{C}$. Each challenge access policy P_i^* in the set should satisfy that for all the access credential queries for $\overrightarrow{R}$ issued in Query Phase 1, $\overrightarrow{R} \notin \mathrm{Pref}(P_i^*)$. $\mathscr{C}$ randomly chooses elements $r_1, r_2, \ldots, r_k \xleftarrow{R} Z_p$ where $p = |\mathbb{G}|$. We denote $\overrightarrow{r} = \{r_1, r_2, \ldots, r_k\}$. $\mathscr{C}$ ignores the input EMRs and regards the components as all-0 messages. Then it encrypts each 0^ℓ message as follows:

$$\mathrm{En}_i \xleftarrow{R} \mathrm{Enc}\left(\mathrm{PK}, P_i^*, 0^\ell, r_i\right), \tag{45}$$

$\mathscr{C}$ returns the set of EMR encapsulation $\overrightarrow{\mathrm{En}} = \{\mathrm{En}_1, \mathrm{En}_2, \ldots, \mathrm{En}_k\}$ to $\mathscr{A}$.

Corrupt: $\mathscr{A}$ outputs a set $I \subseteq [1,n]$ and then $\mathscr{C}$ opens the corresponding ciphertext to get $\overrightarrow{\mathrm{EMR}}[I]$. For each index $i \in I$ and each $j \in [1,\ell]$, $\mathscr{C}$ makes the randomness as follows:

(i) If $\mathrm{EMR}[i][j] = 0$, $\mathscr{C}$ returns the actual randomness it used to generated En_i.

(ii) If $\mathrm{EMR}[i][j] = 1$, $\mathscr{C}$ runs LeakToOne to get the randomness used by EMREnc to compute $\mathrm{En}_{[i][j]}$ when encrypting 1, namely, $r[i][j] \xleftarrow{R}$ Leak to one $(\mathrm{PK}, P_i^*, \mathrm{En}_{[i][j]})$.

Finally, $\mathscr{C}$ returns $(\mathrm{EMR}[i][j], r[i][j])_{i\in[,k], j\in[1,\ell]}$ to $\mathscr{A}$.

Query Phase 2: Query Phase 1 is repeated adaptively except that $\overrightarrow{R} \notin \mathrm{Pref}(P_i^*)$.

Output: When the adversary $\mathscr{A}$ halts with out, $\mathscr{C}$ halts and outputs $(\overrightarrow{\mathrm{EMR}}, P^*, I, \mathrm{out})$.

$\text{Game}^{\mathcal{S}}_{\text{sim}}$ works as follows:

Setup: $\mathcal{S}$ generates the public key PK by running the system setup algorithm and then sends the public key to $\mathcal{A}$.

Query Phase 1: $\mathcal{A}$ requests the secret credential for the medical staff associated with role $\overrightarrow{R}$. $\mathcal{S}$ creates the secret credential by running the credential generation algorithm and return the secret credential to $\mathcal{A}$.

Challenge: $\mathcal{A}$ outputs a set of EMR $\overrightarrow{\text{EMR}} = \{M_1, M_2, \ldots, M_k\}$ and the challenge access policy set $P^* = \{P_1^*, P_2^*, \ldots, P_k^*\}$ to $\mathcal{B}$. Each challenge access policy P_i^* in the set should satisfy that for all the access credential queries for $\overrightarrow{R}$ issued in Query Phase 1, $\overrightarrow{R} \notin \text{Pref}(P_i^*)$. Note that each EMR_i for $i \in [1,k]$ consists of ℓ bits since we let Γ^ℓ be a ℓ-bit RBAC scheme. $\mathcal{S}$ randomly chooses elements $r_1, r_2, \ldots, r_k \overset{R}{\longleftarrow} Z_p$ where $p = |\mathbb{G}|$. We denote $\overrightarrow{r} = \{r_1, r_2, \ldots, r_k\}$. $\mathcal{S}$ ignores the input EMRs and regards the components as all-0 messages. Then it encapsulates each 0^ℓ message as $\text{En}_i \overset{R}{\longleftarrow} \text{Enc}(\text{PK}, \overrightarrow{P_i^*}, 0^\ell, r_i)$. Finally, $\mathcal{S}$ returns the EMR encapsulations $\overrightarrow{\text{En}} = \{\text{En}_1, \text{En}_2, \ldots, \text{En}_k\}$ to $\mathcal{A}$.

Corrupt: $\mathcal{A}$ outputs a set $I \subseteq [1,n]$ and then $\mathcal{S}$ learns $\overrightarrow{\text{EMR}}[I]$. For each index $i \in I$ and each $j \in [1,\ell]$, $\mathcal{S}$ makes the randomness as follows:

(i) If $\text{EMR}[i][j] = 0$, $\mathcal{S}$ returns the actual randomness used to generate En_i.

(ii) If $\text{EMR}[i][j] = 1$, $\mathcal{S}$ runs LeakToOne to get randomness used by EMREnc to compute $\text{En}_{[i][j]}$ for encapsulation of 1, namely, $r[i][j] \overset{R}{\longleftarrow} \text{Leak to one}(\text{PK}, P_i^*, 0^\ell, \text{En}_{[i][j]})$.

Finally, $\mathcal{S}$ returns $(\text{EMR}[i][j], r[i][j])_{i\in[1,k], j\in[1,\ell]}$ to $\mathcal{A}$.

Query Phase 2: Query Phase 1 is repeated adaptively except that $\overrightarrow{R} \notin \text{Pref}(P_i^*)$.

Output: when $\mathcal{A}$ halts with output out, $\mathcal{S}$ halts and outputs $(\overrightarrow{\text{EMR}}, P^*, I, \text{out})$.

$\mathcal{S}$ queries its own Corrupt procedure on I and learns $\overrightarrow{\text{EMR}}[I]$ instead of getting them directly as in $\text{Game}^{\mathcal{A}}_{k\ell}$. From the view of the adversary $\mathcal{A}$, there is no difference of the corrupted EMRs and the sampled randomness. Therefore, $\mathcal{S}$ runs identically with $\text{Game}^{\mathcal{A}}_{k\ell}$. □

Table 4: The efficiency of the proposed scheme.

	The atom roles is n
MSK size	$n+3$
$\text{SC}^{\overrightarrow{R}}$ size	$n+2-\Vert\overrightarrow{R}\Vert$
SCGen time	$(2n-\Vert\overrightarrow{R}\Vert+4)t_e+(n+2)t_m$
SCDeleg time	$(2n-\Vert\overrightarrow{R}\Vert+5)t_e+(n+5)t_m$
EMREnc time	$(\Vert P\Vert+3)t_e+(\Vert P\Vert+2)t_m$
EMRDec time	$2t_p$

Proof of Theorem 2. From the above analysis, the simulator $\mathcal{S}$ described in $\text{Game}^{\mathcal{S}}_{\text{sim}}$ runs identically to $\text{Game}^{\mathcal{A}}_{k\ell}$. So we have $\text{Game}^{\mathcal{A}}_{k\ell}\text{Adv}_{\mathcal{A}}(\lambda) = \text{Game}^{\mathcal{S}}_{\text{sim}}\text{Adv}_{\mathcal{S}}(\lambda)$. Combining all the above lemmas, we get

$$\begin{aligned}
&\left|\text{Game}^{\mathcal{A}}_{\text{real}}\text{Adv}_{\mathcal{A}}(\lambda) - \text{Game}^{\mathcal{S}}_{\text{sim}}\text{Adv}_{\mathcal{S}}(\lambda)\right| \\
&\quad \le \left|\text{Game}^{\mathcal{A}}_{\text{real}}\text{Adv}_{\mathcal{A}}(\lambda) - \text{Game}^{\mathcal{A}}_{\text{real1}}\text{Adv}_{\mathcal{A}}(\lambda)\right| + \left|\text{Game}^{\mathcal{A}}_{\text{real1}}\text{Adv}_{\mathcal{A}}(\lambda) - \text{Game}^{\mathcal{A}}_{\text{real2}}\text{Adv}_{\mathcal{A}}(\lambda)\right| \\
&\qquad + \left|\text{Game}^{\mathcal{A}}_{\text{real2}}\text{Adv}_{\mathcal{A}}(\lambda) - \text{Game}^{\mathcal{A}}_{\text{real3}}\text{Adv}_{\mathcal{A}}(\lambda)\right| + \sum_{\nu=0}^{k\ell-1}\left(\left|\text{Game}^{\mathcal{A}}_{\nu}\text{Adv}_{\mathcal{A}}(\lambda) - \text{Game}^{\mathcal{A}}_{\nu+1}\text{Adv}_{\mathcal{A}}(\lambda)\right|\right) \\
&\qquad + \left|\text{Game}^{\mathcal{A}}_{k\ell}\text{Adv}_{\mathcal{A}}(\lambda) - \text{Game}^{\mathcal{S}}_{\text{sim}}\text{Adv}_{\mathcal{S}}(\lambda)\right| \le k\ell\cdot\delta + k\ell\cdot\text{Adv}^{\text{RBAC-IND}}_{\Gamma,\mathcal{B}}(\lambda).
\end{aligned} \tag{46}$$

According to Definition 6, we get $\text{Adv}^{\text{RBAC-UL}}_{\Gamma^\ell,\mathcal{S},\mathcal{A}}(\lambda) \le k\ell\cdot\text{Adv}^{\text{RBAC-IND}}_{\Gamma,\mathcal{B}}(\lambda) + k\ell\cdot\delta$, which proves Theorem 2. □

7. Performance Analyses

7.1. Improve User Experience. To achieve better user experience, we speed up credential generation and EMR encapsulation by applying online/offine cryptography [38]. The offline phase executes most of heavy computations by assuming a set of random roles, while the online phase only performs light computations to produce the EMR encapsulation and the secret credential once the true roles are available. "Ours&R-BAC" is denoted as the scheme with improved efficiency.

7.2. Theoretical Analysis. Table 4 shows the efficiency of the proposed one-bit RBAC scheme. We denote t_e as one exponent operation time, t_m as one multiplication time, and t_p as one pairing operation time. The maximal depth of the hierarchy for a access policy is $\Vert P\Vert$. $\Vert\overrightarrow{R}\Vert$ is the number of atom roles in a secret credential. In the procedure of SCGen, SCDeleg, EMREnc, and EMRDec, exponentiations can be precomputed by choosing the random exponents.

Table 5 compares several schemes in different perspectives. The properties of scalable sharing, flexible access, and leakage controllability support further rendering our scheme with improved efficiency to practice.

Table 5: Comparison with related work.

	SCGen time	EMREnc time	Parings in EMRDec	Leakage controllability	Scalable sharing	Flexibility
[12]	$(1+4\Vert\vec{R}\Vert)t_e\,(1+4\Vert\vec{R}\Vert)t_m$	$6t_e+t_m+t_p+t_p$	$3\Vert\vec{R}\Vert$	×	√	√
[16]	$3\,(n+4)t_e+\,(3\Vert\vec{R}\Vert+4)t_m$	$(\Vert P\Vert+4)t_e+\,(\Vert P\Vert+4)t_m$	3	×	√	√
[29]	$5t_e+3t_m$	$5t_e+3t_m$	2	√	×	×
[39]	$2\,(\Vert\vec{R}\Vert+1)t_e+\Vert\vec{R}\Vert t_m$	$5t_e+\Vert P\Vert t_m+t_p$	2	×	√	√
[40]	$2\,(\Vert\vec{R}\Vert+1)t_e+\Vert\vec{R}\Vert t_m$	$4t_e+t_m+t_p$	$2\Vert\vec{R}\Vert$	×	×	√
Ours	$(2n-\Vert\vec{R}\Vert+4)t_e+\,(n+2)t_m$	$(\Vert P\Vert+3)t_e+\,(\Vert P\Vert+2)t_m$	2	√	√	√
Ours & RBAC	$\Vert\vec{R}\Vert\cdot t_m$	$\Vert P\Vert\cdot t_m$	2	√	√	√

Table 6: A single computation execution time.

	t_e	t_m	t_p
Prime order bilinear group	4.62 ms	0.04 ms	38.56 ms
Composite order bilinear group	130 ms	0.16 ms	148.52 ms

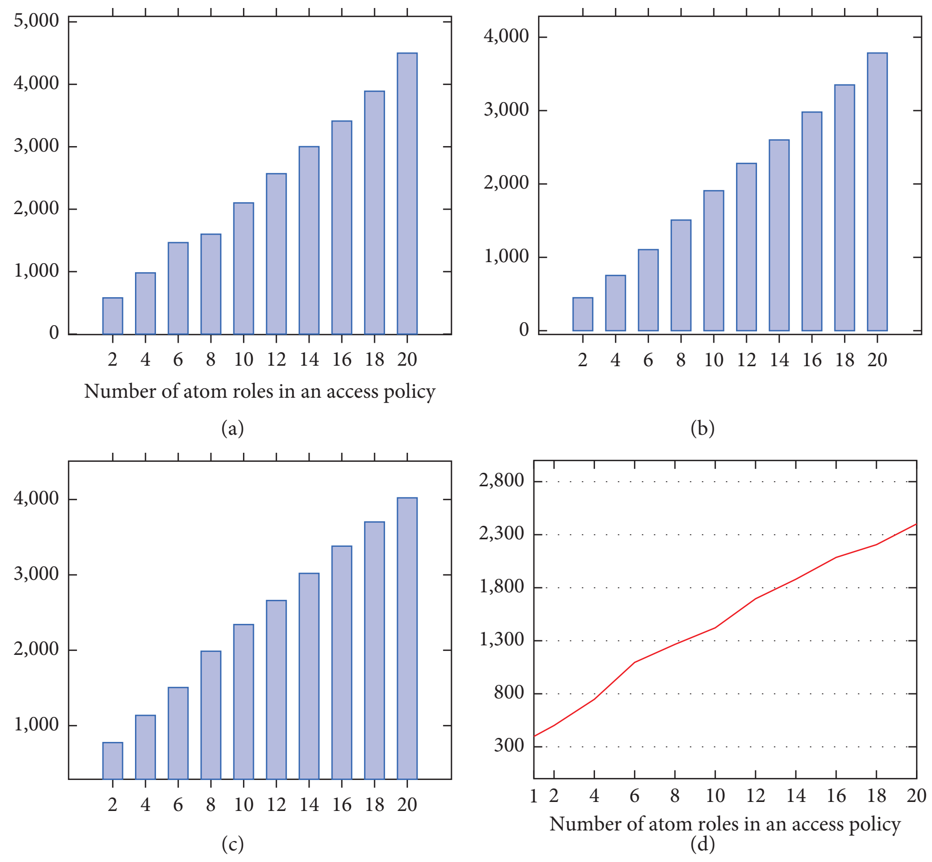
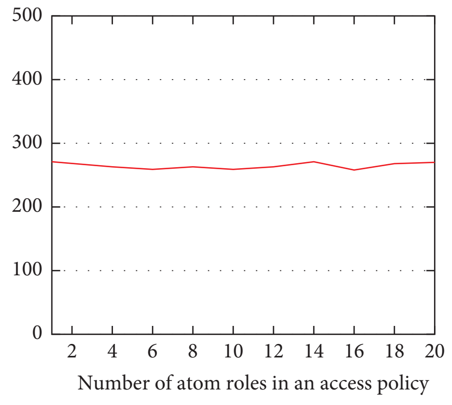
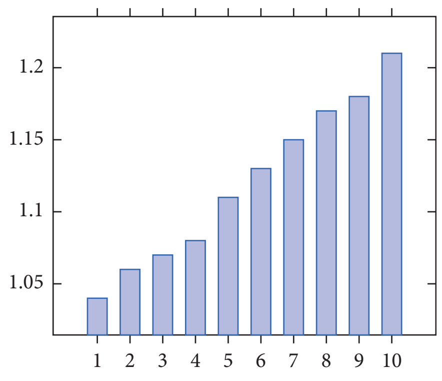


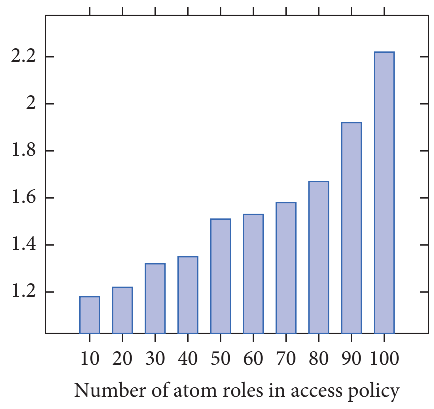
Figure 2: Continued.



(e)

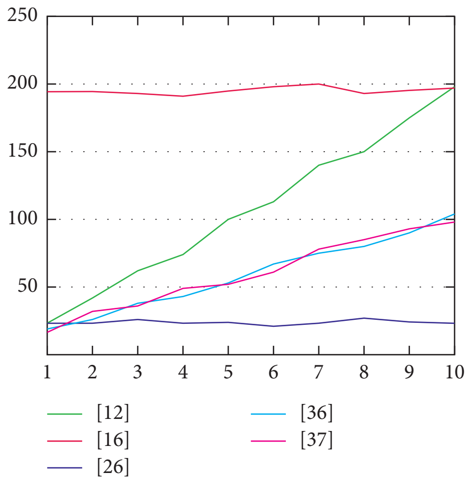


(f)

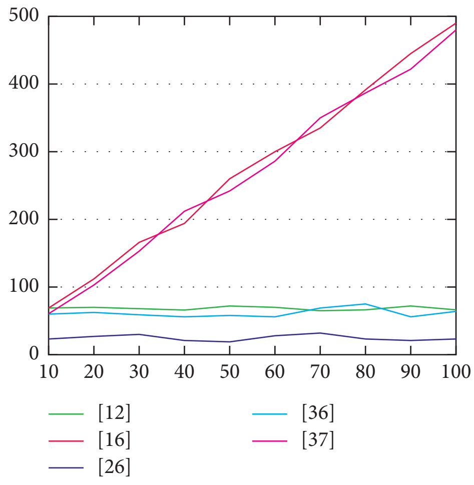


(g)

Figure 2: Experimental results for our proposed system. (a) System setup time (ms). (b) SC generation time (ms). (c) SC delegation time (ms). (d) Encapsulation time (ms). (e) Decapsulation time (ms). (f) Improved SC generation time (ms). (g) Improved encapsulation time (ms).



(a)



(b)

Figure 3: Experimental result for the compared related work. (a) SC Generation time for the related work (ms). (b) EHR Encapsulation time for the related work (ms).

7.3. Performance Analysis. We conduct experiment on Intel Core i7 processor with 8 GB RAM and 2.6 GHZ CPU. We use elliptic curve type A1 for the Tate symmetric pairing. Both group order of $\mathbb{Z}_N$ and element size in $\mathbb{G}$ are configured as 512 bits. The experiment is executed with jPBC library. We test the single computation execution times t_e, t_m, and t_p for the prime order bilinear group and the composite order bilinear group, which are used in the related work and our work separately. Table 6 shows the compared running time.

We also test the operational time for system setup, credential generation, delegation, EMR encapsulation, and decapsulation for our system, as Figure 2 illustrates. Figure 2(f) and 2(g) show the operational time when user experience is improved.

Figures 3(a) and 3(b) show the operational time for the compared related work, where prime order bilinear groups are used. The computation of SC generation time and EHR encapsulation time shows superior efficiency when compared with our work without performance improved. That is why we apply the performance improvement algorithm in our system, so as to ensure both efficiency and security. The Y-axis represents the operational time in milliseconds. The X-axis in Figures 2(b), 2(c), 2(f), and 3(a) means the number of related atom roles included in a role of medical staff. The X-axis in Figures 2(a), 2(d), 2(e), 2(g), and 3(b) means the number of atom roles in an access policy.

8. Conclusion

We consider a multiparty communication scenario in a medical cloud storage system. A lot of medical records are outsourced on the cloud and accessed by medical staff with hierarchical privileges. We summarize different adversarial behaviours and construct a RBAC-UL scheme against many kinds of leakages. Performance analyses show that our scheme has advantages in scalability, flexibility, and the controllability of privacy leakage.

Data Availability

No specific data are available.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key R&D Program of China (2017YFB1400702), the National Natural Science Foundation of China (61972017, 61972018, 61932014), the Beijing Natural Science Foundation (4182033), the National Cryptography Development Fund (MMJJ20180215), and the Special Scientific Research for Civil Aircraft of Ministry of Industry and Information Technology.

References

[1] A. Boldyreva, V. Goyal, and V. Kumar, “Identity-based encryption with efficient revocation,” in *Proceedings of the 15th ACM Conference on Computer and Communications Security CCS 2008*, pp. 417–426, ACM, Alexandria, VA, USA, October 2008.

[2] 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 Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 131–143, 2013.

[3] M. Bellare, A. Desai, and D. Pointcheval, “Relations among notions of security for public-key encryption schemes,” in *CRYPTO’98*, vol. 1462, pp. 26–45, Springer, Berlin, Germany, 1998.

[4] V. T. Hoang, J. Katz, A. O’Neill, and M. Zaheri, “Selective-opening security in the presence of randomness failures,” in *Advances in Cryptology—ASIACRYPT 2016*, vol. 10032, pp. 278–306, Springer, Berlin, Germany, 2016.

[5] B. Mihire and B. Tackmann, “Nonce-based cryptography: retaining security when randomness fails,” in *EUROCRYPT*, vol. 9665, pp. 729–757, Springer, Berlin, Germany, 2016.

[6] M. R. Albrecht and K. G. Paterson, “Lucky microseconds: a timing attack on Amazon’s s2n implementation of TLS,” in *Advances in Cryptology—EUROCRYPT 2016*, vol. 9665, pp. 622–643, Springer, Berlin, Germany, 2016.

[7] M. C. Mont, P. Bramhall, and K. Harrison, “A flexible role-based secure messaging service: exploiting IBE technology for privacy in health care,” in *Proceedings of the International Workshop on Database and Expert Systems Applications*, pp. 432–437, IEEE Computer Society, Prague, Czech Republic, September 2003.

[8] Z. Zhang, H. Wang, and A. V. Vasilakos, “ECG-cryptography and authentication in body area networks,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 6, pp. 1070–1078, 2012.

[9] Z. Qin, K. Feng, S. Hu et al., “A novel identity-based security scheme for wireless sensor networks,” in *Proceedings of the Tenth International Conference on Computational Intelligence and Security*, pp. 662–666, IEEE, Kunming, China, November 2014.

[10] C. C. Tan, H. Wang, S. Zhong, and Q. Li, “Body sensor network security: an identity-based cryptography approach,” in *Proceedings of the ACM Conference on Wireless Network Security*, pp. 148–153, ACM, Alexandria, VA, USA, March 2008.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in *Proceedings of the 13th ACM Conference on Computer and Communications Security CCS 2006*, pp. 89–98, ACM, Incheon, South Korea, October 2006.

[12] Z. Wan, J. E. Liu, and R. H. Deng, “HASBE: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing,” *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 743–754, 2012.

[13] Y. L. Tan, B. M. Goi, R. Komiya, and S. Y. Tan, “A study of attribute-based encryption for body sensor networks,” in *International Conference on Informatics Engineering and Information Science*, vol. 251, pp. 238–247, Springer, Berlin, Germany, 2011.

[14] R. Gandikota, R. Eswara, and J. Appawala, “Fine-grained access control of EHRs in cloud using CP-ABE with user revocation,” *Health and Technology*, vol. 9, no. 4, pp. 487–496, 2019.

[15] C. Guo, R. Zhuang, Y. Jie, Y. Ren, T. Wu, and K.-K. R. Choo, “Fine-grained database field search using attribute-based encryption for e-healthcare clouds,” *Journal of Medical Systems*, vol. 40, no. 11, p. 235, 2016.

[16] W. Liu, X. Liu, J. Liu, Q. Wu, J. Zhang, and Y. Li, “Auditing and revocation enabled role-based access control over outsourced private EHRs,” in *Proceedings of the 2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC 2015)*, pp. 336–341, IEEE, New York, NY, USA, August 2015.

[17] X. Zhou, J. Liu, W. Liu, and Q. Wu, "Anonymous role-based access control on e-health records," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIACCS 2016)*, pp. 559–570, ACM, Xi'an, China, May 2016.

[18] W. Liu, J. Liu, Q. Wu, and B. Qin, "Hierarchical identity-based broadcast encryption," in *Information Security and Privacy*, vol. 8544, pp. 242–257, Springer, Cham, Switzerland, 2014.

[19] L. Qin, V. Varadharajan, and K. Gopinath, "A secure role-based cloud storage system for encrypted patient-centric health records," *The Computer Journal*, vol. 59, no. 11, pp. 1593–1611, 2016.

[20] J. Liu, W. Liu, Q. Wu, and X. Liu, "Auditing revocable privacy-preserving access control for EHRs in clouds," *The Computer Journal*, vol. 60, no. 12, pp. 1125–1132, 2017.

[21] G. Ramu and A. Jayanthi, "Enhancing medical data security in the cloud using RBAC-CPABE and ASS," *International Journal of Applied Engineering Research*, vol. 13, no. 7, pp. 21–25, 2018.

[22] B. Qin and S. Liu, "Leakage-resilient chosen-ciphertext secure public-key encryption from hash proof system and one-time lossy filter," in *Advances in Cryptology—ASIACRYPT 2013*, vol. 8270, pp. 381–400, Springer, Berlin, Germany, 2013.

[23] B. Qin and S. Liu, "Leakage-flexible CCA-secure public-key encryption: simple construction and free of pairing," in *Public-Key Cryptography—PKC 2014*, vol. 8383, pp. 19–36, Springer, Berlin, Germany, 2014.

[24] S. Yilek, "Resettable public-key encryption: how to encrypt on a virtual machine," in *Topics in Cryptology—CT-RSA 2010*, vol. 5985, pp. 41–56, Springer, Berlin, Germany, 2010.

[25] K. G. Paterson, J. C. N. Schuldt, and D. L. Sibborn, "Related randomness attacks for public key encryption," in *Public-Key Cryptography—PKC 2014*, vol. 8383, pp. 465–482, Springer, Berlin, Germany, 2014.

[26] F. Serge, H. Dennis, K. Eike, and W. Hoeteck, "Encryption schemes secure against chosen-ciphertext selective opening attacks," in *EUROCRYPT*, vol. 6110, pp. 381–402, Springer, Berlin, Germany, 2010.

[27] B. Hemenway, B. Libert, R. Ostrovsky, and D. Vergnaud, "Lossy encryption: constructions from general assumptions and efficient selective opening chosen ciphertext security," in *Lecture Notes in Computer Science*, vol. 7073, pp. 70–88, Springer, Berlin, Germany, 2011.

[28] Z. Zhang, S. S. M. Chow, and Z. Cao, "Post-challenge leakage in public-key encryption," *Theoretical Computer Science*, vol. 572, pp. 25–49, 2015.

[29] M. Bellare, B. Waters, and S. Yilek, "Identity-based encryption secure against selective opening attack," in *TCC 2011*, vol. 6597, pp. 321–334, Springer, Berlin, Germany, 2011.

[30] J. Lai, R. H. Deng, S. Liu, J. Weng, and Y. Zhao, "Identity-based encryption secure against selective opening chosen-ciphertext attack," in *Advances in Cryptology—EUROCRYPT 2014*, vol. 8441, pp. 77–92, Springer, Berlin, Germany, 2014.

[31] Y. Chen, Z. Zhang, D. Lin, and Z. Cao, "Generalized (identity-based) hash proof system and its applications," *Security and Communication Networks*, vol. 9, no. 12, pp. 1698–1716, 2016.

[32] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Theory of Cryptography 2005*, vol. 3378, pp. 325–341, Springer, Berlin, Germany, 2005.

[33] C. Gentry, "Practical identity-based encryption without random oracles," in *EUROCRYPT 2006*, pp. 445–464, Springer, Berlin, Germany, 2006.

[34] C. Gentry and S. Halevi, "Hierarchical identity based encryption with polynomially many levels," in *Theory of Cryptography 2009*, vol. 5444, pp. 437–456, Springer, Berlin, Germany, 2009.

[35] J. Horwitz and B. Lynn, "Toward hierarchical identity-based encryption," in *Advances in Cryptology—EUROCRYPT 2002*, vol. 2332, pp. 466–481, Springer, Berlin, Germany, 2002.

[36] R. Canetti, C. Dwork, M., and R. Ostrovsky, "Deniable encryption," *Advances in Cryptology—CRYPTO'97*, vol. 1294, pp. 90–104, Springer, Berlin, Germany, 1997.

[37] A. Naor and B. Waters, "New techniques for dual system encryption and fully secure hibe with short ciphertexts," in *Theory of Cryptography 2010*, vol. 5978, pp. 455–479, Springer, Berlin, Germany, 2010.

[38] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Public-Key Cryptography—PKC 2014*, vol. 8383, pp. 293–310, Springer, Berlin, Germany, 2014.

[39] L.-Y. Yeh, P.-Y. Chiang, Y.-L. Tsai, and J.-L. Huang, "Cloud-based fine-grained health information access control framework for lightweight IoT devices with dynamic auditing and attribute revocation," *IEEE Transactions on Cloud Computing*, vol. 6, no. 2, pp. 532–544, 2018.

[40] M. Barua, X. Liang, R. Lu, and X. Shen, "ESPAC: enabling security and patient-centric access control for ehealth in cloud computing," *International Journal of Security and Networks*, vol. 6, no. 2-3, pp. 67–76, 2011.