# RS-HABE: Revocable-Storage and Hierarchical Attribute-Based Access Scheme for Secure Sharing of e-Health Records in Public Cloud

Jianghong Wei, Xiaofeng Chen, *Senior Member, IEEE*, Xinyi Huang, Xuexian Hu, and Willy Susilo, *Senior Member, IEEE*

**Abstract**—Personal e-health records (EHR) enable medical workers (e.g., doctors and nurses) to conveniently and quickly access each patient's medical history through the public cloud, which greatly facilitates patients' visits and makes telemedicine possible. Additionally, since EHR involve patients' personal privacy information, EHR holders would hesitate to directly outsource their data to cloud servers. A natural and favorite manner of conquering this issue is to encrypt these outsourced EHR such that only authorized medical workers can access them. Specifically, the ciphertext-policy attribute-based encryption (CP-ABE) supports fine-grained access over encrypted data and is considered to be a perfect solution of securely sharing EHR in the public cloud. In this paper, to strengthen the system security and meet the requirement of specific applications, we add functionalities of user revocation, secret key delegation and ciphertext update to the original ABE, and propose a revocable-storage hierarchical attribute-based encryption (RS-HABE) scheme, as the core building of establishing a framework for secure sharing of EHR in public cloud. The proposed RS-HABE scheme features of *forward security* (a revoked user can no longer access previously encrypted data) and *backward security* (a revoked user also cannot access subsequently encrypted data) simultaneously, and is proved to be selectively secure under a complexity assumption in bilinear groups, without random oracles. The theoretical analysis indicates that the proposed scheme surpasses existing similar works in terms of functionality and security, at the acceptable cost of computation overhead. Moreover, we implement the proposed scheme and present experiments to demonstrate its practicability.

**Index Terms**—Personal e-health records, fine-grained access control, attribute-based encryption, secret key delegation, forward and backward security

## 1 Introduction

The rapid development of technologies on cloud computing and big data has brought great changes in the medical field. Specifically, the emergence and wide use of electronic health records (EHR) enable medical workers (such as doctors and nurses) to conveniently and quickly access each patient's medical history through the internet. Compared with the traditional management manner of medical data, EHR greatly improve the treatment efficiency, which is especially important in medical emergencies. Additionally, the digitization of personal health information results in tremendous amounts of medical data, from which some invaluable information that cannot be obtained by reading individual medical data can be disclosed by employing the data mining technology, such as flu forecast and pathology disclosure. Due to the serious challenges introduced by the storage, access and management of a large collection of EHR data, various EHR based healthcare services are popularly outsourced to external cloud computing service providers, such as Microsoft HealthVault and IBM Watson Health.

Although EHR can benefit patients and medical workers by providing high-quality of medical diagnosis and services, the EHR owner has to be faced with the serious risk of EHR data exposure, since he/she completely loses the control of the data when it is outsourced to cloud servers. These security and privacy issues will impede the wide adoption of EHR based heathcare services. While there have been many legislations focusing on the preservation of personal health information in countries around the world, this cannot present malicious cybercriminals from attacking those EHR cloud servers for the high value of EHR data. For example, it was reported that a database containing sensitive EHR of half a million children, including the names of these children and

- *J. Wei is with the State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, Shaanxi 710126, China, and the State Key Laboratory of Mathematical Engineering and Advanced Computing, PLA Strategic Support Force Information Engineering University, Zhengzhou, Henan 450001, China. E-mail: jianghong.wei.xxgc@gmail.com.*
- *X. Chen is with the State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, Shaanxi 710126, China, and the State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China. E-mail: xfchen@xidian.edu.cn.*
- *X. Huang is with the School of Mathematics and Computer Science, Fujian Normal University, Fuzhou 350108, China. E-mail: xyhuang81@gmail.com.*
- *X. Hu is with the State Key Laboratory of Mathematical Engineering and Advanced Computing, PLA Strategic Support Force Information Engineering University, Zhengzhou, Henan 450001, China. E-mail: xuexian_hu@hotmail.com.*
- *W. Susilo is with Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia. E-mail: wsusilo@uow.edu.au.*

*Manuscript received 3 Aug. 2018; revised 15 Sept. 2019; accepted 11 Oct. 2019. Date of publication 21 Oct. 2019; date of current version 1 Sept. 2021. (Corresponding author: Xiaofeng Chen.)*
*Digital Object Identifier no. 10.1109/TDSC.2019.2947920*

their parents, social security numbers, phone numbers, addresses and health issues, was popularly sold on the dark web [1]. Therefore, to make patients actively share their EHR data in the context of public cloud, it is absolutely necessary to place patient-centric access control mechanisms over their EHR data in semi-trusted cloud servers.

A feasible and promising approach of solving the above problem is to outsource the encrypted EHR data to cloud servers such that only those authorized medical workers can access it. Recently, attribute-based encryption (ABE) [2] has been widely employed to realize secure sharing of EHR data in cloud computing. This is mainly because that ABE can preserve patient privacy and achieve fine-grained access control over the encrypted EHR data. Specifically, an EHR owner is allowed to define his/her own access policy, and then embed it into the ciphertext of the EHR data by calling a (ciphertext-policy) ABE scheme. Consequently, only those medical workers with attributes satisfying the defined access policy can access the plaintext of the EHR data by decrypting the corresponding ciphertext. Even cloud servers cannot acquire their data in the absence of their authorization.

So far, there have been a few works focusing on utilizing the variants of ABE to preserve the security of EHR data outsourced to cloud servers. For example, Li et al. [3] employed multi-authority key-policy ABE with attribute revocation to achieve secure sharing of EHR in cloud computing. Lu et al. [4] combined ABE with scalar product computation protocol to put forward a framework for securing access of EHR data in medical emergencies. Liu et al. [5] used ciphertext-policy attribute-based signcryption to guarantee the confidentiality and integrity of EHR data stored in cloud servers. Zhou et al. [6] proposed a traceable and revocable multi-authority ABE to strength the security of EHR based medical systems. Au et al. [7] gave a general framework for secure sharing of EHR in cloud systems based on ABE and proxy re-encryption.

In summary, compared with the original ABE, these attribute-based access schemes for secure sharing of the EHR data in cloud computing provide several additional functionalities, such as attribute or user revocation, malicious user trace and ciphertext update, which enable these schemes to perfectly meet specific security requirements of sharing EHR data in public cloud. Nevertheless, we observe that there are still three important issues that are not well studied or omitted in these schemes.

1) Attribute revocation. In available ABE schemes with the functionality of attribute revocation, a popular revocation manner is to assign a dynamic version key to each attribute. Then, when an attribute of some user is revoked, the attribute authority immediately distributes a new version key of this attribute to those users holding this attribute except for the revoked one. Accordingly, the component of the public key corresponding to this attribute is synchronously updated. However, such a manner cannot resist collusion attack since a malicious unrevoked user might share the new version key with the revoked user. In other words, these ABE schemes in fact fail to achieve attribute revocation in the presence of collusion attacks. Additionally, the attribute authority also needs to independently maintain a secure channel to issue new version keys for each unrevoked user, which will bring heavy workload for the authority.
2) Ciphertext update. When some attributes of a user are revoked, the operation of ciphertext update should be instantly performed to prevent the user from continuing to access EHR data with the original secret key. In the above mentioned schemes, this is done by employing proxy re-encryption scheme. More precisely, each EHR owner would generate a proxy re-encryption key, and delegate the cloud server to update the corresponding ciphertext by running the re-encryption operation over it with the given re-encryption key. Though this update manner makes the EHR data unavailable to the cloud server, each EHR owner has to generate proxy re-encryption keys by frequently calling his/her secret key, which increases the risk of secret key disclosure, as discussed in [8]. Moreover, the high frequency communication between EHR owners and the cloud server would bring large communication overhead, especially when EHR owners run these operations on resource-constrained mobile devices.
3) Secret key delegation. In our daily life, most common institutions are organized in hierarchical manner. For instance, in a hospital, the dean is located at the top of the architecture, and manages several associate deans. Moreover, one of these associate deans may be in charge of leading all medical departments of the hospital, and each department director takes charge of managing the staff. Naturally, from the view of authorization process in ABE, the dean just needs to distribute secret keys to those associate deans according to their attributes, and each associate dean further generates secret keys for his/her direct reports, and so on. In short, due to the feature of hierarchical architecture, an ABE scheme supporting hierarchical secret key delegation is more desirable for secure sharing of EHR data. Nevertheless, aforementioned schemes omit this functionality.

## 1.1 Our Contributions

In this paper, we further study the problem of secure sharing of outsourced EHR data with ABE. We mainly focus on addressing the problems of user revocation, ciphertext update and secret key delegation, which are not well studied or omitted in existing schemes. To this end, we first develop the original ciphertext-policy ABE by adding essential functionalities, and further combine it with symmetric encryption scheme to build a more practical attribute-based access scheme for secure sharing of EHR data in public cloud. Specifically, the main contributions of this paper are summarized as follows.

- We extend the original ciphertext-policy ABE to meet the specific security requirements of sharing EHR data in public cloud, and formalize it as the notion of revocable-storage hierarchical ABE (RS-HABE). Such a scheme captures the functionalities of dynamic user revocation, public ciphertext update and scalable secret key delegation simultaneously. It perfectly conquers the above mentioned issues, and features of forward and backward security.

- Based on (ciphertext-policy) RS-HABE and symmetric encryption, we present a general attribute-based access framework for secure sharing of EHR data outsourced to public cloud. Due to the advantages of RS-HABE, this framework enables each EHR owner to share his/her data securely and efficiently in semi-trusted cloud servers.
- To instantiate the above access framework, we propose a concrete construction of RS-HABE scheme in bilinear groups. We define the security notion of RS-HABE, and prove its selective security under a $q$-type complexity assumption over bilinear groups, without random oracles.
- We theoretically evaluate the performance of the proposed RS-HABE scheme in terms of security and functionality. Besides, we implement the proposed RS-HABE scheme with Charm [9], a framework of prototyping cryptosystems using Python language. The experiments indicates that our scheme achieves preferable practicality.

## 1.2 Related Work

Due to its popular features of preserving user privacy and realizing fine-grained access control of encrypted data, ABE [2], [10] is regarded as one of the most desirable solutions of achieving secure sharing of big data in cloud computing [11]. Motivated by specific security requirements of sharing EHR data outsourced to semi-trusted cloud servers, original ABE schemes are modified and extended to match this scenario well, such as revocable ABE and hierarchical ABE. In addition, there are other well-studied cryptographic primitives that can be employed to guarantee the security of outsourced EHR data, for instance, multi-factor authentication [12], [13] and outsourced computation [14], [15], [16], [17], [18], but they are beyond the scope of this paper, and we omit them here.

### *1.2.1 Revocable ABE*

The mechanism of user and attribute revocation is necessary for an ABE scheme used to achieve secure sharing of EHR data. This will prevent those users, who have left the system or whose attributes/credentials have changed, from continuing to access subsequently encrypted EHR data using the original secret key. Furthermore, to strengthen the system security, those previously encrypted EHR data should also be immediately updated after the revocation operation. As a result, by performing user revocation and ciphertext update simultaneously, the revoked user can no longer access any encrypted EHR data, even if his/her attributes still satisfy the defined access policy.

Narayan et al. [19] combined broadcast encryption with ABE to establish privacy preserving EHR systems. Although their scheme fulfills direct user revocation, the communication overhead is linear with the number of unrevoked users. In addition, they didn't consider the issue of ciphertext update after user revocation. Yu et al. [20] utilized key-policy ABE to secure the outsourced data. To achieve dynamic attribute revocation, they issued a version key for each attribute, and generated a new one if this attribute of some user is revoked. Moreover, for those users holding this attribute but has not been revoked, the attribute authority sends an identical update key to them for updating their original secret keys. On the other hand, each EHR owner needs to produce a proxy re-encryption key and delegates semi-trusted cloud servers to re-encrypt previously outsourced EHR data. However, we can see that this kind of attribute revocation manner fails to achieve the intended security goal, since a malicious user might share the same attribute update key with a revoked user. Later, based on the above attribute revocation manner, Li et al. [3] presented a revocable ciphertext-policy ABE to secure the outsourced EHR data in the scenario of multiple owners. In addition, there are some other similar revocable ABE schemes [21], [22] that use the same revocation manner. Consequently, they all suffer from the collusion attack. Notably, Hur and Noh [23] used group key distribution mechanism to achieve attribute revocation.

Attrapadung and Imai [24] initially studied the problem of user revocation in the setting of key-policy ABE, and gave directly/indirectly revocable ABE schemes. Specifically, in the paradigm of direct revocation, an encryptor himself controls the revocation list, which is further embedded into the ciphertext. This method will bring heavy workload for the EHR owner to update the ciphertext when user revocation occurs. Liu et al. [25] recently improved this manner by introducing a secret key time validation technique. In the paradigm of indirect revocation, the authority periodically generates an update key for unrevoked users, to assist them updating their original secret keys. Liang et al. [26] further presented an indirectly revocable ciphertext-policy ABE, Shi et al. [27] proposed a directly revocable key-policy ABE based on multilinear maps. However, the above revocable ABE schemes all don't support ciphertext update. Sahai et al. [8] put forward a generic framework of adding the functionalities of indirect user revocation and ciphertext update into ABE, which is named as revocable-storage ABE. Lee et al. [28], [29] further generalized this notion and presented self-updatable encryption. Zhou et al. [6] extended Sahai et al.'s framework by additionally providing the functionality of traceability. The above two schemes are built upon bilinear groups of composite order, and thus are not efficient enough. To improve revocation efficiency, Cui et al. [30] introduced a server-aided revocable ciphertext-policy ABE, Yang et al. [31] gave the proxy-assisted user revocation approach. In addition, there have been some works [32], [33] focusing on revocable ABE with decryption key exposure resistance.

### *1.2.2 Hierarchical ABE*

The functionality of secrete key delegation is important and desirable for various practical applications. It enables the private key generator to delegate the secret key generation to low-level users. It not only reduces the workload of the private key generator, but also greatly facilitates the users to get their secret keys. For example, the hospital chief can produce all assistant dean's attribute secret keys with his/her own, and each assistant dean is permitted to generate his/

her subordinates' attribute secret keys. This functionality essentially captures the fact that most institutions/corporations and corresponding identities/attributes are organized in a hierarchical manner. On the other hand, existing pairing-based ABE schemes can also provide limited delegation capabilities. That is, for a user that has the secret key corresponding to an attribute set $S$, he/she can produce the secret key for any user that holds an attribute subset $S'$ of $S$. This can be seen as a built-in property of pairing-based (hierarchical) ABE schemes. However, in the setting of hierarchical ABE (HABE), the functionality of secret key delegation is customized for the hierarchical attribute structure.

Wang et al. [34] proposed a HABE scheme by combining a hierarchical identity-based encryption (HIBE) scheme with an ABE scheme. In addition, they further used Yu et al.'s [20] manner to complete attribute revocation. In this HABE scheme, only those users located at the bottom of the architecture hold attributes, and each intermediate user is just assigned to a leveled identity. That is to say, this scheme supports hierarchical identity, rather than hierarchical attribute. Wan et al. [35] gave a hierarchical attribute-set-based encryption scheme, and utilized a natural manner to achieve attribute revocation. Namely, a user's secret key is connected with a key component that corresponds to an expiration time, and those unrevoked users' secret keys are updated by producing key components of new expiration times. However, this revocation method also cannot withstand collusion attack. Moreover, there are a few other similar ABE constructions supporting hierarchical structure of users [36], [37], [38]. Particularly, Deng et al. [39] presented a genuine ciphertext-policy HABE over bilinear groups of composite order. Wei et al. [40] put forward a HABE scheme with forward security. Note that all these schemes miss the functionality of user revocation.

### 1.3 Outline

The remainder of this paper is organized as follows. We introduce preliminaries in Section 2 and present our attribute-based access framework for secure sharing of EHR data in Section 3. In Section 4 we put forward a RS-HABE construction and prove its security in the supplemental file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TDSC.2019.2947920. The performance of the proposed scheme is evaluated in Section 5. We conclude the paper in Section 6.

## 2 Preliminaries

### 2.1 Notations

Throughout this paper, we denote by $\mathbb{Z}_p$ the set of integers modulo $p$, where $p$ is a prime number. For positive integers $n, n_1, n_2$ $(n_1 < n_2)$, denote by $[n]$ the set $\{1, 2, \ldots, n\}$ and $[n_1, n_2]$ the set $\{n_1, n_1+1, \ldots, n_2\}$. Let $x \xleftarrow{\$} \mathbb{Z}_p$ mean the operation of randomly sampling $x$ from $\mathbb{Z}_p$. $\vec{x} = (x_1, \ldots, x_k)$ indicates a row vector of length $k$ and $(\vec{x})^{\mathrm{T}}$ is its transpose. $A_{l\times n}$ means a matrix of size $l \times n$. PPT is short for probabilistic polynomial time.

### 2.2 Complexity Assumption

The security of our RS-HABE construction is built upon the decisional bilinear Diffie-Hellman exponent (BDHE) assumption, which was introduced by Boneh et al. [41] and proved to hold in generic bilinear groups. Specifically, given a security parameter $\lambda$, let $\mathcal{G}(\lambda)$ be an algorithm that generates a bilinear map described as $(\mathbb{G}_1, \mathbb{G}_2, g, p, e)$, where $g$ is a generator of $\mathbb{G}_1$ with prime oder $p$, $e$ is a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let $a, s$ be two integers randomly sampled from $\mathbb{Z}_p$. The decisional $q$-BDHE problem defined over $(\mathbb{G}_1, \mathbb{G}_2, g, p, e)$ is stated as follows: given a vector $\vec{f} = (g, g^a, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}}, g^s)$ and a group element $R \in \mathbb{G}_2$, decide $R = e(g, g)^{a^{q+1}s}$ or $R \xleftarrow{\$} \mathbb{G}_2$. The advantage of an algorithm $\mathcal{C}$ solving the decisional $q$-BDHE problem by outputting a bit $\beta \in \{0, 1\}$ is captured as

$$\mathsf{Adv}_{\mathcal{C}}^{q\text{-BDHE}}(\lambda) = \Big| \Pr\Big[\mathcal{C}\Big(\vec{f}, R = e(g,g)^{a^{q+1}s}\Big) = 0\Big] - \Pr\Big[\mathcal{C}\Big(\vec{f}, R \xleftarrow{\$} \mathbb{G}_2\Big) = 0\Big]\Big|.$$

**Definition 1 (Decisional $q$-BDHE Assumption [41]).** *The decisional q-BDHE assumption holds over $(\mathbb{G}_1, \mathbb{G}_2, g, p, e)$ provided that for any PPT algorithm, its advantage of solving the decisional q-BDHE problem is a negligible function $\mathsf{negl}(\lambda)$ of the security parameter $\lambda$.*

### 2.3 Linear Secret Sharing Schemes

We adopt the concepts of access structures and LSSS introduced by Beimel [42] and slightly modify them to match our setting. As proved in [42], a monotone access structure can be realized by a LSSS, which is defined as follows:

**Definition 2 (LSSS [42]).** *Let $\mathcal{U}$ be the attribute universe and $p$ be a prime. A secret sharing scheme $\Pi$ realizing an access structure $\mathbb{A}$ over $\mathcal{U}$ is said to be linear over $\mathbb{Z}_p$ if*

*(i) The shares of a secret $s \in \mathbb{Z}_p$ derived from $\Pi$ form a vector over $\mathbb{Z}_p$.*

*(ii) There exists a share generation matrix $\boldsymbol{A} \in \mathbb{Z}_p^{l\times n}$ such that for $i = 1$ to $l$, the ith row $A_i$ of $\boldsymbol{A}$ is mapped to an attribute $\rho(i) \in \mathcal{U}$ by a function $\rho : [l] \to \mathcal{U}$. Moreover, given a row vector $\vec{v} = (s, s_2, \ldots, s_n)$, where $s_2, \ldots, s_n$ are randomly sampled from $\mathbb{Z}_p$, then $\boldsymbol{A} \cdot (\vec{v})^{\mathrm{T}}$ is the vector that consists of $l$ shares of the secret $s$ generated according to $\boldsymbol{A}$. For $i \in [l]$, the share $\lambda_i = A_i \cdot (\vec{v})^{\mathrm{T}}$ belongs to the attribute $\rho(i)$.*

The above definition of LSSS enjoys the property of *linear reconstruction*. That is, given a LSSS $(\boldsymbol{A}_{l\times n}, \rho)$ realizing an access structure $\mathbb{A}$, then for any attribute set $\mathsf{S} \in \mathbb{A}$, there exist constants $\{w_i \in \mathbb{Z}_p\}_{i\in I}$ such that $\sum_{i\in I} w_i\lambda_i = s$, where $I = \{i \in [l] | \rho(i) \in \mathsf{S}\}$ and $\{\lambda_i\}_{i\in I}$ are valid shares of $s$. However, for an attribute set $\mathsf{S} \notin \mathbb{A}$, there are no such constants. But, there exists another vector $\vec{\omega} \in \mathbb{Z}_p^n$ such that $\vec{\omega} \cdot (1, 0, \ldots, 0)^{\mathrm{T}} = 1$ and $A_i \cdot (\vec{\omega})^{\mathrm{T}} = 0$ for all $i \in I$.

### 2.4 KUNode Algorithm

We mainly follow Boldyreva et al.'s [43] approach to achieve scalable user revocation by using a binary tree. Specifically, given a complete binary tree $\mathcal{BT}$ with $N$ leaf nodes, denote by $\mathsf{root}$ its root node. Let $\theta_l$ and $\theta_r$ be the left and right child of each non-leaf node $\theta$, respectively. Each user $\mathsf{ID}$ is randomly assigned to a leaf node $\eta$ of $\mathcal{BT}$, and denote by $\mathsf{Path}(\mathsf{ID})$ the set of all nodes over the path from $\mathsf{root}$ to $\eta$.

If the user $\mathsf{ID}$ is revoked at a time period $t$ then the tuple $(\mathsf{ID}, \eta, t)$ is added to a revocation list $RL$.

Moreover, we define an algorithm $\mathsf{KUNode}(\mathcal{BT}, RL, t)$ that generates a minimal node set $\mathsf{Y}$ such that for any tuple $(\mathsf{ID}, \eta, t) \in RL$, it holds that $\mathsf{Path}(\mathsf{ID}) \cap \mathsf{Y} = \emptyset$. On the other hand, for a user $\mathsf{ID}$ not in $RL$, there exists a node $\theta \in \mathsf{Y}$ such that $\theta$ is an ancestor of the leaf node $\eta$ assigned to $\mathsf{ID}$. Below we present the details of this algorithm.

**Algorithm 1.** KUNode

```
1: procedure KUNode($\mathcal{BT}, RL, t$)
2:   X,Y← $\emptyset$
3:   for $(\eta_i, t_i) \in RL$ do
4:     if $t_i \le t$ then
5:       X ← X $\cup$ Path($\eta_i$)
6:   for $\theta \in$ X do
7:     if $\theta_l \notin$ X then
8:       Y ← Y $\cup \{\theta_l\}$
9:     if $\theta_r \notin$ X then
10:      Y ← Y $\cup \{\theta_r\}$
11:  if Y = $\emptyset$ then
12:    Y ← Y $\cup$ {root}
13:  return Y
```

### 2.5 Revocable-Storage Hierarchical ABE

In the context of RS-HABE, each system user holds an attribute set comprising of vectors. More precisely, we denote by $\mathsf{ID}|_k$ a user at level $k$, which means that his/her attribute set $\mathsf{S}_{\mathsf{ID}|_k}$ consists of vectors with length $k$. In addition, each user $\mathsf{ID}|_k$ individually maintains a binary tree $\mathcal{BT}_k$ to manage his/her children. The semantic definition of RS-HABE is captured by the following polynomial-time algorithms:

$\mathsf{Setup}(\lambda, N, T, \ell, \phi) \to (\mathsf{msk}, \mathsf{pp})$: The system setup algorithm takes as input a security parameter $\lambda$, a maximum number $N$ of users at each level, a maximum hierarchical length $\ell$, a total number $T$ of time periods, and a maximum number $\phi$ of attribute categories. The algorithm outputs the public parameter $\mathsf{pp}$ and master secret key $\mathsf{msk}$.

$\mathsf{SKeyGen}(\mathsf{pp}, \mathsf{st}_{\mathsf{ID}|_{k-1}}, \mathsf{S}_{\mathsf{ID}|_k}) \to \mathsf{sk}_{\mathsf{ID}|_k}$: The secret key generation algorithm takes as input $\mathsf{pp}$, the state information $\mathsf{st}_{\mathsf{ID}|_{k-1}}$ of the binary tree $\mathcal{BT}_{\mathsf{ID}|_{k-1}}$[1] maintained by the user $\mathsf{ID}|_{k-1}$, and an attribute set $\mathsf{S}_{\mathsf{ID}|_k}$ comprised of vectors with length $k \le \ell$. It generates a secret key $\mathsf{sk}_{\mathsf{ID}|_k}$ for $\mathsf{S}_{\mathsf{ID}|_k}$.

$\mathsf{UKeyGen}_{\mathsf{AA}}(\mathsf{pp}, \mathsf{msk}, \mathsf{st}_0, RL_0, t) \to \mathsf{uk}_{0,t}$: The attribute authority's update key generation algorithm takes as input $\mathsf{pp}$, $\mathsf{msk}$, the state information $\mathsf{st}_0$ of $\mathcal{BT}_0$, the revocation list $RL_0$ and the time period $t$. It produces an update key $\mathsf{uk}_{0,t}$ for unrevoked users at the first level.

$\mathsf{DKeyGen}_{\mathsf{ID}|_1}(\mathsf{pp}, \mathsf{sk}_{\mathsf{ID}|_1}, \mathsf{uk}_{0,t}) \to \mathsf{dk}^t_{\mathsf{ID}|_1}$: This algorithm is performed by a user $\mathsf{ID}|_1$ at the first level. It takes as input $\mathsf{pp}$, $\mathsf{sk}_{\mathsf{ID}|_1}$ and $\mathsf{uk}_{0,t}$, and derives a decryption key $\mathsf{dk}^t_{\mathsf{ID}|_1}$ for the user $\mathsf{ID}|_1$.

$\mathsf{UKeyGen}_{\mathsf{ID}|_{k-1}}(\mathsf{pp}, \mathsf{dk}^t_{\mathsf{ID}|_{k-1}}, RL_{\mathsf{ID}|_{k-1}}) \to \mathsf{uk}_{\mathsf{ID}|_{k-1},t}$: This algorithm is performed by a user $\mathsf{ID}|_{k-1}$. It takes as input $\mathsf{pp}$, $\mathsf{dk}^t_{\mathsf{ID}|_{k-1}}$ and revocation list $RL_{\mathsf{ID}|_{k-1}}$, and produces an update key $\mathsf{uk}_{\mathsf{ID}|_{k-1},t}$ for unrevoked children of $\mathsf{ID}|_{k-1}$.

1. In the case that $k = 1$, we denote $\mathcal{BT}_{\mathsf{ID}|_{k-1}}$ by $\mathcal{BT}_0$, and $\mathsf{ID}_0$ is the attribute authority.

$\mathsf{DKeyGen}_{\mathsf{ID}|_k}(\mathsf{pp}, \mathsf{sk}_{\mathsf{ID}|_k}, \mathsf{uk}_{\mathsf{ID}|_{k-1},t}) \to \mathsf{dk}^t_{\mathsf{ID}|_k}$: This algorithm is run by a user $\mathsf{ID}|_k$. It takes as input $\mathsf{pp}$, $\mathsf{sk}_{\mathsf{ID}|_k}$ and $\mathsf{uk}_{\mathsf{ID}|_{k-1},t}$, and generates a decryption key $\mathsf{dk}^t_{\mathsf{ID}|_k}$ for the user $\mathsf{ID}|_k$.

$\mathsf{Encrypt}(\mathsf{pp}, m, t, (A, \rho))\mathsf{ct}^t$: The encryption algorithm takes as input $\mathsf{pp}$, a message $m$ to encrypt, the current time period $t$ and a LSSS access structure $(A, \rho)$. It generates a ciphertext $\mathsf{ct}^t$ of $m$.

$\mathsf{CTUpdate}(\mathsf{pp}, \mathsf{ct}^t, t') \to \mathsf{ct}^{t'}$: The ciphertext update algorithm takes as input $\mathsf{pp}$, $\mathsf{ct}^t$ and the next time period $t'$. It updates the ciphertext $\mathsf{ct}^t$ to $\mathsf{ct}^{t'}$ and then erases $\mathsf{ct}^t$.

$\mathsf{Decrypt}(\mathsf{pp}, \mathsf{dk}^t_{\mathsf{ID}|_k}, \mathsf{ct}^t, (A, \rho), t) \to m$: The decryption algorithm takes as input $\mathsf{pp}$, $\mathsf{dk}^t_{\mathsf{ID}|_k}$, $\mathsf{ct}^t$, $(A, \rho)$ and a time period $t$. If $\mathsf{S}_{\mathsf{ID}|_k}$ satisfies the access structure $(A, \rho)$, then the algorithm outputs the message $m$; otherwise it produces an error symbol.

$\mathsf{Revoke}_{\mathsf{ID}|_{k-1}}(\mathsf{st}_{\mathsf{ID}|_{k-1}}, RL_{\mathsf{ID}|_{k-1}}, \mathsf{ID}|_k, t) \to RL'_{\mathsf{ID}|_{k-1}}$: The user revocation algorithm, which is run by $\mathsf{ID}|_{k-1}$, takes as input the state information $\mathsf{st}_{\mathsf{ID}|_{k-1}}$ of $\mathcal{BT}_{\mathsf{ID}|_{k-1}}$ and the current revocation list $RL_{\mathsf{ID}_{k-1}}$, a user identity $\mathsf{ID}|_k$ to be revoked as well as the revocation time period $t$. It outputs an updated revocation list $RL'_{\mathsf{ID}|_{k-1}}$.

## 3 The Framework of EHR Sharing Scheme

### 3.1 Participants

The participants involved in our framework mainly include the following entities:

- *Attribute authority*. The attribute authority is a third party assumed to be totally trusted. It takes charge of managing all users at the first level and the corresponding attributes. More precisely, it maintains the system public parameter and master secret key, and issues secret keys for users located at the first level according to their attributes. In addition, it also needs to periodically publish the update key for unrevoked users at the first level. By delegating the assignment of generating secret keys and update keys of users located at other levels, the workload of the attribute authority greatly decreases.
- *Cloud service provider*. This entity is an honest but curious third party, which is responsible for storing and periodically updating encrypted EHR data. That is, when EHR owners upload their encrypted data to the corresponding servers, the provider may attempt to recover the plaintext of the encrypted EHR data, but would honestly perform the assigned operations, e.g., periodically updating the encrypted EHR data.
- *EHR owner*. Each EHR owner is an entity of holding the corresponding EHR data, which usually contains sensitive information about the holder, such as medical history and home address. As a consequence, the EHR owner would only like to share his/her EHR data with intended medical workers in a secure way. To this end, the EHR owner encrypts the EHR data with a carefully defined access structure, and further outsources the resulted ciphertext to the public cloud server managed by the service provider. This makes that only those medical workers with attributes satisfying the defined access structure can access the EHR data with their decryption keys.

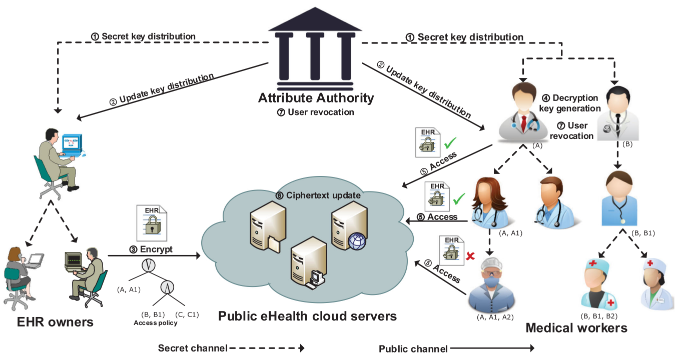


Fig. 1. The framework of secure sharing of EHR in the public cloud.

- *Medical worker*. A medical worker might be a nurse, a doctor or a relevant administrator. Depending on their own permissions, each medical worker possibly has different attributes, which are located at different levels. For example, in the management structure of a hospital, the dean is at the top (the first level) of the architecture and manages all department directors that are at the second level. Moreover, each director is in charge of managing his/her staff at the third level. We can see that the permission of a medical worker goes down with the decrease of the location level in the architecture. In addition, each medical worker needs to produce secret keys and update keys for other workers that are located at the next level and managed by him/her. Particularly, if a medical worker's attribute set or an attribute set that can be derived from his/hers satisfies the access structure of some encrypted EHR data, then this medical worker is allowed to access the EHR data by decrypting the ciphertext.

## 3.2 Framework Overview

As depicted in Fig. 1, the process of the proposed framework is specialized as follows:

– System setup and secret key distribution. Initially, the attribute authority selects an appropriate security parameter $\lambda$, according to which it specializes a symmetric encryption scheme $\mathsf{SE}$ and a RS-HABE scheme $\mathsf{RS-HABE}$. Then, it runs the setup algorithm of $\mathsf{RS-HABE}$ to get the public parameter $\mathsf{pp}$ and the master secret key $\mathsf{msk}$. Moreover, the attribute authority generates secret keys for all users at the first level of the architecture. Meanwhile, each user is also in charge of generating secret keys for those users that are located at the next level and managed by him/her via running the secret key generation algorithm of $\mathsf{RS-HABE}$. All these secret keys are distributed via a secure channel.

– Update key distribution. At the beginning of each time period, the attribute authority issues an update key for all unrevoked users at the first level by running the update key generation algorithm of $\mathsf{RS-HABE}$. After that, in turn, a user at level $k$ generates an update key for his/her unrevoked children that are located at level $k+1$ by calling the update key generation algorithm of $\mathsf{RS-HABE}$. Note that all these update keys are transmitted through a public channel since they are useless for those revoked users.

– EHR data encryption and upload. Before outsourcing the EHR data to the cloud server, the EHR owner carefully defines an access structure, with which the owner encrypts a symmetric key by performing the encryption algorithm of $\mathsf{RS-HABE}$ such that only those intended medical workers can recover it. Then, the owner encrypts the EHR data with the selected symmetric key by running the encryption algorithm of $\mathsf{SE}$. Finally, the EHR owner uploads the ciphertexts of the symmetric key and the EHR data to the cloud server, along with the access structure and the current time period.

– Decryption key generation. For each unrevoked user, at the beginning of each time period, he derives a decryption key used throughout this time period by combining the original secret key with the update key. Meanwhile, at the end of the time period, the user erases the decryption key.

– EHR data access. If a medical worker's attributes or those attributes that can be derived from his/her attributes satisfy the access structure defined by an EHR owner, then he is allowed to decrypt the ciphertext of the symmetric key with the current decryption key, by running the decryption algorithm of $\mathsf{RS-HABE}$. Furthermore, the medical worker can download the ciphertext of the EHR data and decrypt it with the recovered symmetric key through calling the decryption algorithm of $\mathsf{SE}$. Here, to accurately recover the symmetric key, we require that the time period of the cloud server should be synchronized with medical workers'. On the other hand, if a medical worker's attributes do not satisfy the access structure, then he is denied to access the corresponding EHR data.

– Ciphertext update. At the end of each time period, the cloud server would update the ciphertext of the symmetric key from the current time period to the next one by running the ciphertext update algorithm of $\mathsf{RS-HABE}$, and remains the ciphertext of the EHR data unchanged. After this, the cloud server erases the previous ciphertext of the symmetric key. Throughout this process, the cloud server only uses the public parameter $\mathsf{pp}$, without the involvement any secret information.

– User revocation. In the case that a user (e.g., a medical worker or an EHR owner) leaves the system, the permission of such a user should be immediately abolished. To this end, the manager of this user revokes the user by running the revocation algorithm of $\mathsf{RS-HABE}$ such that the revoked user can no longer derive correct decryption keys for subsequent time periods.

## 4 The RS-HABE Construction

### 4.1 Overview

In our scheme, each system user has a unique identifier $\mathsf{ID}|_k$ and an attribute set $\mathsf{S}_{\mathsf{ID}|_k}$ consisting of vectors with length $k$.

Here, the integer $k$ also indicates the user $\mathsf{ID}|_k$'s position in the hierarchy. In addition, each user $\mathsf{ID}|_k$ maintains a binary tree $\mathcal{BT}_{\mathsf{ID}|_k}$ to manage all child users whose attributes can be derived from $\mathsf{S}_{\mathsf{ID}|_k}$. Namely, if $\mathsf{ID}|_{k+1}$ is a child user of $\mathsf{ID}|_k$, then for each attribute $\vec{x} = (x_1, \ldots, x_{k+1}) \in \mathsf{S}_{\mathsf{ID}|_{k+1}}$, there exists an attribute $\vec{x}' \in \mathsf{S}_{\mathsf{ID}|_k}$ such that $\vec{x} = (\vec{x}', x_{k+1})$. Particularly, when $k = 0$, we denote by $\mathsf{ID}|_0$ the attribute authority and $\mathcal{BT}_0$ the binary tree maintained by the attribute authority, which is used to manage all users at the first level.

To achieve scalable and efficient revocation, we mainly follow Boldyreva et al.'s [43] idea. Specifically, for any child user $\mathsf{ID}|_{k+1}$ of $\mathsf{ID}|_k$, $\mathsf{ID}|_k$ randomly assigns a retained leaf node of $\mathcal{BT}_{\mathsf{ID}|_k}$ to $\mathsf{ID}|_{k+1}$, and produces a secret component $sk_\theta$ for each node $\theta \in \mathsf{Path}(\mathsf{ID}|_{k+1})$. Meanwhile, depending on the revocation list $RL_{\mathsf{ID}|_k}$ at the current time period $t$, $\mathsf{ID}|_k$ periodically generates an update key component $uk_\theta$ for each node $\theta \in \mathsf{Y}_{\mathsf{ID}|_k} \leftarrow \mathsf{KUNode}(\mathcal{BT}_{\mathsf{ID}|_k}, RL_{\mathsf{ID}|_k}, t)$. As a result, by the principle of the algorithm $\mathsf{KUNode}$, if $\mathsf{ID}|_{k+1}$ is revoked by $\mathsf{ID}|_k$ before time period $t$, then the update key is completely useless for $\mathsf{ID}|_{k+1}$ since $\mathsf{Path}(\mathsf{ID}|_{k+1}) \cap \mathsf{Y}_{\mathsf{ID}|_k} = \emptyset$; otherwise, there will exist a node $\theta' \in \mathsf{Path}(\mathsf{ID}|_{k+1}) \cap \mathsf{Y}_{\mathsf{ID}|_k}$, which enables $\mathsf{ID}|_{k+1}$ to derive a decryption key $\mathsf{dk}^t_{\mathsf{ID}_{k+1}}$ by combining $sk_{\theta'}$ and $uk_{\theta'}$.

Secret key delegation is another favorite functionality of the proposed RS-HABE scheme. To achieve this, we adopt the structure form of user identities introduced in the setting of hierarchical identity-based encryption (HIBE) schemes [41], [44]. Intuitively, we can independently run the HIBE secret key delegation mechanism multiple times for delegating secret keys of low-level attribute vectors. However, this would make the scheme suffer from the collusion attack. To this end, for each node $\theta \in \mathsf{Path}(\mathsf{ID}|_k$, we associate each secret key component with a common random value $r_\theta$ in the form of $u_\tau^{r_\theta}$. As a result, without the knowledge of $r_\theta$, different users cannot connect their secret key components to participate the collusion attack. More precisely, for each attribute $\vec{x} \in \mathsf{S}_{\mathsf{ID}|_k}$ holding by a system user $\mathsf{ID}|_k$, we compute decryption key components $dk_{\vec{x},0}, dk_{\vec{x},1}, dk_{\vec{x},k+1}, \ldots, dk_{\vec{x},\ell}$, where $\ell$ is the maximum length of attribute vectors. This enables $\mathsf{ID}|_k$ to produce update key components $uk_{\vec{x},0}, uk_{\vec{x},1}, uk_{\vec{x},k+1}, \ldots, uk_{\vec{x},\ell}$. Furthermore, for an attribute $\vec{x}'$ of a $\mathsf{ID}|_k$'s child $\mathsf{ID}|_{k+1}$ that can be derived from $\vec{x}$, $\mathsf{ID}|_{k+1}$ can compute correct decryption key components $dk_{\vec{x}',0}, dk_{\vec{x}',1}, dk_{\vec{x}',k+1}, \ldots, dk_{\vec{x}',\ell}$ by combing the above update key components and his/her original secret key components $sk_{\vec{x}',0}, sk_{\vec{x}',1}, sk_{\vec{x}',k+1}, \ldots, sk_{\vec{x}',\ell}$, which completes the key delegation process.

To prevent a revoked system user from accessing those previously encrypted data with his/her former decryption keys, we utilize the idea of forward secure public encryption and signature schemes[2] [45], [46], and associate each ciphertext with the current time period. Meanwhile, the ciphertext is periodically updated in a public way, without using secret keys. Specifically, the whole lifetime of each ciphertext is divided into $T$ discrete time periods $\{0, 1, \ldots, T = 2^d - 1\}$ and use a binary tree $\mathcal{T}$ of depth $d$ to manage them. That is, each time period $t$ is assigned to a leaf node $\vartheta_t$ of $\mathcal{T}$ in chronological order from left to right. For a node $\vartheta$ of $\mathcal{T}$, denote by $\mathsf{R}(\vartheta)$ its right child, and $b_\vartheta$ the binary string that corresponds to the path from the root node to $\vartheta$, where 0 and 1 indicate that the path traverses the left child and right child of the parent node, respectively. Moreover, for a time period $t$, we define a set $\mathcal{N}_t = \{\mathsf{R}(\vartheta) | \vartheta \in \mathsf{Path}(\vartheta_t) \wedge \mathsf{R}(\vartheta) \notin \mathsf{Path}(\vartheta_t)\} \cup \{\vartheta_t\}$. As shown in [46], such defined sets have the following property.

**Property 1.** *Give two time periods $t$ and $t'$ satisfying $t < t'$, for each node $\vartheta' \in \mathcal{N}_{t'}$, there exists a node $\vartheta \in \mathcal{N}_t$ such that $b_\vartheta$ is a prefix of $b_{\vartheta'}$.*

Then, for each node $\vartheta \in \mathcal{N}_t$, we independently compute a corresponding ciphertext component comprised of two parts: one part is an essential input of the decryption algorithm that enables those unrevoked users to decrypt the ciphertext; another part is just used to update the ciphertext from $t$ to $t+1$.

In short, combining Waters' [47] ciphertext-policy ABE scheme with the above techniques in a tight and novel way enables us to construct a RS-HABE scheme that features of dynamic user revocation, scalable secret key delegation and public ciphertext update simultaneously.

## 4.2 The Construction

The proposed ciphertext-policy RS-HABE construction is specified as follows.

$\mathsf{Setup}(\lambda, N, T, \ell, \phi)$: The setup algorithm takes as input a security parameter $\lambda$, a maximum number $N$ of system users at each level, a maximum hierarchical length $\ell$, a total number $T = 2^d$ of time periods, and a maximum number $\phi$ of attribute categories. Then, it conducts the following steps to produce the public parameter and master secret key:

(1) Utilize the algorithm $\mathcal{G}(\cdot)$ with the security parameter $\lambda$ to produce bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, g, p, e) \leftarrow \mathcal{G}(\lambda)$, and choose random integers $\alpha, a \in \mathbb{Z}_p$.

(2) Define the attribute universe to be $\mathbb{U} = \bigcup_{k=1}^{\ell} \mathbb{Z}_p^k$. Each attribute vector $\vec{x} \in \mathbb{Z}_p^k$ is associated with two parameters $(\tau, k) \in [\phi] \times [\ell]$ that corresponds to its category and length, respectively. Let $\psi(\cdot)$ be a function that maps an attribute $\vec{x}$ to its category,[3] which is determined by the first component of $\vec{x}$.

(3) For integers $k \in [\ell]$, $\tau \in [\phi]$ and $j \in [d]$, choose random group elements $h_k, u_\tau, v_j \in \mathbb{G}_1$, respectively. Additionally, select another random group element $v_0 \in \mathbb{G}_1$. To simplify subsequent descriptions, we define the following two functions for each time period $t$ and attribute vector $\vec{x} = (x_1, \ldots, x_k)$:

$$V(t) = v_0 \cdot \prod_{j=1}^{d} v_j^{t[j]}, \; H(\vec{x}) = \prod_{j=1}^{k} h_j^{x_j},$$

where $t[j]$ means the $j$th bit of $t$ in the form of $d$-length binary string.

(4) Set the master secret key as $\mathsf{msk} = \{g^\alpha\}$ and publish the public parameter as

$$\mathsf{pp} = \{e(g,g)^\alpha, g^a, g, h_1, \ldots, h_\ell, u_1, \ldots, u_\phi, v_0, \ldots, v_d\}.$$

2. In the context of forward secure public encryption schemes, the secret key is periodically updated. Here, we evolve the ciphertext.

3. In the implementation of the proposed scheme, we define the map function as $\psi(\vec{x}) = x_1$.

$\mathsf{SKeyGen}(\mathsf{pp}, \mathsf{st}_{\mathsf{ID}|_{k-1}}, \mathsf{S}_{\mathsf{ID}|_k})$: This algorithm takes as input the public parameter $\mathsf{pp}$, the state information $\mathsf{st}_{\mathsf{ID}|_{k-1}}$ of the binary tree $\mathcal{BT}_{\mathsf{ID}|_{k-1}}$ maintained by the user $\mathsf{ID}|_{k-1}$, and an attribute set $\mathsf{S}_{\mathsf{ID}|_k}$ composed of vectors with length $k \leq \ell$[4], and then generates a secret key for $\mathsf{S}_{\mathsf{ID}|_k}$ as follows:

(1) Randomly select an unassigned leaf node $\eta$ of $\mathcal{BT}_{\mathsf{ID}|_{k-1}}$ and associate it with $\mathsf{ID}|_k$. For each node $\theta \in \mathsf{Path}(\mathsf{ID}|_k)$, recall the group element $R_\theta$ if it was defined. Otherwise, choose $R_\theta \in \mathbb{G}_1$ at random and store it in $\theta$.

(2) Choose a random exponent $r_\theta \in \mathbb{Z}_p$, and compute

$$sk_{\theta,0} = g^{r_\theta},\ sk_{\theta,1} = R_\theta \cdot g^{ar_\theta}.$$

(3) For each attribute vector $\vec{x} = (x_1, \ldots, x_k) \in \mathsf{S}_{\mathsf{ID}|_k}$, identify its category $\tau = \psi(\vec{x})$ and randomly select $\xi_{\vec{x}} \in \mathbb{Z}_p$. Then, calculate the following secret key components:

$$sk_{\vec{x},0} = u_\tau^{r_\theta} \cdot H(\vec{x})^{\xi_{\vec{x}}},\ sk_{\vec{x},1} = g^{\xi_{\vec{x}}},\ sk_{\vec{x},j} = h_j^{\xi_{\vec{x}}},$$

where $k+1 \leq j \leq \ell$.

(4) Return the secret key of $\mathsf{S}_{\mathsf{ID}|_k}$ in the form of

$$\mathsf{sk}_{\mathsf{ID}|_k} = \left\{sk_{\theta,0}, sk_{\theta,1}, sk_{\vec{x},0}, sk_{\vec{x},1}, sk_{\vec{x},k+1}, \ldots, sk_{\vec{x},\ell}\right\},$$

where $\theta \in \mathsf{Path}(\mathsf{ID}|_k)$ and $\vec{x} \in \mathsf{S}_{\mathsf{ID}|_k}$.

$\mathsf{UKeyGen}_{\mathsf{AA}}(\mathsf{pp}, \mathsf{msk}, \mathsf{st}_0, RL_0, t)$: This algorithm, which is performed by the attribute authority, takes as input the public parameter $\mathsf{pp}$, the master secret key $\mathsf{msk}$, the state information $\mathsf{st}_0$ of $\mathcal{BT}_0$, the revocation list $RL_0$ and the current time period $t$. It produces an update key for unrevoked users at the first level in the following way:

(1) Call the algorithm $\mathsf{KUNode}(\mathcal{BT}_0, RL_0, t)$ to generate a node set $\mathsf{Y}_0$.

(2) For each node $\theta \in \mathsf{Y}_0$, recall the group element $R_\theta$ if it was previously defined. Otherwise, choose $R_\theta \in \mathbb{G}_1$ at random and store it in $\theta$.

(3) For each node $\theta \in \mathsf{Y}_0$, select a random exponent $\gamma_\theta \in \mathbb{Z}_p$ and compute

$$uk_{\theta,0} = g^\alpha \cdot R_\theta^{-1} \cdot V(t)^{\gamma_\theta}, \quad uk_{\theta,1} = g^{\gamma_\theta}.$$

(4) Return the current update key of in the form of

$$\mathsf{uk}_{0,t} = \{uk_{\theta,0}, uk_{\theta,1}\}_{\theta \in \mathsf{Y}_0}.$$

$\mathsf{DKeyGen}_{\mathsf{ID}|_1}(\mathsf{pp}, \mathsf{sk}_{\mathsf{ID}|_1}, \mathsf{uk}_{0,t})$: This algorithm is run by an unrevoked 1-level user $\mathsf{ID}|_1$, and takes as input the public parameter $\mathsf{pp}$, $\mathsf{ID}|_1$'s secret key $\mathsf{sk}_{\mathsf{ID}|_1}$ and the update key $\mathsf{uk}_{0,t}$. It derives a decryption key for $\mathsf{ID}|_1$ as follows:

(1) Find out the common node $\theta' \in \mathsf{Path}(\mathsf{ID}|_1) \cap \mathsf{Y}_0$ and pick random integers $r', \gamma' \in \mathbb{Z}_p$, then compute

$$dk_0 = sk_{\theta',0} \cdot g^{r'} = g^{r_{\theta'}+r'}, dk_1 = uk_{\theta',1} \cdot g^{\gamma'} = g^{\gamma'+\gamma_{\theta'}},$$
$$dk_2 = sk_{\theta',1} g^{ar'} uk_{\theta',0} V(t)^{\gamma'} = g^\alpha g^{a(r_{\theta'}+r')} V(t)^{\gamma'+\gamma_{\theta'}}.$$

(2) For each attribute $\vec{x} \in \mathsf{S}_{\mathsf{ID}|_1}$, let its category be $\tau = \psi(\vec{x})$ and calculate

$$dk_{\vec{x},0} = sk_{\vec{x},0} u_\tau^{r'} = u_\tau^{r_{\theta'}+r'} H(\vec{x})^{\xi_{\vec{x}}},\ dk_{\vec{x},1} = sk_{\vec{x},1} = g^{\xi_{\vec{x}}},$$
$$dk_{\vec{x},j} = sk_{\vec{x},j} = h_j^{\xi_{\vec{x}}} \text{ for } j = 2 \text{ to } \ell.$$

(3) Return the decryption key in the form of

$$\mathsf{dk}^t_{\mathsf{ID}|_1} = \left\{dk_0, dk_1, dk_2, \{dk_{\vec{x},0}, dk_{\vec{x},1}, \{dk_{\vec{x},j}\}_{j=2}^{\ell}\}_{\vec{x} \in \mathsf{S}_{\mathsf{ID}|_1}}\right\}.$$

$\mathsf{UKeyGen}_{\mathsf{ID}|_{k-1}}(\mathsf{pp}, \mathsf{dk}^t_{\mathsf{ID}|_{k-1}}, \mathsf{st}_{\mathsf{ID}|_{k-1}}, RL_{\mathsf{ID}|_{k-1}}, t)$: This algorithm is run by a user $\mathsf{ID}|_{k-1}$, takes as input the public parameter $\mathsf{pp}$, the user's decryption key $\mathsf{dk}^t_{\mathsf{ID}|_{k-1}}$, the state information $\mathsf{st}_{\mathsf{ID}|_{k-1}}$ of $\mathcal{BT}_{\mathsf{ID}|_{k-1}}$ and the revocation list $RL_{\mathsf{ID}|_{k-1}}$ as well as the time period $t$. It produces an update key for unrevoked children of $\mathsf{ID}|_{k-1}$ as follows:

(1) Call the algorithm $\mathsf{KUNode}(\mathcal{BT}_{\mathsf{ID}|_{k-1}}, RL_{\mathsf{ID}|_{k-1}}, t)$ to get a node set $\mathsf{Y}_{\mathsf{ID}|_{k-1}}$.

(2) For each node $\theta \in \mathsf{Y}_{\mathsf{ID}|_{k-1}}$, recall $R_\theta$ if it was defined. Otherwise, randomly pick $R_\theta \in \mathbb{G}_1$ and store it in $\theta$.

(3) Select a random integer $r' \in \mathbb{Z}_p$ and compute the first component of the update key $uk_0 = dk_0 \cdot g^{r'} = g^{r+r'}$.

(4) For each node $\theta \in \mathsf{Y}_{\mathsf{ID}|_{k-1}}$, randomly pick $\gamma_\theta \in \mathbb{Z}_p$ and compute the following components of the update key

$$uk_{\theta,0} = R_\theta^{-1} g^{ar'} V(t)^{\gamma_\theta} dk_2 = g^\alpha g^{a(r+r')} R_\theta^{-1} V(t)^{\gamma_\theta+\gamma},$$
$$uk_{\theta,1} = dk_1 g^{\gamma_\theta} = g^{\gamma+\gamma_\theta}.$$

(5) For each attribute $\vec{x} \in \mathsf{S}_{\mathsf{ID}|_{k-1}}$, first identify its category $\tau = \psi(\vec{x})$ and randomly choose $\xi'_{\vec{x}} \in \mathbb{Z}_p$, then compute

$$uk_{\vec{x},0} = dk_{\vec{x},0} \cdot u_\tau^{r'} \cdot H(\vec{x})^{\xi'_{\vec{x}}} = u_\tau^{r+r'} H(\vec{x})^{\xi_{\vec{x}}+\xi'_{\vec{x}}},$$
$$uk_{\vec{x},1} = dk_{\vec{x},1} \cdot g^{\xi'_{\vec{x}}} = g^{\xi_{\vec{x}}+\xi'_{\vec{x}}},$$
$$uk_{\vec{x},j} = dk_{\vec{x},j} \cdot h_j^{\xi'_{\vec{x}}} = h_j^{\xi_{\vec{x}}+\xi'_{\vec{x}}} \quad \text{for} \quad j = k \text{ to } \ell.$$

(6) Return the update key in the form of

$$\mathsf{uk}_{\mathsf{ID}|_{k-1},t} = \Big\{uk_0, \{uk_{\theta,0}, uk_{\theta,1}\}_{\theta \in \mathsf{Y}_{\mathsf{ID}|_{k-1}}}, \{uk_{\vec{x},0}, uk_{\vec{x},1}, \{uk_{\vec{x},j}\}_{j=k}^{\ell}\}_{\vec{x} \in \mathsf{S}_{\mathsf{ID}|_{k-1}}}\Big\}.$$

$\mathsf{DKeyGen}_{\mathsf{ID}|_k}(\mathsf{pp}, \mathsf{sk}_{\mathsf{ID}|_k}, \mathsf{uk}_{\mathsf{ID}|_{k-1},t})$: This algorithm is run by an unrevoked user $\mathsf{ID}|_k$, and takes as input the public parameter $\mathsf{pp}$, $\mathsf{ID}|_k$'s secret key $\mathsf{sk}_{\mathsf{ID}|_k}$ and the current update key $\mathsf{uk}_{\mathsf{ID}|_{k-1},t}$. It outputs a decryption key for $\mathsf{ID}|_k$ as follows:

(1) Identify the common node $\theta' \in \mathsf{Path}(\mathsf{ID}|_k) \cap \mathsf{Y}_{\mathsf{ID}|_{k-1}}$ and select random integers $r', \gamma' \in \mathbb{Z}_p$, then compute

$$\begin{aligned} dk_0 &= sk_{\theta',0} \cdot uk_0 \cdot g^{r'} = g^{r_{\theta'}+r''+r'}, \\ dk_1 &= uk_{\theta',1} \cdot g^{\gamma'} = g^{\gamma_{\theta'}+\gamma'}, \\ dk_2 &= sk_{\theta',1} \cdot g^{ar'} \cdot uk_{\theta',0} \cdot V(t)^{\gamma'} \\ &= g^\alpha \cdot g^{a(r_{\theta'}+r''+r')} \cdot V(t)^{\gamma_{\theta'}+\gamma'}, \end{aligned}$$

where $uk_0 = g^{r''}$, $uk_{\theta',0} = g^\alpha g^{ar''} R_{\theta'}^{-1} V(t)^{\gamma_{\theta'}}$.

4. We suppose that a user's attributes all lie at the same level.

(2) For each attribute $\vec{x} \in \mathsf{S}_{\mathsf{ID}|_k}$, let $\vec{x}' \in \mathsf{S}_{\mathsf{ID}|_{k-1}}$ be the prefix of $\vec{x}$, i.e., $\vec{x} = (\vec{x}', x_k)$. Then, compute

$$dk_{\vec{x},0} = sk_{\vec{x},0} uk_{\vec{x}',0} u_\tau^{r'} (uk_{\vec{x}',k})^{x_k} = u_\tau^{r_{\theta'}+r'+r''} H(\vec{x})^{\xi_{\vec{x}}+\xi_{\vec{x}'}},$$
$$dk_{\vec{x},1} = sk_{\vec{x},1} \cdot uk_{\vec{x}',1} = g^{\xi_{\vec{x}}+\xi_{\vec{x}'}},$$
$$dk_{\vec{x},j} = sk_{\vec{x},j} \cdot uk_{\vec{x}',j} = h_j^{\xi_{\vec{x}}+\xi_{\vec{x}'}} \quad \text{for} \quad j = k+1 \ \text{ to } \ \ell,$$

where $uk_{\vec{x}',0} = u_\tau^{r''} H(\vec{x}')^{\xi_{\vec{x}'}}$.

(3) Return the decryption key of $\mathsf{S}_{\mathsf{ID}|_k}$ in the form of

$$\mathsf{dk}^t_{\mathsf{ID}|_k} = \left\{ dk_0, dk_1, dk_2, \{dk_{\vec{x},0}, dk_{\vec{x},1}, \{dk_{\vec{x},j}\}_{j=k+1}^{\ell}\}_{\vec{x} \in \mathsf{S}_{\mathsf{ID}|_k}} \right\}.$$

$\mathsf{Encrypt}(\mathsf{pp}, m, t, (A, \rho))$: This algorithm takes as input the public parameter $\mathsf{pp}$, a message $m \in \mathbb{G}_2$ to encrypt, the current time period $t$ and a LSSS access structure $(A, \rho)$, where $A$ is a $l \times n$ matrix and $\rho$ is a function mapping rows of $A$ to attributes of length $k$ (i.e., $\rho : \{1, \ldots, l\} \rightarrow \mathbb{Z}_p^k$). It generates a ciphertext $\mathsf{ct}^t$ as follows:

(1) For each node $\vartheta \in \mathcal{N}_t$, choose a random row vector $\vec{v}_\vartheta = (s_\vartheta, s_2, \ldots, s_n) \in \mathbb{Z}_p^n$. For each index $i \in [l]$, pick a random integer $z_i \in \mathbb{Z}_p$, utilize the $i$th row $A_i$ of the matrix $A$ to compute $\lambda_i = A_i \cdot (\vec{v}_\vartheta)^{\mathrm{T}}$. Then, produce the following ciphertext components:

$$\widetilde{c}_{\vartheta,1} = m \cdot e(g,g)^{\alpha s_\vartheta}, \ \widetilde{c}_{\vartheta,2} = g^{s_\vartheta}, \ \widetilde{c}_{\vartheta,i,0} = g^{a\lambda_i} \cdot u_\tau^{z_i},$$
$$\widetilde{c}_{\vartheta,i,1} = g^{-z_i}, \ \widetilde{c}_{\vartheta,i,2} = H(\vec{x})^{z_i} \text{ for } \rho(i) = \vec{x} \text{ and } \psi(\vec{x}) = \tau.$$

Furthermore, calculate $\vec{c}_\vartheta = (c_{\vartheta,0}, c_{\vartheta,|b_\vartheta|+1}, \ldots, c_{\vartheta,d})$, where

$$c_{\vartheta,0} = \left( v_0 \cdot \prod_{j=1}^{|b_\vartheta|} v_j^{t[j]} \right)^{s_\vartheta}, \ c_{\vartheta,j} = v_j^{s_\vartheta} \text{ for } j = |b_\vartheta| + 1 \text{ to } d.$$

Note that for the leaf node $\vartheta_t$, the ciphertext component $\vec{c}_{\vartheta_t} = (c_{\vartheta_t,0})$ is computed as $c_{\vartheta_t,0} = V(t)^{s_\vartheta}$.

(2) Return the ciphertext in the form of

$$\mathsf{ct}^t = \left\{ \widetilde{c}_{\vartheta,1}, \widetilde{c}_{\vartheta,2}, \{\widetilde{c}_{\vartheta,i,0}, \widetilde{c}_{\vartheta,i,1}, \widetilde{c}_{\vartheta,i,2}\}_{i=1}^{l}, \vec{c}_\vartheta \right\}_{\vartheta \in \mathcal{N}_t},$$

associated with $(A, \rho)$ and $t$.

$\mathsf{CTUpdate}(\mathsf{pp}, \mathsf{ct}^t, t')$: This algorithm takes as input the public parameter $\mathsf{pp}$, a ciphertext $\mathsf{ct}^t$ at the time period $t$ and the next time period $t'$. It updates the ciphertext from $t$ to $t'$ as follows:

(1) For each node $\vartheta' \in \mathcal{N}_{t'}$, find out a node $\vartheta \in \mathcal{N}_t$ such that the binary string $b_\vartheta$ is a prefix of $b_{\vartheta'}$, and also pick a random vector $\vec{v}_{\vartheta'} = (s_{\vartheta'}, s_2', \ldots, s_n')$ from $\mathbb{Z}_p^n$

(2) For each index $i \in [l]$, select a random integer $z_i' \in \mathbb{Z}_p$, compute $\lambda_i' = A_i \cdot (\vec{v}_{\vartheta'})^{\mathrm{T}}$, and then calculate the following ciphertext components:

$$\widetilde{c}_{\vartheta',1} = \widetilde{c}_{\vartheta,1} \cdot e(g,g)^{\alpha s_{\vartheta'}}, \ \widetilde{c}_{\vartheta',2} = \widetilde{c}_{\vartheta,2} \cdot g^{s_{\vartheta'}},$$
$$\widetilde{c}_{\vartheta',i,0} = \widetilde{c}_{\vartheta,i,0} \cdot g^{a\lambda_i'} \cdot u_\tau^{z_i'}, \ \widetilde{c}_{\vartheta',i,1} = \widetilde{c}_{\vartheta,i,1} \cdot g^{-z_i'},$$
$$\widetilde{c}_{\vartheta',i,2} = \widetilde{c}_{\vartheta,i,2} \cdot H(\vec{x})^{z_i'} \text{ for } \rho(i) = \vec{x} \text{ and } \psi(\vec{x}) = \tau.$$

Moreover, compute $\vec{c}_{\vartheta'} = (c_{\vartheta',0}, c_{\vartheta',|b_{\vartheta'}|+1}, \ldots, c_{\vartheta',d})$, where

$$c_{\vartheta',0} = c_{\vartheta,0} \cdot \left( v_0 \cdot \prod_{j=1}^{|b_{\vartheta'}|} v_j^{t[j]} \right)^{s_{\vartheta'}} \cdot \prod_{j=|b_\vartheta|+1}^{|b_{\vartheta'}|} (c_{\vartheta,j})^{t'[j]},$$
$$c_{\vartheta',j} = c_{\vartheta,j} \cdot v_j^{s_{\vartheta'}} \text{ for } j = |b_{\vartheta'}| + 1 \text{ to } d.$$

(3) Return the updated ciphertext in the form of

$$\mathsf{ct}^{t'} = \left\{ \widetilde{c}_{\vartheta',1}, \widetilde{c}_{\vartheta',2}, \{\widetilde{c}_{\vartheta',i,0}, \widetilde{c}_{\vartheta',i,1}, \widetilde{c}_{\vartheta',i,2}\}_{i=1}^{l}, \vec{c}_{\vartheta'} \right\}_{\vartheta' \in \mathcal{N}_{t'}},$$

associated with $(A, \rho)$ and $t'$. Meanwhile, erase $\mathsf{ct}^t$.

$\mathsf{Decrypt}(\mathsf{pp}, \mathsf{dk}^t_{\mathsf{ID}|_k}, \mathsf{ct}^t, (A, \rho), t)$: This algorithm takes as input the public parameter $\mathsf{pp}$, a decryption key $\mathsf{dk}^t_{\mathsf{ID}|_k}$ and a ciphertext $\mathsf{ct}^t$ associated with an access structure $(A, \rho)$ and a time period $t$. The decryption procedure only uses the ciphertext components corresponding to the leaf node $\vartheta_t$, i.e., $\{\widetilde{c}_{\vartheta_t,1}, \widetilde{c}_{\vartheta_t,2}, \{\widetilde{c}_{\vartheta_t,i,0}, \widetilde{c}_{\vartheta_t,i,1}, \widetilde{c}_{\vartheta_t,i,2}\}_{i=1}^{l}, \vec{c}_{\vartheta_t} = (c_{\vartheta_t,0})\}$. If $\mathsf{S}_{\mathsf{ID}|_k}$ satisfies the access structure of the ciphertext,[5] then the message can be correctly recovered as follows:

(1) Let $I = \{i \in [l] | \rho(i) \in \mathsf{S}_{\mathsf{ID}|_k}\}$, compute a set of constants $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ such that $\sum_{i \in I} w_i \lambda_i = s_\vartheta$, where $\{\lambda_i\}_{i \in I}$ are valid shares of the secret $s_\vartheta \in \mathbb{Z}_p$ generated according to $(A, \rho)$.

(2) Compute

$$C = \prod_{i \in I} \left( e(\widetilde{c}_{\vartheta_t,i,0}, dk_0) \cdot e(\widetilde{c}_{\vartheta_t,i,1}, dk_{\rho(i),0}) \right)^{w_i} \cdot \prod_{i \in I} \left( e(\widetilde{c}_{\vartheta_t,i,2}, dk_{\rho(i),1}) \right)^{w_i},$$
$$C' = \frac{e(\widetilde{c}_{\vartheta_t,2}, dk_2)}{C \cdot e(c_{\vartheta_t,0}, dk_1)},$$

and recover the message as $m = \widetilde{c}_{\vartheta_t,1} / C'$.

$\mathsf{Revoke}_{\mathsf{ID}|_{k-1}}(\mathsf{st}_{\mathsf{ID}|_{k-1}}, RL_{\mathsf{ID}|_{k-1}}, \mathsf{ID}|_k, t)$: This algorithm, which is performed by a user $\mathsf{ID}|_{k-1}$, takes as input the state information $\mathsf{st}_{\mathsf{ID}|_{k-1}}$ of the binary tree $\mathcal{BT}_{\mathsf{ID}|_{k-1}}$ and the current revocation list $RL_{\mathsf{ID}_{k-1}}$, a user identifier $\mathsf{ID}|_k$ to be revoked and the revocation time period $t$. It updates the revocation list as $RL_{\mathsf{ID}|_{k-1}} \leftarrow RL_{\mathsf{ID}|_{k-1}} \cup \{(\mathsf{ID}|_k, t)\}$.

Correctness proof. Now we show the correctness of the proposed RS-HABE construction.

First, observe that a user $\mathsf{ID}|_k$'s well-formed decryption key $\mathsf{dk}^t_{\mathsf{S}_{\mathsf{ID}|_k}}$ at a time period $t$ is in the form of

$$\mathsf{dk}^t_{\mathsf{ID}|_k} = \left\{ dk_0, dk_1, dk_2, \{dk_{\vec{x},0}, dk_{\vec{x},1}, \{dk_{\vec{x},j}\}_{j=k+1}^{\ell}\}_{\vec{x} \in \mathsf{S}_{\mathsf{ID}|_k}} \right\},$$

where,

$$dk_0 = g^r, \ dk_1 = g^\gamma, \ dk_2 = g^\alpha \cdot g^{ar} \cdot V(t)^\gamma,$$
$$dk_{\vec{x},0} = u_\tau^r \cdot H(\vec{x})^{\xi_{\vec{x}}}, \ dk_{\vec{x},1} = g^{\xi_{\vec{x}}}, \ dk_{\vec{x},j} = h_j^{\xi_{\vec{x}}},$$

and $r, \gamma, \xi_{\vec{x}} \in \mathbb{Z}_p$ are random integers.

5. We assume that the length of attributes in the decryption key is equal to the length of attribute vectors in the policy. If not, we can extend the length of attribute vectors in the decryption key and delegate new decryption key by performing the corresponding algorithms.

TABLE 1
Comparisons of Computation Cost with Related Works

| Schemes | Setup | SKeyGen | UKeyGen | DKeyGen | Encrypt | CTUpdate | Decrypt |
|---|---|---|---|---|---|---|---|
| Li et al. [3] | $\mathcal{O}(\lvert\mathcal{U}\rvert)$E+ $\mathcal{O}(1)$P | $\mathcal{O}(\lvert\mathcal{S}\rvert)$E | $\mathcal{O}(r_a)$E | $\mathcal{O}(r_a)$E | $\mathcal{O}(k)$E | $\mathcal{O}(r_a)$E | $\mathcal{O}(\lvert I\rvert)$P |
| Zhou et al. [6] | $\mathcal{O}(\lvert\mathcal{U}\rvert)$E+ $\mathcal{O}(1)$P | $\mathcal{O}(\lvert\mathcal{S}\rvert\log N)$E | $\mathcal{O}(r_u\log(N/r_u))$E | N/A | $\mathcal{O}(ld)$E | $\mathcal{O}(ld)$E | $\mathcal{O}(\lvert I\rvert)$E+$\mathcal{O}(\lvert I\rvert)$P |
| Yu et al. [20] | $\mathcal{O}(\lvert\mathcal{U}\rvert)$E+ $\mathcal{O}(1)$P | $\mathcal{O}(\lvert\mathcal{U}\rvert)$E | $\mathcal{O}(r_a)$E | $\mathcal{O}(r_a)$E | $\mathcal{O}(\lvert\mathcal{U}\rvert)$E | $\mathcal{O}(r_a)$E | $\mathcal{O}(\lvert\mathcal{U}\rvert)$P |
| Yang et al. [21] | $\mathcal{O}(\lvert\mathcal{U}\rvert)$E+ $\mathcal{O}(1)$P | $\mathcal{O}(\lvert\mathcal{S}\rvert)$E | $\mathcal{O}(r_a(N-r_u))$E | N/A | $\mathcal{O}(l)$E | $\mathcal{O}(r_a)$E | $\mathcal{O}(\lvert I\rvert)$E+$\mathcal{O}(\lvert I\rvert)$P |
| Yang et al. [22] | $\mathcal{O}(\lvert\mathcal{U}\rvert)$E+ $\mathcal{O}(1)$P | $\mathcal{O}(\lvert\mathcal{S}\rvert)$E | $\mathcal{O}(r_a(N-r_u))$E | N/A | $\mathcal{O}(l)$E | $\mathcal{O}(r_a)$E | $\mathcal{O}(\lvert I\rvert)$E+$\mathcal{O}(\lvert I\rvert)$P |
| Liang et al. [26] | $\mathcal{O}(1)$E+ $\mathcal{O}(1)$P | $\mathcal{O}(\lvert\mathcal{S}\rvert+\log N)$E | $\mathcal{O}(r_u\log(N/r_u))$E | N/A | $\mathcal{O}(l)$E | N/A | $\mathcal{O}(\lvert I\rvert)$E+$\mathcal{O}(\lvert I\rvert)$P |
| Deng et al. [39] | $\mathcal{O}(1)$P | $\mathcal{O}(\ell\lvert\mathcal{S}\rvert)$E | N/A | N/A | $\mathcal{O}(\ell l)$E | N/A | $\mathcal{O}(\lvert I\rvert)$E+$\mathcal{O}(\lvert I\rvert)$P |
| Sahai et al. [8] | $\mathcal{O}(\lvert\mathcal{U}\rvert)$E+ $\mathcal{O}(1)$P | $\mathcal{O}(\lvert\mathcal{S}\rvert\log N)$E | $\mathcal{O}(r_u\log(N/r_u))$E | N/A | $\mathcal{O}(ld)$E | $\mathcal{O}(ld)$E | $\mathcal{O}(\lvert I\rvert)$E+$\mathcal{O}(\lvert I\rvert)$P |
| Ours | $\mathcal{O}(1)$P | $\mathcal{O}(\ell\lvert\mathcal{S}\rvert\log N)$E | $\mathcal{O}(r_u\log(N/r_u)+\ell\lvert\mathcal{S}\rvert)$E | $\mathcal{O}(\ell\lvert\mathcal{S}\rvert)$E | $\mathcal{O}(d(\ell l+d))$E | $\mathcal{O}(d(\ell l+d))$E | $\mathcal{O}(\lvert I\rvert)$E+$\mathcal{O}(\lvert I\rvert)$P |

*$N$ = the number of users in the system. $r_u$ = the number of users to be revoked. $d$ = the depth of the binary tree used to manage all time periods. $l$ = the number of rows of LSSS matrix. $\ell$ = the maximum length of attribute vectors. $k$ = the number of attributes associated with the ciphertext in KP-ABE. $|I|$ = the number of attributes involved in decryption algorithm. $|\mathcal{U}|$ = the size of attributes managed by an authority. $|\mathcal{S}|$ = the size of a user's attribute set. $r_a$ = the number of attributes to be revoked. E = the running time of one exponentiation operation. P = the running time of one pairing operation.*

Then, for each index $i \in I$, it holds that

$$\begin{aligned} C_i &= e(\widetilde{c}_{\vartheta_t,i,0}, dk_0)\cdot e(\widetilde{c}_{\vartheta_t,i,1}, dk_{\rho(i),0})\cdot e(\widetilde{c}_{\vartheta_t,i,2}, dk_{\rho(i),1}) \\ &= e(g^{a\lambda_i}u_\tau^{z_i}, g^r)e(g^{-z_i}, u_\tau^r H(\rho(i))^{\xi_{\rho(i)}})e(H(\rho(i))^{z_i}, g^{\xi_{\rho(i)}}) \\ &= e(g^{a\lambda_i}, g^r) = e(g,g)^{ar\lambda_i} \end{aligned}$$

which implies that $C = \prod_{i\in[I]} C_i^{w_i} = e(g,g)^{ars_{\vartheta_t}}$ since $\sum_{i\in I} w_i\lambda_i = s_{\vartheta_t}$. Consequently, we have that

$$C' = \frac{e(\widetilde{c}_{\vartheta_t,2}, dk_2)}{C\cdot e(c_{\vartheta_t,0}, dk_1)} = \frac{e(g^{s_{\vartheta_t}}, g^\alpha g^{ar}V(t)^\gamma)}{e(g,g)^{ars_{\vartheta_t}}\cdot e(V(t)^{s_{\vartheta_t}}, g^\gamma)} = e(g,g)^{\alpha s_{\vartheta_t}}$$

and $c_{\vartheta_t,1}/C' = m$. This completes the correctness proof.

Security proof. The security of the proposed RS-HABE scheme is captured by the following theorem.

**Theorem 1.** *If the decisional q-BDHE assumption holds, then no PPT adversary can break the selective security of the proposed RS-HABE scheme with a challenge access structure $(A^*_{l\times n}, \rho^*)$ and total number of time periods $T = 2^d$, where $n, d \le q$.*

Due to the limit of space, we provide the security model and corresponding proof in the supplemental file, available online.

## 5 Performance Evaluation

### 5.1 Theoretical Analysis

In Table 1 we compare the computation cost of the proposed RS-HABE scheme with several existing works. Since some of these listed ABE constructions (i.e., [6], [21], [22]) are constructed for multi-authority environment, we only consider the computation cost of these schemes in the special case of single attribute authority, for consistency with other listed schemes. Moreover, the computation cost of some particular algorithms is ignored, namely, the global setup algorithm in [6], [21], [22] and the trace algorithm in [6].

As indicated in Table 1, though it seems that the Setup algorithm in schemes of Liang et al. [26], Deng et al. [39] and our RS-HABE scheme has constant computation cost, the attribute authority in fact needs to sample a random group element for each system attribute. Thus, the computation cost of the Setup algorithm in all listed schemes mainly depends on the size of attribute universe $|\mathcal{U}|$. As in general CP-ABE schemes, the complexity of the secret key generation algorithm SKeyGen in [3], [21], [22] is determined by the size of input attribute set $|\mathcal{S}|$. To provide the functionalities of dynamic user revocation and scalable key delegation, the algorithm SKeyGen increases a factor of $\log N$ in [6], [8], $\ell$ in [39] and $\ell\log N$ in our RS-HABE scheme, respectively. In schemes of Li et al. [3] and Yu et al. [20], when some attribute of a user is revoked, the attribute authority needs to produce a new public version key of this attribute. As a consequence, the complexity of the UKeyGen algorithm in their schemes is linear with the number of revoked attributes $r_a$. In Yang et al.'s schemes [21], [22], when an attribute of some user is revoked, then the attribute authority independently generates a new secret key component for each user holding this attribute but not revoked. This implies that the computation overhead of UKeyGen algorithm in their schemes grows linear with $r_a(N-r_u)$. The ABE schemes of Zhou et al. [6], Liang et al. [26], Sahai et al. [8] and ours uses the user revocation manner

TABLE 2
Comparisons of Secret Key Size, Update Key Size and Ciphertext Size with Related Works

| Schemes | Secret key size | Update key size | Ciphertext size |
|---|---|---|---|
| Waters [47] | $\mathcal{O}(\lvert\mathcal{S}\rvert)\lvert\mathbb{G}_1\rvert$ | N/A | $\mathcal{O}(l)\lvert\mathbb{G}_1\rvert+\mathcal{O}(1)\lvert\mathbb{G}_2\rvert$ |
| Deng et al. [39] | $\mathcal{O}(\ell\lvert\mathcal{S}\rvert)\lvert\mathbb{G}_1\rvert$ | N/A | $\mathcal{O}(l)\lvert\mathbb{G}_1\rvert+\mathcal{O}(1)\lvert\mathbb{G}_2\rvert$ |
| Liang et al. [26] | $\mathcal{O}(\lvert\mathcal{S}\rvert+\log N)\lvert\mathbb{G}_1\rvert$ | $\mathcal{O}(r_u\log(N/r_u))\lvert\mathbb{G}_1\rvert$ | $\mathcal{O}(l)\lvert\mathbb{G}_1\rvert+\mathcal{O}(1)\lvert\mathbb{G}_2\rvert$ |
| Sahai et al. [8] | $\mathcal{O}(\lvert\mathcal{S}\rvert\cdot\log N)\lvert\mathbb{G}_1\rvert$ | $\mathcal{O}(r_u\cdot\log(N/r_u))\lvert\mathbb{G}_1\rvert$ | $\mathcal{O}(d\cdot l)\lvert\mathbb{G}_1\rvert+\mathcal{O}(d)\lvert\mathbb{G}_2\rvert$ |
| Zhou et al. [6] | $\mathcal{O}(\lvert\mathcal{S}\rvert\cdot\log N)\lvert\mathbb{G}_1\rvert$ | $\mathcal{O}(r_u\cdot\log(N/r_u))\lvert\mathbb{G}_1\rvert$ | $\mathcal{O}(d\cdot l)\lvert\mathbb{G}_1\rvert+\mathcal{O}(d)\lvert\mathbb{G}_2\rvert$ |
| Ours | $\mathcal{O}(\ell\cdot\lvert\mathcal{S}\rvert\cdot\log N)\lvert\mathbb{G}_1\rvert$ | $\mathcal{O}(r_u\cdot\log(N/r_u)+\ell\cdot\lvert\mathcal{S}\rvert)\lvert\mathbb{G}_1\rvert$ | $\mathcal{O}(d\cdot(d+l))\lvert\mathbb{G}_1\rvert+\mathcal{O}(d)\lvert\mathbb{G}_2\rvert$ |

*$|\mathbb{G}_i|(i=1,2)$ means the size of a group element in $\mathbb{G}_i$.*

TABLE 3
Comparisons of Dominant Operations in Different Bilinear Groups (ms)

| Operations | Prime | Prod. of 2 primes | Prod. of 3 primes |
|---|---|---|---|
| Pairing | 3.89 | 22.05 | 67.02 |
| Exp. | 1.18 | 5.76 | 15.27 |

introduced by Boldyreva et al. [43], which thus achieve scalable revocation. As in a general ABE scheme, the complexity of Encrypt, CTUpdate and Decrypt algorithms in [3], [20], [21], [22], [26] is determined by the size of involved (revoked) attributes. However, to achieve public ciphertext update, the computation cost of Encrypt and CTUpdate algorithms in schemes of Zhou et al. [6] and Sahai et al. [8] as well as our RS-HABE scheme increases a factor of $d$ (i.e., the depth of a binary tree used to manage time periods). As in Deng et al.'s [39] scheme, to provide the functionality of key delegation, the complexity of the two algorithms in our scheme additionally increases another factor of $\ell$, the maximum length of attribute vector. However, by precomputing $H(\vec{x}) = \prod_{j=1}^{k} h_j^{x_j}$, we can reduce their computation cost to $\mathcal{O}(d(l+d))$E.

In Table 2, we provide comparisons between the proposed RS-HABE scheme and related ABE schemes in terms of secret key size, update key size and ciphertext size. We can see that, similar to previous analysis, the proposed RS-HABE scheme also needs more storage space for secret key, update key and ciphertext. In fact, this is consistent with previous comparison results, since more components of secret key, update key and ciphertext need more computation overhead.

In all, when compared with other similar ABE schemes, the proposed RS-HABE scheme requires more storage overhead and computation cost. But they are upper bounded by the depth $\log N$ of the binary tree employed to manage all users, the depth $d$ of the binary tree used to manage all time periods and the maximum length $\ell$ of attribute vectors. In addition, the ABE schemes in [6], [8], [39] are built upon bilinear groups of composite order, in which the size of an element is much larger than that in a group of prime order, and the computation cost of the exponentiation and pairing operations is much higher than in bilinear groups with prime order. To illustrate this, in Table 3, we compare the time overhead of these two operations. The evaluation is performed with PBC 0.5.14 and Type A1 curve. Given three primes $p_1, p_2, p_3$ of 512 bits, the order of the underlying bilinear group successively is $p_1$, $p_1p_2$ and $p_1p_2p_3$. It can be seen that there exists a significant gap between the time overhead in bilinear groups of prime and composite orders.

Table 4 summarizes the security properties of these listed ABE schemes. We can see that previous ABE schemes [3], [20] supporting attribute revocation only realize rather simple access policies, such as conjunctive normal form and AND gate. In addition, in these two schemes, a same proxy re-key is used to update both ciphertext and secret key. Therefore, their schemes cannot resist the collusion attack from revoked users and the cloud server or malicious unrevoked users, and fail to achieve forward and backward security. Though Yang et al. [21], [22] attempted to improve the above attribute revocation manner by producing a unique update key for each unrevoked user, Hong et al. [48] pointed out that the scheme in [21] fails to provide the claimed forward/backward security, Wu et al. [49] demonstrated that the scheme in [22] also suffers from collusion attack. In addition, the revocation procedure in their schemes is performed in a private way. Namely, the authority needs to independently maintain a secure channel for each unrevoked user to issue the corresponding update key.

Zhou et al.'s [6] scheme and Sahai et al.'s [8] scheme as well as our RS-HABE scheme all support expressive access policies in the form of LSSS, and also capture the forward and backward security by means of dynamic user revocation and public ciphertext update. However, the other two schemes [6], [8] cannot withstand decryption key exposure (DKE) attack, a realistic threat initially considered by Seo and Emura [50]. More concretely, the decryption key in their schemes is a direct combination of the original secret key and the current update key. That is, when a user's decryption key gets exposed, an adversary can extract the original secret key from it with the public update key. The proposed RS-HABE scheme overcomes this by introducing the UKeyGen algorithm that derives the decryption key from multiplying the original secret key and the current update key in a random manner. As discussed in [50], re-randomizing the decryption key structured as in [6], [8] cannot make these schemes resist DKE attack. On the other hand, although the decryption key generation in ABE schemes of Yang et al. [21], [22] adopt a manner similar to ours, the derived decryption key is not re-randomized, and thus also cannot resist the DKE attack. In addition, only

TABLE 4
Comparisons of Security Property with Related Works

| Schemes | Access policy | Revocation level | HKD | Public update | Forward security | Backward security | DKE |
|---|---|---|---|---|---|---|---|
| Li et al. [3] | CNF | Attribute | × | × | × | × | N/A |
| Zhou et al. [6] | LSSS | User | × | ✓ | ✓ | ✓ | × |
| Yu et al. [20] | AND | Attribute | × | × | × | × | N/A |
| Yang et al. [21] | LSSS | Attribute | × | × | × | × | × |
| Yang et al. [22] | LSSS | Attribute | × | × | × | × | × |
| Liang et al. [26] | LSSS | User | × | ✓ | × | ✓ | × |
| Sahai et al. [8] | LSSS | User | × | ✓ | ✓ | ✓ | × |
| Deng et al. [39] | LSSS | N/A | ✓ | N/A | × | × | N/A |
| Ours | LSSS | User | ✓ | ✓ | ✓ | ✓ | ✓ |

**DKE = decryption key exposure. HKD = hierarchical key delegation.*

TABLE 5
Parameter Setting in the Basic Test

| Parameter values | Description |
|---|---|
| $r_u = 0$ | the number of revoked users. |
| $T = 2^8$ | the total number of time periods. |
| $\mathcal{S} = 10$ | the size of each user's attribute set. |
| $\ell = 8$ | the maximum length of attribute vector. |
| $\phi = 10$ | the maximum number of attribute categories. |
| $N = 2^8$ | the number of child users managed by a user. |

Deng et al.'s [39] ABE scheme and our RS-HABE scheme enjoy the functionality of hierarchical key delegation.

On the whole, from Tables 1, 2, 3, and 4, we can see only the proposed RS-HABE scheme features of the functionalities of dynamic user revocation, key delegation and public ciphertext update simultaneously, at the cost of acceptable computation overhead. Meanwhile, it is also secure against realistic key exposure attack and supports expressive access polices. Thus, the EHR data sharing framework based on RS-HABE, as described is in Section 3, is more desirable.

## 5.2 Implementation and Evaluation

We implemented the proposed RS-HABE scheme in Python 3.4 using Charm 0.43 framework [9]. We tested the proposed scheme on a super-singular symmetric elliptic curve group (SS512) with 160-bit prime order, and conducted all experiments on a PC with Intel Core(TM) i7-6700 CPU@3.40GHz and 16.0 GB RAM.

First, with the parameter setting in Table 5 and the access policy $(\vec{x}_1 \wedge \vec{x}_2 \wedge \vec{x}_3 \wedge \vec{x}_4 \wedge \vec{x}_5)$, where each $\vec{x}_i (i \in [1,5])$ is an attribute vector with length 4, we tested the average time cost of each algorithm in the proposed RS-HABE scheme. Particularly, in the parameter setting, a time period can be one hour, one day, or one week. Naturally, higher update frequency means higher security, but it also means higher computation and communication overhead. Thus, how often should the keys be updated depends on the configuration of practical security policy, which can be seen as a tradeoff between security and efficiency. From Fig. 2, we can see that the average time cost of these algorithms listed in part 1 and part 2 are no more than 30 milliseconds and 400 milliseconds, respectively. In particular, the average running time of SKeyGen is the largest one, which is nearly 1200 milliseconds. Since it is so far away from time costs of other algorithms, we do not display it in the figure. On the whole, in the given parameter setting, the practical performance of the proposed RS-HABE scheme is acceptable.

In Fig. 3, we demonstrate the average running time of the proposed RS-HABE scheme under the different choice of the maximum length $\ell$ of attribute vectors supported by the scheme. Specifically, throughout this experiment, we let the value of $\ell$ range from 6 to 15, and other parameters remain unchanged as in previous experiment, including the access policy. We can see that the average time cost of $\mathsf{UKeyGen}_{\mathrm{AA}}$, $\mathsf{DKeyGen}_{\mathrm{ID}_1}$, $\mathsf{DKeyGen}_{\mathrm{ID}_k}$, Encrypt, CTUpdate and Decrypt is free from $\ell$, and the average time cost of Setup and $\mathsf{UKeyGen}_{\mathrm{ID}_k}$ is slightly influenced by $\ell$. Moreover, note that the average time cost of SKeyGen grows linearly at a fast rate when the maximum length of attribute vector increases. This is mainly because that SKeyGen needs to generate more secret key components $\{sk_{\vec{x},j}\}_{j=k+1}^{\ell}$ when $\ell$ increases.

In Fig. 4, we illustrate how the sizes of user attribute set and access policy affect the performance of the proposed RS-HABE scheme. In this experiment, we let the size of user attribute $\mathcal{S}$ range from 2 to 10, and the access policy range from $(\vec{x}_1 \wedge \vec{x}_2)$ to $(\vec{x}_1 \wedge \vec{x}_2 \wedge \ldots \wedge \vec{x}_{10})$. Namely, the size of user attribute set and the size of access policy increase simultaneously. Meanwhile, other parameters remain the same as in Table 5. We can see that the performance of Setup and $\mathsf{UKeyGen}_{\mathrm{AA}}$ is independent of the sizes of user attribute set and access policy. When we consider the average time cost of Encrypt, CTUpdate and Decrypt, the X-axis is the size of access policy. In other cases it is the size of user attribute set. Fig. 4 indicates that the size of access policy has a small influence on the performance of Decrypt, and has a significant and nearly identical affect on the average time cost of Encrypt and CTUpdate. The performance of $\mathsf{DKeyGen}_{\mathrm{ID}_1}$ and $\mathsf{DKeyGen}_{\mathrm{ID}_k}$ sightly depends on the size of user attribute set. Particularly, the average running time of $\mathsf{UKeyGen}_{\mathrm{ID}_k}$ and SKeyGen is seriously constrained by the size of user attribute set.

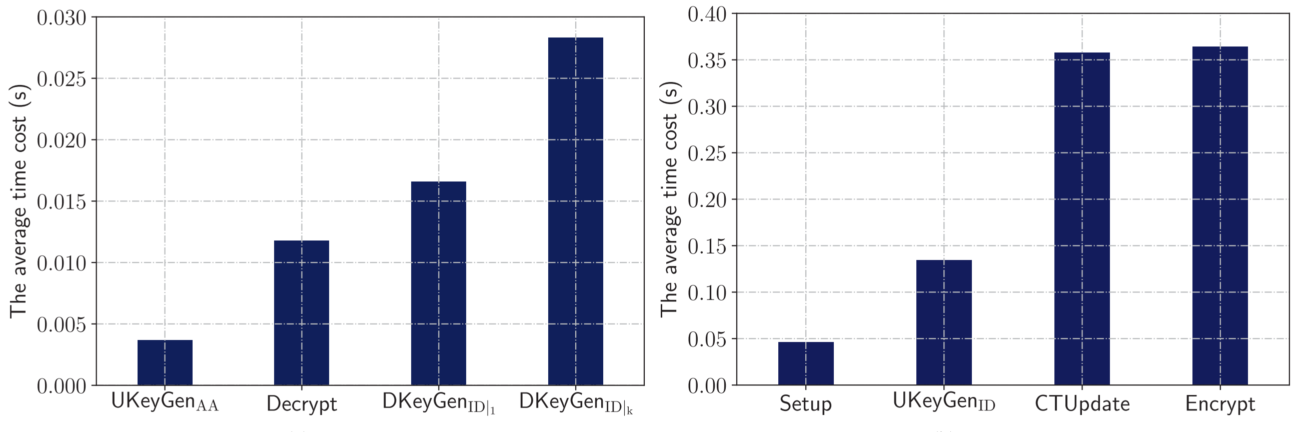
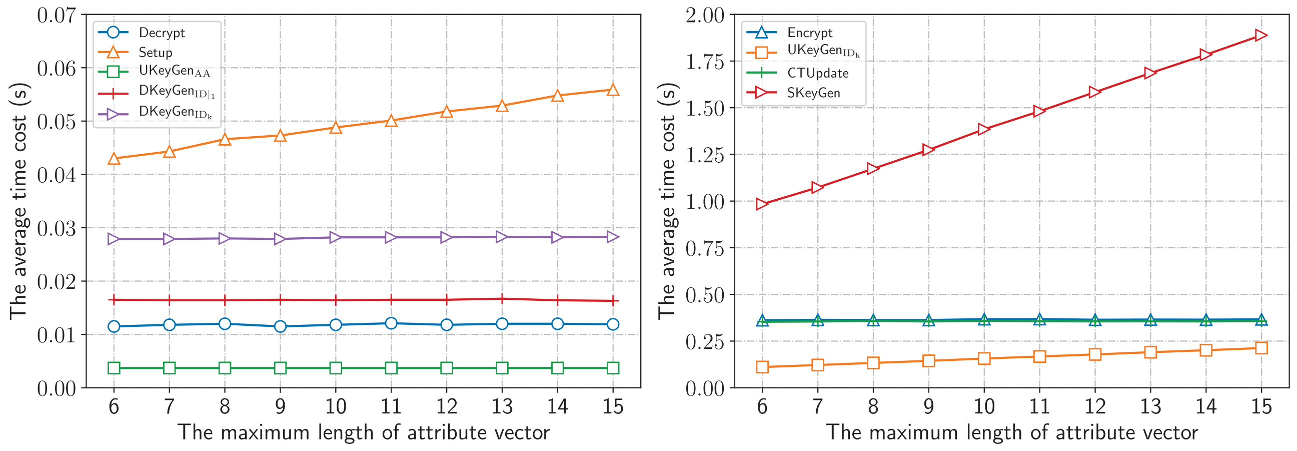
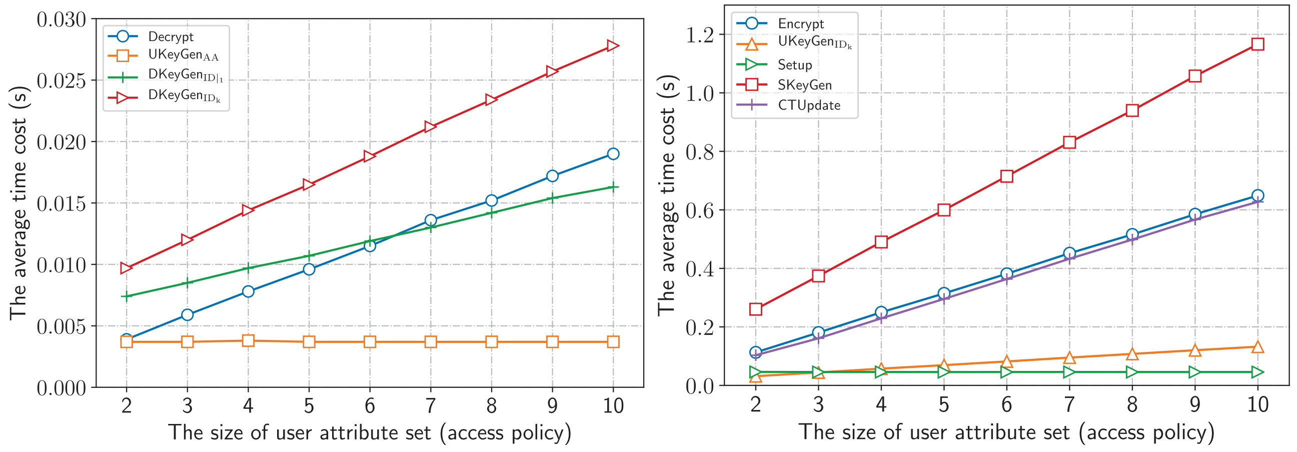


Fig. 2. The average running time of several algorithms in the proposed RS-HABE scheme.



(a) Part 1 (b) Part 2

Fig. 3. The average running time of the proposed RS-HABE scheme under the different choice of the maximum length of attribute vector.



(a) Part 1 (b) Part 2

Fig. 4. The average running time of the proposed RS-HABE scheme under the different choice of the size of user attribute set (access policy).

In Fig. 5, we show how the number of revoked users affects the performance of update key generation algorithms $\mathsf{UKeyGen}_{\mathrm{ID}_k}$ and $\mathsf{UKeyGen}_{\mathrm{AA}}$. To this end, we let the number of revoked users rang from 0 to 950, and the difference between two adjacent numbers be 50. We can see that the revocation method used in these two algorithms scales well.

## 6 Conclusion

In this paper, to overcome several practical security issues (i.e., secret key delegation, user revocation and ciphertext update) that occur when utilizing CP-ABE to secure sharing of EHR data in public cloud, we introduce a new cryptographic primitive named RS-HABE, based on which we further present an access framework for secure sharing of EHR data. Furthermore, we put forward a concrete construction of RS-HABE to instantiate the framework. The proposed RS-HABE scheme can guarantee the forward and backward security of the encrypted EHR data, meanwhile, delegates each user to generate secret keys for his/her own children. We prove the selective security of the proposal in the standard model, and provide the theoretical analysis to show its advantages in terms of functionality and security. We implement the proposed RS-HABE scheme and evaluate its practical performance. On the whole, the proposed RS-HABE scheme is more desirable for securing EHR sharing in the public cloud.

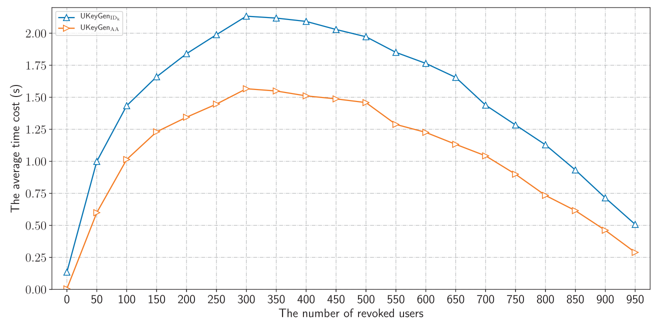


Fig. 5. The average running time of $\mathsf{UKeyGen}_{\mathrm{ID}_k}$ and $\mathsf{UKeyGen}_{\mathrm{AA}}$ under the different number of revoked users.

## Acknowledgments

This work was supported in part by the National Nature Science Foundation of China under Grants 61702549, 61960206014, 61822202 and 61872449, and in part by the Open Foundation of State Key Laboratory of Integrated Services Networks (Xidian University) under Grant ISN19-12.

## References

[1] [Online]. Available: http://www.healthcareitnews.com/news/hacker-patient-data-500000-children-stolen-pediatricians

[2] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2005, pp. 457–473.

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

[4] R. Lu, X. Lin, and X. Shen, “SPOC: A secure and privacy-preserving opportunistic computing framework for mobile-healthcare emergency,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 3, pp. 614–624, Mar. 2013.

[5] J. Liu, X. Huang, and J. K. Liu, “Secure sharing of personal health records in cloud computing: Ciphertext-policy attribute-based signcryption,” *Future Gener. Comput. Syst.*, vol. 52, pp. 67–76, 2015.

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

[7] M. H. Au et al., “A general framework for secure sharing of personal health records in cloud system,” *J. Comput. System Sci.*, vol. 90, pp. 46–62, 2017.

[8] A. Sahai, H. Seyalioglu, and B. Waters, “Dynamic credentials and ciphertext delegation for attribute-based encryption,” in *Proc. Annu. Cryptology Conf.*, 2012, pp. 199–217.

[9] J. A. Akinyele et al., “Charm: A framework for rapidly prototyping cryptosystems,” *J. Cryptographic Eng.*, vol. 3, no. 2, pp. 111–128, 2013.

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

[11] [Online]. Available: https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Expanded_Top_Ten_Big_Data_Security_and_Privacy_Challenges.pdf

[12] D. He, N. Kumar, H. Shen, and J.-H. Lee, “One-to-many authentication for access control in mobile pay-tv systems,” *Sci. China Inf. Sci.*, vol. 59, no. 5, 2016, Art. no. 052108.

[13] D. Wang, D. He, P. Wang, and C.-H. Chu, “Anonymous two-factor authentication in distributed systems: Certain goals are beyond attainment,” *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 4, pp. 428–442, Jul./Aug. 2015.

[14] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, “New algorithms for secure outsourcing of modular exponentiations,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2386–2396, Sep. 2014.

[15] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, “New publicly verifiable databases with efficient updates,” *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 5, pp. 546–556, Sep./Oct. 2015.

[16] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, “Verifiable computation over large database with incremental updates,” *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 3184–3195, Oct. 2016.

[17] J. Wang, X. Chen, X. Huang, I. You, and Y. Xiang, “Verifiable auditing for outsourced database in cloud computing,” *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3293–3303, Nov. 2015.

[18] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, “An efficient privacy-preserving outsourced calculation toolkit with multiple keys,” *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2401–2414, Nov. 2016.

[19] S. Narayan, M. Gagné, and R. Safavi-Naini, “Privacy preserving EHR system using attribute-based infrastructure,” in *Proc. ACM Workshop Cloud Comput. Security Workshop*, 2010, pp. 47–52.

[20] S. Yu, C. Wang, K. Ren, and W. Lou, “Attribute based data sharing with attribute revocation,” in *Proc. 5th ACM Symp. Inf. Comput. Commun. Security*, 2010, pp. 261–270.

[21] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, “DAC-MACS: Effective data access control for multiauthority cloud storage systems,” *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1790–1801, Nov. 2013.

[22] K. Yang and X. Jia, “Expressive, efficient, and revocable data access control for multi-authority cloud storage,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1735–1744, Jul. 2014.

[23] J. Hur and D. K. Noh, “Attribute-based access control with efficient revocation in data outsourcing systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 7, pp. 1214–1221, Jul. 2011.

[24] N. Attrapadung and H. Imai, “Attribute-based encryption supporting direct/indirect revocation modes,” in *Proc. IMA Int. Conf.*, 2009, pp. 278–300.

[25] J. K. Liu, T. H. Yuen, P. Zhang, and K. Liang, “Time-based direct revocable ciphertext-policy attribute-based encryption with short revocation list,” in *Proc. Int. Conf. Appl. Cryptography Netw. Security*, 2018, pp. 516–534.

[26] X. Liang, X. Li, R. Lu, X. Lin, and X. Shen, “An efficient and secure user revocation scheme in mobile social networks,” in *Proc. Global Telecommun. Conf.*, 2011, pp. 1–5.

[27] Y. Shi, Q. Zheng, J. Liu, and Z. Han, “Directly revocable key-policy attribute-based encryption with verifiable ciphertext delegation,” *Inf. Sci.*, vol. 295, pp. 221–231, 2015.

[28] K. Lee, S. G. Choi, D. H. Lee, J. H. Park, and M. Yung, “Self-updatable encryption: Time constrained access control with hidden attributes and better efficiency,” in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Security*, 2013, pp. 235–254.

[29] K. Lee, “Self-updatable encryption with short public parameters and its extensions,” *Designs Codes Cryptography*, vol. 79, no. 1, pp. 121–161, 2016.

[30] H. Cui, R. H. Deng, Y. Li, and B. Qin, “Server-aided revocable attribute-based encryption,” in *Proc. Eur. Symp. Res. Comput. Security*, 2016, pp. 570–587.

[31] Y. Yang, J. Liu, Z. Wei, and X. Huang, “Towards revocable fine-grained encryption of cloud data: Reducing trust upon cloud,” in *Proc. Australas. Conf. Inf. Security Privacy*, 2017, pp. 127–144.

[32] S. Xu, G. Yang, and Y. Mu, “Revocable attribute-based encryption with decryption key exposure resistance and ciphertext delegation,” *Inf. Sci.*, vol. 479, pp. 116–134, 2019.

[33] B. Qin, Q. Zhao, D. Zheng, and H. Cui, “(Dual) server-aided revocable attribute-based encryption with decryption key exposure resistance,” *Inf. Sci.*, vol. 490, pp. 74–92, 2019.

[34] G. Wang, Q. Liu, J. Wu, and M. Guo, “Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers,” *Comput. Security*, vol. 30, no. 5, pp. 320–331, 2011.

[35] Z. Wan, J. E. Liu, and R. H. Deng, “HABBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing,” *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 743–754, Apr. 2012.

[36] X. Liu, J. Ma, J. Xiong, Q. Li, T. Zhang, and H. Zhu, “Threshold attribute-based encryption with attribute hierarchy for lattices in the standard model,” *IET Inf. Security*, vol. 8, no. 4, pp. 217–223, 2014.

[37] E. Luo, Q. Liu, and G. Wang, “Hierarchical multi-authority and attribute-based encryption friend discovery scheme in mobile social networks,” *IEEE Commun. Lett.*, vol. 20, no. 9, pp. 1772–1775, Sep. 2016.

[38] Q. Huang, Y. Yang, and M. Shen, “Secure and efficient data collaboration with hierarchical attribute-based encryption in cloud computing,” *Future Gener. Comput. Syst.*, vol. 72, pp. 239–249, 2017.

[39] H. Deng et al., “Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts,” *Inf. Sci.*, vol. 275, pp. 370–384, 2014.

[40] J. Wei, X. Huang, W. Liu, and X. Hu, “Cost-effective and scalable data sharing in cloud storage using hierarchical attribute-based encryption with forward security,” *Int. J. Foundations Comput. Sci.*, vol. 28, no. 07, pp. 843–868, 2017.

[41] D. Boneh, X. Boyen, and E.-J. Goh, “Hierarchical identity based encryption with constant size ciphertext,” in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2005, pp. 440–456.

[42] A. Beimel, “Secure schemes for secret sharing and key distribution,” Technion-Israel Inst. Technol., Faculty Comput. Sci., Haifa, Israel, 1996.

[43] A. Boldyreva, V. Goyal, and V. Kumar, “Identity-based encryption with efficient revocation,” in *Proc. 15th ACM Conf. Comput. Commun. Security*, 2008, pp. 417–426.

[44] J. H. Seo and K. Emura, “Revocable hierarchical identity-based encryption: History-free update, security against insiders, and short ciphertexts,” in *Proc. Cryptographers Track RSA Conf.*, 2015, pp. 106–123.

[45] R. Canetti, S. Halevi, and J. Katz, “A forward-secure public-key encryption scheme,” *J. Cryptology*, vol. 20, no. 3, pp. 265–294, 2007.

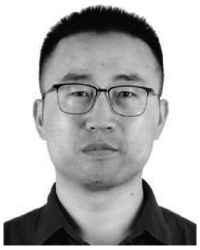
[46] X. Boyen, H. Shacham, E. Shen, and B. Waters, “Forward-secure signatures with untrusted update,” in *Proc. 13th ACM Conf. Comput. Commun. Security*, 2006, pp. 191–200.

[47] B. Waters, “Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization,” in *Proc. Int. Workshop Public Key Cryptography*, 2011, pp. 53–70.

[48] J. Hong, K. Xue, and W. Li, "Comments on DAC-MACS: Effective data access control for multiauthority cloud storage systems/ security analysis of attribute revocation in multiauthority data access control for cloud storage systems," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1315–1317, Jun. 2015.

[49] X. Wu, R. Jiang, and B. Bhargava, "On the security of data access control for multiauthority cloud storage systems," *IEEE Trans. Services Comput.*, vol. 10, no. 2, pp. 258–272, Mar./Apr. 2017.

[50] J. H. Seo and K. Emura, "Revocable identity-based encryption revisited: Security model and construction," in *Proc. Int. Workshop Public Key Cryptography*, 2013, pp. 216–234.



**Jianghong Wei** received the PhD degree in information security from the Zhengzhou Information Science and Technology Institute, Zhengzhou, China, in 2016. He is currently a lecturer in State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China. His research interests include applied cryptography and network security.

**Xiaofeng Chen** (SM'16) received the BS and MS degrees in mathematics from Northwest University, China, in 1998 and 2000, respectively, and the PhD degree in cryptography from Xidian University, in 2003. Currently, he works at Xidian University as a professor. His research interests include applied cryptography and cloud computing security. He has published more than 200 research papers in refereed international conferences and journals. His work has been cited more than 7000 times at Google Scholar. He is in the Editorial Board of the *IEEE Transactions on Dependable and Secure Computing*, Security and Privacy, and Computing and Informatics (CAI) etc. He has served as the program/general chair or program committee member in more than 30 international conferences. He is a senior member of the IEEE.

**Xinyi Huang** received the PhD degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently a professor with the College of Mathematics and Informatics, Fujian Normal University, China. His research interests include cryptography and information security. He has published more than 160 research papers in refereed international conferences and journals, such as ACM CCS, the *IEEE Transactions on Computers*, the *IEEE Transactions on Parallel and Distributed Systems*, and the *IEEE Transactions on Information Security and Forensics*. His work has been cited more than 6200 times at Google Scholar. He is in the editorial Board of International Journal of Information Security and Science China Information Sciences. He has served as the program/general chair or program committee member in more than 120 international conferences.

**Xuexian Hu** received the PhD degree in information security from the Zhengzhou Information Science and Technology Institute, Zhengzhou, China, in 2009. He is a lecturer with the State Key Laboratory of Mathematical Engineering and Advanced Computing. His current research interests include applied cryptography, network security.

**Willy Susilo** (SM'01) received the PhD degree in computer science from the University of Wollongong, Australia. He is a professor and head of School of Computing and Information Technology and the director of Institute of Cybersecurity and Cryptology (iC$^2$) at the University of Wollongong. He was awarded the prestigious Australian Research Council (ARC) Future fellow. His main research interests include cryptography and information security. His main contribution is in the area of digital signature schemes. He has served as a program committee member in dozens of international conferences. He has published numerous publications in the area of digital signature schemes and encryption schemes. He is a senior member of the IEEE.