Privacy-Preservation Enhanced and Efficient Attribute-Based Access Control for Smart Health in Cloud-Assisted Internet of Things

Hui Yin, Yin Zhu, Hua Deng, Lu Ou, Zheng Qin, and Keqin Li, *Fellow, IEEE*

***Abstract*—The deep integration of Internet of Things (IoT) and cloud computing promotes a wide deployment of body area networks (BANs) for smart health services. The data security raises new challenges when patients' health records (HRs) are uploaded into the cloud server by BAN. The attribute-based encryption (ABE) primitive is a potential option to ensure HRs security, which provides the data confidentiality guarantee and fine-grained access control simultaneously via cryptographic means. However, most ABE schemes are unsuitable to be deployed in smart health application as access policies associated with encrypted HRs reveal patient's privacies. Though the recently proposed ABE with partially hidden access policy based on composite order can alleviate the privacy leakage by only disclosing the attribute names and concealing the practical attribute values, the exposed attribute names still leak individual privacies. In this article, we put forward a privacy-enhanced and efficient ABE construction with fully hidden access policy over prime order group based on the prominent ABE construction due to Bethencourt et al.. Our scheme hides the sensitive attributes in the access structure by several nontrivial designs without compromising the correctness and security. Moreover, our scheme's performance is far superior to the attribute partially hidden schemes. Extensive experiments demonstrate the conclusion.**

***Index Terms*—Access control, attribute-based encryption (ABE), cloud computing, Internet of Things (IoT), privacy protection, smart health.**

I. Introduction

THE DEEP integration of Internet of Things (IoT) and cloud computing has driven the increasing usage and deployment of body area networks (BANs), which plays crucial role in modern smart health-care field. In the wireless BAN, various wearable sensors are implanted into a patient's body to monitor and collect key vital signs (e.g., temperature, heart rate, etc.) periodically. Via an IoT gateway, these data organized as the patient's health records (HRs) are uploaded to the cloud server for the promising and smart health-care service application. Conveniently, an authorization doctor is able to provide a seasonable, personalized, and optimizing diagnosis by accessing the HRs from the cloud platform. Despite being such a powerful technical integration serving to modern smart health-care service, the data security raises new challenges when patients' HRs are uploaded into the semi-trusted cloud server as HRs contain numerous individual privacies. It is well known encrypting data before upload is an effective measure to ensure data security against the curious cloud server or even the malicious attacker. However, traditional approaches, including the full-fledged public-key (e.g., RSA) and private-key (e.g., AES) encryption always work in the all-or-nothing way and lack of the fine-grained access control of encrypted data. For example, as existed sensitive information in HRs, a patient suffering from a massive heart disease does not wish her encrypted HRs to be decrypted by anyone except for a specified cardiologist. Here, an encrypted HRs should possess some ability that can control who has privilege to decrypt the ciphertext.

Attribute-based encryption (ABE) may be a prospective cryptographic tool for building indestructible smart health-care applications since it can provide the data confidentiality guarantee and fine-grained access control simultaneously via cryptographic instrumentation. For instance, a patient uses an ABE scheme to encrypt his HRs gathered from BAN with an access policy "(Heart Hospital: Cleveland Clinic) AND (Department: Myocarditis) AND (Doctor: Prof. David Hilbert)," which indicates that the encrypted HRs can only be decrypted by Prof. David Hilbert from the department of Myocarditis at Cleveland Clinic. Generally speaking, ABE schemes can be categorized into two major factions: 1) ciphertext-policy ABE (CP-ABE) [1] and 2) key-policy ABE (KP-ABE) [2]. CP-ABE attaches the access policy to the ciphertext and embeds a set of attributes into the private key, but KP-ABE intentionally exchanges their positions to construct the ciphertext and the private key. The common functional aspects of these two systems are that the private key can decrypt the ciphertext if and only if the attribute set satisfies the access policy. As opposed to KP-ABE, CP-ABE is more suitable for the smart health-care application scenario since the HRs owner needs to establish an access policy to specify the *special* decryptor. However,

Received 30 July 2024; revised 22 September 2024; accepted 25 September 2024. Date of publication 30 September 2024; date of current version 25 December 2024. This work was supported in part by the National Key Research and Development Program of China under Project 2022YFB3103500 and Project 2022YFB3103504, and in part by the National Natural Science Foundation of China under Grant 62372067 and Grant 62372068. *(Corresponding author: Yin Zhu.)*

Hui Yin, Yin Zhu, and Hua Deng are with the College of Computer Science and Engineering, Changsha University, Changsha 410022, China (e-mail: 463420739@qq.com).

Lu Ou and Zheng Qin are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410008, China.

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA.

Digital Object Identifier 10.1109/JIOT.2024.3470891

though the HRs themselves have been randomized, actually most AEB ciphertexts are leaking patients' privacy information to the cloud server by the clear access policy containing sensitive information. For example, from HRs ABE ciphertexts with policy "(HEART HOSPITAL: CLEVELAND CLINIC) AND (DEPARTMENT: MYOCARDITIS)," the cloud server can infer that this is a myocarditis patient. This obviously violates individual privacy.

To mitigate the privacy leakage from access policies, Lai et al. [3] proposed the commonsense viewpoint that attribute values are much more sensitive than attribute names and realized an expressive CP-ABE construction with partially hidden access structures such that the above policy can be expressed as "(HEART HOSPITAL: ***) AND (DEPARTMENT: ***) AND (DOCTOR: ***)." Zhang et al. [4] improved the decryption performance of this scheme by introducing a decryption test and developed it to a large universe construction. Hiding the attribute values makes the apparent privacy information "this is a myocarditis patient who is being treated by the famous doctor Prof. David Hilbert at Cleveland Clinic" invisible to the cloud server. The privacy leakage issue can be indeed improved greatly, however, such a partially hidden access structure still leaks the privacy "the patient is suffering from a heart illness" through the attribute name "HEART HOSPITAL." Therefore, the first motivation of this article is that if attribute names can also be hidden, the privacy issue incurred by the access policy can be eliminated thoroughly. On the other hand, both Lai et al.'s and Zhang et al.'s schemes are constructed over the composite order group, in which the computation complexity is prohibitively expensive to some resource-constrained IoT devices. Though we can find several ABE constructions over prime order with partially hidden access structures [5], [6], [7], they support only **AND**-gate policy leading to a limited expression in practice. The second motivation is that if the expressive and policy-hiding ABE can be constructed over the prime order group, the practical performance requirements can be achieved.

In this article, we intend to answer the following question: Can we construct an expressive CP-AEB scheme with fully hidden access structure over prime order group that achieves stronger privacy protection and the highly desirable performance in practice?

Contribution: we make an affirmative response to the above question and propose an expressive CP-ABE construction with fully hidden access policy over prime order group based on the prominent CP-ABE due to Bethencourt et al. [8]. Our scheme is with highly desirable features in practice that simultaneously: 1) fully hides both attribute names and attribute values in the access policy, such as "(*** AND ***) OR (*** AND ***)"; 2) is realized on the prime order group; 3) supports any monotone access structure; and 4) puts no restriction on size of attribute value set (large universe construction).

In addition, In order to achieve the real deployment (executable program) in smart health-care applications, we present an effective algorithm to solve all authorized subset (Ψ_T) for any given access policy T. In practice, without Ψ_T the decryption algorithm cannot find an exact attribute set in the private key satisfying the attribute-hiding access policy T. We implement our scheme and several similar works, and extensive experiments demonstrate our scheme is more efficient. To the best of our knowledge, this is the first expressive CP-ABE built on the prime order group with fully hidden access policy, which can be deployed in the real cloud-assisted IoT applications, such as the smart health-care service.

II. RELATED WORK

Recently, the ABE primitive has been sufficiently researched due to the prominent ability of being able to enforce fine-grained access control over encrypted data. It provides a promising measure to realize secure data sharing and utilization in the Internet of everything era.

By generalizing identities in the identity-based encryption as attributes, Sahai and Waters [9] initiated the ABE construction, whose ciphertext and private key components *bind* a set of attributes, respectively; magically, if the cardinality of the intersection of the those two sets is greater than or equal to a preset threshold value, the private key can decrypt the ciphertext. Owing to the limited expression of only threshold access policies, Goyal et al. [2] and Bethencourt et al. [8] designed more flexible and expressive ABE constructions that support any monotone access formula consisting of **AND**, **OR**, and threshold gates by introducing the access tree structure. Ostrovsky et al. [10] further improved the expressivity of the access policy and developed the new ABE construction supporting any access formula over attributes, including non-monotonic access structures, such that the important the **NOT** gate (express certain attribute is not present) is added into the ABE system.

Besides boolean formula and access tree, a more general class called monotone span program (MSP) is regarded to be more suitable to the design of ABE construction [1], [3]. An MSP is described by a $n \times l$ matrix M with a function ρ mapping each row of M to an attribute. In practice, (M, ρ) can be instantiated using a linear secret sharing scheme (LSSS).

On the mathematical background side, modern ABE schemes are generally constructed over the bilinear elliptic curve group with either prime order or composite order. The prime group-based schemes are computationally more efficient but relatively weaker security, and composite order schemes [3], [4], [11], [12], [13] are on the contrary. In ABE, paring operations linear to the number of attributes are a key reason to lead to the performance decline. Recent several works [14], [15], [16] have made efforts to optimize the decryption efficiency by greatly reducing the time-consuming pairing operations and make ABE systems more practical.

According to the position that the access policy is attached to, ABE schemes are classified into two categories: 1) CP-ABE [1], [8], [17], [18] and 2) KP-ABE [2], [10], [19]. CP-ABE attaches the access policy on the ciphertext and embeds a set of attributes into the private key, but KP-ABE intentionally exchanges their positions to construct the ciphertext and the private key. The common functional aspects between these two systems are that the private key can decrypt the ciphertext if and only if the attribute set satisfies the access

policy. Based on the traditional ABE constructions above, recently the researchers proposed a variety of functionally enhanced ABE, such as the ABE with shared decryption [20], the ABE with accountability [21], the revocable ABE [22], etc.

Most existing ABE schemes reveal the clear attributes from the access policy, which usually contain individual privacy information. The privacy protection requirement leads us to study the ABE construction where the attributes in the access policy are hidden from the public. Lai et al. [3] proposed the commonsense viewpoint that attribute values are much more sensitive than attribute names and realized an expressive CP-ABE construction with the partially hidden access policy [3], where only generic attribute names are revealed. Zhang et al. [4] improved their scheme's decryption overhead by introducing decryption test and developed it to a large universe construction. Actually, we can use a predicate encryption [23], [24], [25] to construct an ABE scheme with the fully hidden access policy. However, supporting only threshold policies in predicate encryption limits the expressivity of access structures. Moreover, since those policy-hiding ABE schemes are built on the composite order group, the expensive computation overhead may be unacceptable for some lightweight applications. Several efficient and partially hidden ABE schemes [5], [6], [7] were realized on the prime order group, but they support only **AND**-gate policy leading to a limited expression in practice.

III. Preliminaries

A. Bilinear Map

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups of prime order q and g be a generator of group $\mathbb{G}$. There exists a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ that is a bilinear map if e is efficiently computable and satisfies the following two properties: 1) $\forall a, b \in G$ and $x, y \in \mathbb{Z}_p$, $e(a^x, b^y) = e(a, b)^{xy}$ and 2) $e(g, g) \neq 1$, then we say e is a bilinear pairing map over groups $\mathbb{G}$ and $\mathbb{G}_T$.

B. Access Structure

Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure is a collection $\mathbb{A}$ of nonempty subsets of $\{P_1, P_2, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets. In ABE, parties are represented by attributes.

C. Linear Secret Sharing Scheme

A LSSS Π over a set of parties $\mathcal{P}$ is called linear if 1) the shares for each party form a vector over $\mathbb{Z}_p$ and 2) there exists an $l \times n$ matrix A. For all $i = 1, \ldots, l$, the i^{th} row of A is labeled by a party $\rho(i)$, where ρ is a function from $\{1, \ldots, l\}$ to $\mathcal{P}$. When we consider the column vector $v = (s, r_2, \ldots, r_n)$, where the secret $s \in \mathbb{Z}_p$ to be shared and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then Av is the vector of l shares of the secret s according to Π. The share $(Av)_i$ belongs to party $\rho(i)$.

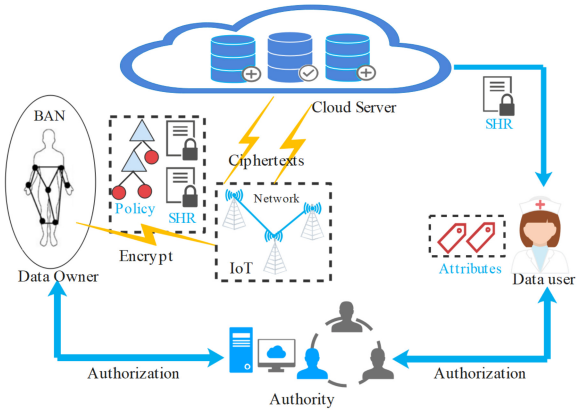


Fig. 1. System model.

In ABE, suppose that $S \in \mathbb{A}$ is an authorized set and we define the subset $I = \{i : \rho(i) \in S\} \subseteq \{1, 2, \ldots, l\}$ to be an authorized set satisfying (A, ρ). For any I, we can find constants $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ such that $\sum_{i \in I} w_i \lambda_i = s$ holds, for any valid shares $\{\lambda_i\}$ of the secret s.

In this article, we first convert an access tree T into the LSSS matrix (A, ρ) by the standard approach presented in [18]. Then, we propose an effective algorithm to find out all authorized sets $I_1, I_2, \ldots,$ from (A, ρ) and further write notation Ψ_T to denote set $\{I_1, I_2, \ldots, \}$.

IV. Problem Formulations

A. System Model

Fig. 1 shows the system model which consists of five entities, i.e., the data owner, the cloud server, the data user, the authority, and the IoT gateway. The data owner, such as a patient collects data by BAN, and encrypts them using an ABE scheme. The encrypted data via IoT gateway is uploaded to the cloud server. On the other hand, the authority is in charge of the system key management and user authorization. When a data user, such as a doctor joins in the system, the authority issues to the data user a secret key associated with the data user's attribute set. The data user is able to use the secret key to access the encrypted BAN data stored in the cloud server if his attribute set satisfies the access policy in the ciphertext. In our system, data confidentiality against the cloud server and data access control against the illegal data user can be guaranteed by ABE system, and privacy protection of data owner against the cloud server can be achieved by the fully hidden access policy.

B. Adversary Model

In essence, in this work we just hide the attribute information (including attribute names and attribute values) in the access structure by several nontrivial designs for privacy protection without changing the basic algorithm framework of [8]. Therefore, the selective security over generic bilinear group model is still applicable to our scheme. Consequently, the data security can be naturally guaranteed, and the collusion attack does not occur due to the collusion-resistance property proved in that work.

Our goal is to fully hide the attribute information to prevent the cloud server from inferring user's privacy by peeping at the access structure such that the cloud server is assumed to be honest-but-curious, who would comply with the legitimate data storage and computational commitments but may be curious to infer as many user's privacies as possible.

V. Adapted Access Tree

In this section, we present an adapted access tree, where the leaf nodes reveal nothing versus the partially hidden structures in [3] and [4], where the attribute categories are exposed.

We first introduce serval notations describing the access tree. The notation $\mathtt{num}_x$ is used to denote the number of children of a node x, and x's threshold value is defined as $1 \le k_x \le \mathtt{num}_x$, which presents the type of gate, for example the values 1 and $\mathtt{num}_x$ denote the OR ($\vee$) and AND ($\wedge$) gate, respectively; in particular, if x is a leaf node, then $k_x = 1$. On the other hand, in order to ensure our scheme to be structurally successful when revealing nothing, we redefine the function $\mathtt{index}$ as follows. If x is a nonleaf node, $\mathtt{index}(x)$ is identical to the original access tree in [8]; otherwise, we define the other function $\mathtt{index}'(x) = \pi(u_i) = i$, where u_i denotes the underlying attribute category associated with the x node and π is a function mapping an attribute category to the corresponding subscript, which will be defined in the next section. For example, if a leaf node x associates the attribute $(u_3 : v)$, we have $\mathtt{index}'(x) = \pi(u_3) = 3$, where v is a current attribute value corresponding to the attribute category u_3. Additional, we define $\mathtt{index}(x) = 0$ if x is a leaf node and $\mathtt{index}'(x) = 0$ if x is a nonleaf node.

Note that given a leaf node and a nonleaf node with a common parent, there exists a possible conflict that these two nodes have a same index value due to existing two functions $\mathtt{index}(\cdot)$ and $\mathtt{index}'(\cdot)$. If the conflict appears, the decryption algorithm may fail determined by the type of the gate of their parent node. For example, we assume there is an **AND**-gate node that has one leaf node and two nonleaf nodes denoted by x, y, z from left to right. Obviously $\mathtt{index}(y) = 2, \mathtt{index}(z) = 3$ according to our definition, if x's index value is $\mathtt{index}'(x) = \pi(u_3) = 3$ exactly, the conflict appears, which will lead to the decryption fail as two points do not satisfy the **AND**-gate. We solve the conflict by letting index values of other nodes increment by 1 repeatedly until all index values are different. In this example, when the conflict occurs $\mathtt{index}'(x) = 3$ we compute $\mathtt{index}(y) = 2 + 1 + 1 = 4, \mathtt{index}(z) = 3 + 1 + 1 = 5$.

Fig. 2 shows an example of an access tree of boolean formula ((SS#:"*123-45-6789*" **OR** (Affiliation: "*Park Hospital*" **AND** Occupation:"*Cardiologist*")), where $\vee$ and $\wedge$ denote **OR** and **AND** gate, respectively, and the attribute name universal set is $\{u_1 = \text{Country}, u_2 = \text{SS\#}, u_3 = \text{Affiliation}, u_4 = \text{Occupation}\}$.

VI. Construction

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups of prime order p, equipped with the bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. As usual, g denotes a generator of group $\mathbb{G}$. We write $\triangle_{a,S}(x) = \prod_{b \in S, a \neq b} ([x-b]/[a-b])$ as the Lagrange coefficient so that $\triangle_{a,S}(0) = \prod_{b \in S, a \neq b} (-b/[a-b])$, where $a \in \mathbb{Z}_p$ and $S \subseteq \mathbb{Z}_p$. Let $\mathcal{U} = \{u_1, \ldots, u_n\}$ be the attribute category universe. Define an injective function $\pi(\cdot)$ mapping an attribute category to its subscript, i.e., $\pi(u_i) : u_i \in \{u_1, \ldots, u_n\} \to i \in \{1, \ldots, n\}$. Our construction consists of *Setup*, *Encryption*, *Key Generation*, and *Decryption* 5 algorithms, which are described in detail as follows.

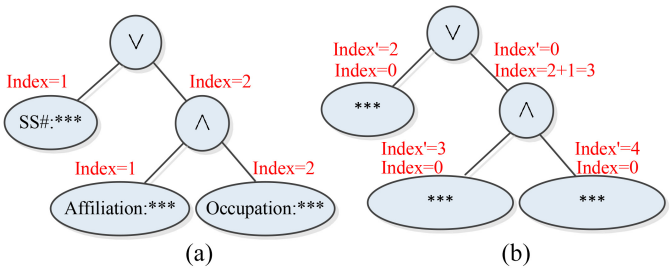


Fig. 2. (a) Access structure hiding the attribute values while exposing the attribute categories. (b) Our access structure hiding both attribute values and attribute categories.

A. Setup

The *Setup* algorithm first puts up the system running environment $(\mathbb{G}, \mathbb{G}_T, e, p, g, \pi)$ under a security parameter k. Next, it chooses two random elements $\alpha, \beta \in \mathbb{Z}_p$ and computes $\varphi = e(g,g)^\alpha, \lambda = g^\beta$. Given a set of attribute values $\mathcal{V} = \{v_{\pi(u_i)} | i \in I\}$ corresponding to attribute names $\mathcal{U}' = \{u_i | i \in I\} \subseteq \mathcal{U}$ where $I \subseteq \{1, 2, \ldots, n\}$, define two hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p$ mapping a string of arbitrary length to an element in $\mathbb{G}$ and $\mathbb{Z}_p$, respectively. The system public key is $(\varphi, \lambda, H_1, H_2)$ and (α, β) is kept secret as the master key.

B. Encryption

The *Encryption* algorithm is responsible for encrypting a message $M \in \mathbb{G}_T$ under the public key and an access tree T. For each node t in T, we first generate two polynomials $q_t(x) = a_0 + a_1 x + \cdots + a_{d_t} x^{d_t}$ and $q'_t(x) = a'_0 + a'_1 x + \cdots + a'_{d'_t} x^{d_t}$, where $a_1, \ldots, a_{d_t}$ and $a'_1, \ldots, a'_{d'_t}$ are randomly chosen from group $\mathbb{G}$ and the degree d_t, d'_t are set to be $d_t = d'_t = k_t - 1$. If t is a leaf node, the corresponding polynomials are $q_t(x) = a_0$ and $q'_t(x) = a'_0$ due to $k_t = 1$. Now, we determine a_0 and a'_0 starting with the root node R of T as follows. The algorithm chooses two random exponents μ and μ' and sets

$$q_t(0) = a_0 = \begin{cases} \mu & t \text{ is the root node } R \\ q_{\mathtt{parent}(t)}(\mathtt{index}(t)) & t \text{ is a non-leaf node} \\ q_{\mathtt{parent}(t)}(\mathtt{index}'(t)) & t \text{ is a leaf node} \end{cases}$$

$$q'_t(0) = a'_0 = \begin{cases} \mu' & t \text{ is the root node } R \\ q'_{\mathtt{parent}(t)}(\mathtt{index}(t)) & t \text{ is a non-leaf node} \\ q'_{\mathtt{parent}(t)}(\mathtt{index}'(t)) & t \text{ is a leaf node.} \end{cases}$$

Let $\mathcal{V} = \{v_{\pi(u_i)} | i \in I\}$ be the underlying attribute value set of leaf nodes in access tree T, where $\{u_i\}_{i \in I}$ is the corresponding attribute name set. For ease of description, we use the same notation u as the attribute name to denote the node in access

tree, and the algorithm uses the public key $(\varphi, \lambda, H_1, H_2)$ to encrypt message M as follows.

$$CT_1 = \Big(C_1 = \varphi^{\mu} = Me(g,g)^{\alpha\mu}, C_2 = \lambda^{\mu} = g^{\beta\mu} \quad \forall v \in \mathcal{V} : C_{\pi(u)} = g^{q_u(0)} \quad C'_{\pi(u)} = H_1(v_{\pi(u)})^{q_u(0)} H_1(v_{\pi(u)})^{q_u(0)H_2(v_{\pi(u)})} \Big)$$

$$CT_2 = \Big(\widehat{C}_1 = \varphi^{\mu'} = e(g,g)^{\alpha\mu'}, \widehat{C}_2 = \lambda^{\mu'} = g^{\beta\mu'} \quad \forall v \in \mathcal{V} : \widehat{C}_{\pi(u)} = g^{q'_u(0)} \quad \widehat{C}'_{\pi(u)} = H_1(v_{\pi(u)})^{q'_u(0)} H_1(v_{\pi(u)})^{q'_u(0)H_2(v_{\pi(u)})} \Big)$$

M's ciphertexts consist of three components (T, CT_1, CT_2), in which the actual attribute values are hidden in ciphertext components $C'_{\pi(u)}$ and $\widehat{C}'_{\pi(u)}$ by binding a random exponent $t_{\pi(u)}$ for corresponding attribute name u. CT_2 can be seen as an encryption of 1 and is used in *Decryption* algorithm to implicitly discovery an attribute set satisfying the access tree.

C. Key Generation

Given an attribute value set $S = \{s_{\pi(u_i)} | i \in I\}$, where $\{u_i\}_{i\in I}$ is the corresponding attribute name set, the algorithm generates the private key with the help of the master key (α, β). It first chooses a random $\gamma \in \mathbb{Z}_p$ and computes $g^{\alpha/\beta} g^{\gamma/\beta}$. Then, for each attribute $s_{\pi(u)} \in S$, the algorithm chooses a random γ_s and computes $g^{\gamma} H_1(s_{\pi(u)})^{\gamma_s(1+H_2(s_{\pi(u)})}, g^{\gamma_s}$. The private key with respect to S is denoted as

$$SK = \Big(K = g^{\alpha/\beta} g^{\gamma/\beta} \quad \forall s \in S : K_{\pi(u)} = g^{\gamma} H_1(s_{\pi(u)})^{\gamma_s(1+H_2(s_{\pi(u)})} \widehat{K}_{\pi(u)} = g^{\gamma_s} \Big).$$

D. Decryption

If the attribute set in private key satisfies the access tree in ciphertext, the *decryption* algorithm can recover message M without having any knowledge of attribute information.

The most critical challenge is how to find an exact attribute subset in private key satisfying the access tree without attribute information. An obvious observation is that while the attribute subset cannot be determined on the fly in the process of the decryption, we know it is certainly in the set Ψ_T. Therefore, the first problem necessary to solve is to compute Ψ_T from an attribute-hiding access tree T. In this article, we propose an effective algorithm to gain Ψ_T, which is described in Algorithm 1. The main idea behind of this algorithm is to first convert the access tree T into an equivalent LSSS matrix A and then solve the general solution of the linear equation system $Au = b$, where $b^{\top} = (1, 0, \ldots, 0)$ (lines 1–7 in Algorithm 1). Obviously, the general solution summarizes all constant sets that satisfy $\sum_{1\le i\le n} A_i t_i = (1, 0, \ldots, 0)$ in the LSSS system, where A_i is the ith row of LSSS matrix A, t_i denotes the ith component of the general solution, and n is the number of attributes in T. Next, we need to retain those attributes corresponding to nonzero components to form a set when setting a proper constant t. All such sets make up Ψ_T (lines 8–17 in Algorithm 1). Figs. 3 and 4 demonstrate an example of Algorithm 1, where x, y, z, m denote four attributes in an access tree. We can employ a standard approach to convert an access tree into an LSSS matrix in the literature [1]. Note that, in our scheme we let x, y, z, m be symbolically equivalent to a leaf node as well as an attribute name, respectively, the algorithm actually returns $\Psi_T = \{\{\pi(x), \pi(y), \pi(z), \pi(m)\}, \{\pi(x), \pi(m)\}, \{\pi(x), \pi(y), \pi(z)\}\}$, in which both attribute names and attribute values are invisible to the cloud server.

With the help of Ψ_T, we can use the ciphertext CT_1 to determine an exact attribute subset in private key satisfying the access tree (lines 2–18 in Algorithm 2), and further employ the attribute subset to recover the message from ciphertext CT_2 (lines 19–16 in Algorithm 2) in the attribute-hiding manner.

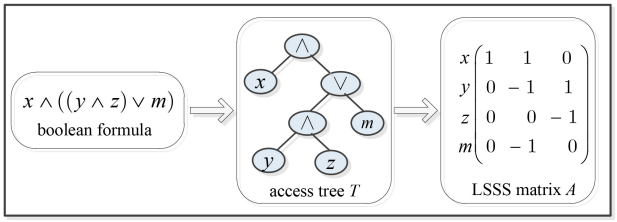


Fig. 3. Example to convert an access tree to an LSSS matrix.

Algorithm 1 Solution of All Subset Satisfying an Access Tree

Input:
Access tree T.
Output:
The set Ψ_T of all attribute subset satisfying the access tree T.
1: Convert T into an LSSS matrix A.
2: Define a variable vector $\mathbf{u} = (u_1, \ldots, u_n)^{\top}$.
3: Transpose matrix A to obtain A's transposed matrix $A^{\top}$.
4: Solve system of linear homogeneous equations $A^{\top}\mathbf{u} = \mathbf{b}$, where $\mathbf{b}$ is a zero vector $(0, \ldots, 0)^{\top}$ of dimension m.
5: Present the general solution of $A^{\top}\mathbf{u} = \mathbf{b}$ as $(u_1, \ldots, u_n)^{\top} = (a_1, \ldots, a_n)^{\top} \cdot t$, where t is a constant integer.
6: Solve a particular solution $(s_1, \ldots, s_n)$ of system of linear inhomogeneous equations $A^{\top}\mathbf{u} = \mathbf{b}'$, where $\mathbf{b}'$ is the vector $(1, 0, \ldots, 0)^{\top}$ of dimension n.
7: Write the general solution of $A^{\top}\mathbf{u} = \mathbf{b}'$ as $(u_1, \ldots, u_n)^{\top} = (a_1, \ldots, a_n)^{\top} \cdot t + (s_1, \ldots, s_n)^{\top} = (a_1 t + s_1, \ldots, a_n t + s_n)^{\top}$.
8: Write $(a_1 t + s_1, \ldots, a_n t + s_n)$ as $(u_1 = a_1 t + s_1, \ldots, u_n = a_n t + s_n)$ and set $\Psi_T = \emptyset$.
9: **if** exist some 0-components $u_j, \ldots, u_k (1 \le j < \ldots < k \le n)$ in $(u_1 = a_1 t + s_1, \ldots, u_n = a_n t + s_n)$ **then**
10: Compute set $I = \{u_1, \ldots, u_n\} - \{u_j, \ldots, u_k\}$ and add I into Ψ_T.
11: **else**
12: Add set $I = \{u_1, \ldots, u_n\}$ into Ψ_T.
13: **end if**
14: **repeat**
15: Find a integer t such that making some components $u_j, \ldots, u_k (1 \le j < \ldots < k \le n)$ equal to 0.
16: Compute set $I = \{u_1, \ldots, u_n\} - \{u_j, \ldots, u_k\}$.
17: Add I into Ψ_T.
18: **until** (such a integer t exists no longer)
19: **return** Ψ_T.

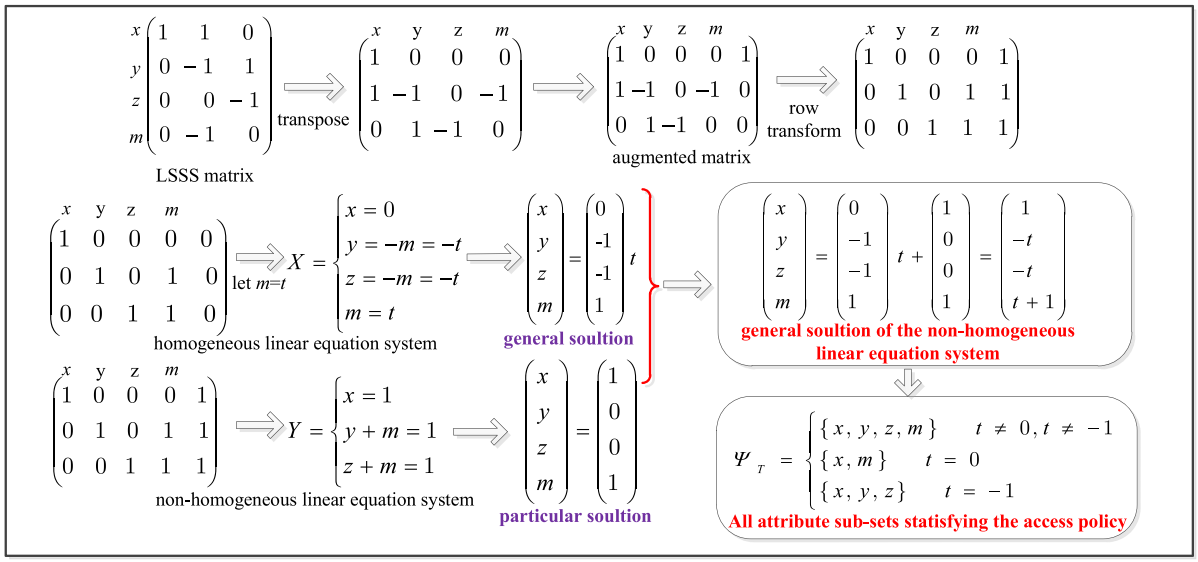


Fig. 4. Example to solve Ψ_T.

VII. Correctness and Security Analysis

A. Correctness Analysis

Given the ciphertexts (CT_1, CT_2) with the attribute-hiding access tree T, we first run Algorithm 1 to calculate set Ψ_T. We show that running codes lines 2–18 in Algorithm 2 can find an exact attribute subset in private key satisfying T if the subset exists in Ψ_T. Without loss of generality, we suppose that $I = \{\pi(u)_{u\in U} | U \subseteq \mathcal{U}\} \in \Psi_T$ is such one. For each leaf node u, we compute

$$\begin{aligned}\widehat{f}_u &= \frac{e\left(\widehat{C}_{\pi(u)}, K_{\pi(s)}\right)}{e\left(\widehat{C}'_{\pi(u)}, \widehat{K}_{\pi(u)}\right)} \\ &= \frac{e\left(g^{q'_u(0)}, g^{\gamma}\cdot g^{\gamma} H_1\left(s_{\pi(u)}\right)^{\gamma_s(1+H_2\left(s_{\pi(u)}\right)}\right)}{e\left(H_1\left(v_{\pi(u)}\right)^{q'_u(0)} H_1\left(v_{\pi(u)}\right)^{q'_u(0)H_2\left(v_{\pi(u)}\right)}, g^{\gamma_s}\right)} \\ &= \frac{e\left(g^{q'_u(0)}, g^{\gamma}\right) e\left(g^{q'_u(0)}, H_1\left(s_{\pi(u)}\right)^{\gamma_s}\right)}{e\left(H_1\left(v_{\pi(u)}\right)^{q'_u(0)}, g^{\gamma_s}\right)} \\ &\quad \frac{e\left(g^{q'_u(0)}, H_1\left(s_{\pi(u)}\right)^{\gamma_s H_2\left(s_{\pi(u)}\right)}\right)}{e\left(H_1\left(v_{\pi(u)}\right)^{q'_u(0)H_2\left(v_{\pi(u)}\right)}, g^{\gamma_s}\right)} \\ &= e(g,g)^{q'_u(0)\gamma} \quad \left(\text{impliedly } s_{\pi(u)} = v_{\pi(u)}\right). \end{aligned} \tag{1}$$

For each nonleaf node u (here, we also use the notation u to label the nonleaf node for description uniformity), we result f_u by recursively calculating f_z for $u's$ every child node z. Let S_u be an arbitrary k_u-sized set of child nodes z such that $f_z \neq \perp$, we classify the nodes in S_u into two sets: 1) nonleaf node set denoted by S'_u, and 2) leaf node set denoted by S''_u, then $S_u = S'_u \cup S''_u$. We compute

$$\begin{aligned}\widehat{f}_u &= \prod_{z\in S'_u\cup S''_u} f_z^{\Delta_{i,\widehat{S}_u\cup\widetilde{S}_u}(0)} \\ &= \prod_{z\in S'_u\cup S''_u} \left(e(g,g)^{\gamma\cdot q'_z(0)}\right)^{\Delta_{i,\widehat{S}_u\cup\widetilde{S}_u}(0)} \\ &= \prod_{z\in S'_u\cup S''_u} \left(e(g,g)^{\gamma\cdot q'_{parent(z)}(\mathtt{index}(z)+\mathtt{index}'(z))}\right)^{\Delta_{i,\widehat{S}_u\cup\widetilde{S}_u}(0)} \\ &= \prod_{z\in S'_u\cup S''_u} \left(e(g,g)^{\gamma\cdot q'_x(i)}\right)^{\Delta_{i,\widehat{S}_u\cup\widetilde{S}_u}(0)} \\ &= e(g,g)^{\gamma\cdot\sum_{i\in\widehat{S}_u\cup\widetilde{S}_u} q'_u(i)\cdot\left(\Delta_{i,\widehat{S}_u\cup\widetilde{S}_u}(0)\right)} \\ &= e(g,g)^{\gamma\cdot q'_u(0)} \end{aligned} \tag{2}$$

where $i = \mathtt{index}(z) + \mathtt{index}'(z)$, $\widehat{S}_u = \{\mathtt{index}(z) + \mathtt{index}'(z) : z \in S'_u, \mathtt{index}'(z) = 0\}$ if z is the nonleaf node and $\widetilde{S}_u = \{\mathtt{index}'(z) + \mathtt{index}(z) : z \in S''_u, \mathtt{index}(z) = 0\}$ if z is the leaf node.

Since $I = \{\pi(u)_{u\in U} | U \subseteq \mathcal{U}\} \in \Psi_T$ implicitly satisfies the access tree T, according to (1) and (2), we can get $\widehat{f}_R = e(g,g)^{q'_R(0)\gamma} = e(g,g)^{\gamma\mu'}$ by recursive computations until root node R. The algorithm further verifies

$$\begin{aligned} e(\widehat{C}_2, K)/\widehat{f}_R &= \frac{e\left(g^{\beta\mu'}, g^{\alpha/\beta} g^{\gamma/\beta}\right)}{e(g,g)^{\gamma\mu'}} = \frac{e\left(g^{\mu'}, g^{\alpha} g^{\gamma}\right)}{e(g,g)^{\gamma\mu'}} \\ &= e(g,g)^{\alpha\mu'} = \widehat{C}_1. \end{aligned}$$

With the set $I = \{\pi(u)_{u\in U} | U \subseteq \mathcal{U}\}$, similarly, we can recursively compute $f_R = e(g,g)^{q_R(0)\gamma} = e(g,g)^{\gamma\mu}$ and decrypt the ciperhtext CT_1 as $C_1 \cdot (e(C_2, K)/f_R)^{-1} = M$.

Example: We give a concrete example to help to understand how (2) work, as shown in Fig. 5. The access tree contains three leaf nodes A, B, and C corresponding the underlying attribute values x, y, and z, respectively, where $[\cdot]$ represents our hidden structure of an attribute value, which includes two cryptographic hash functions and an exponentiation operation over group $\mathbb{G}$. Here, we only demonstrate the computation process of f_u of the slightly complex case that u is a nonleaf node and $u's$ children contain both nonleaf and leaf nodes. In this example, the root node R is such a node whose children contain a nonleaf node F and a leaf node C. According to the encryption algorithm, we set the polynomial associated with the root node R to be $q_R(x) = \mu + a_1 x$ due to $R's$ threshold value being $k_R = 2$ such that the degree of the corresponding

Algorithm 2 Decryption

Input:

Ciphertext $CT = (T, CT_1, CT_2)$, Key SK

Output:

Message M

Run Algorithm 1 to calculate set Ψ_T on input the access tree T.

$i := 1, find := 0, obj := null$

repeat

  Let $I_i = \{\pi(u)_{u\in U}\} \in \Psi_T, U \subseteq \mathcal{U}$;

  for each leaf node $u \in U$ **do**

    Calculate

$$\widehat{f}_u = \frac{e(\widehat{C}_{\pi(u)}, K_{\pi(u)})}{e(\widehat{C}'_{\pi(u)}, \widehat{K}_{\pi(u)})}$$

  end for

  Similarly, let notation u be a nonleaf node and S_u denote its child node set.

  Recursively calculate $\widehat{f}_u = \prod_{z\in S'_u\cup S''_u} f_z^{\Delta_{i,\widehat{S}_u\cup\widetilde{S}_u}(0)}$ until root node R, where $i = \mathtt{index}(z) + \mathtt{index}'(z)$, $\widehat{S}_u = \{\mathtt{index}(z) + \mathtt{index}'(z) : z \in S'_u, \mathtt{index}'(z) = 0\}$ if z is the nonleaf node and $\widetilde{S}_u = \{\mathtt{index}'(z) + \mathtt{index}(z) : z \in S''_u, \mathtt{index}(z) = 0\}$ if z is the leaf node.

  if $e(\widehat{C}_2, K)/\widehat{f}_R == \widehat{C}_1$ **then**

    $find := 1$;

    $obj := (I_i = \{\pi(u)_{u\in U}\}, U \subseteq \mathcal{U})$

  end if

  $i++$;

until $((i > |\Psi_T|)||(find == 1))$

if $obj == null$ **then**

  return $\perp$

end if

Use set obj to recovery message M as follows.

for each leaf node $u \in U$ **do**

  Calculate

$$f_u = \frac{e(C_{\pi(u)}, K_{\pi(u)})}{e(C'_{\pi(u)}, K'_{\pi(u)})}$$

end for

Let u be a nonleaf node and S_u denote its child node set.

Recursively calculate $f_z = \prod_{z\in S'_u\cup S''_u} f_z^{\Delta_{i,\widehat{S}_u\cup\widetilde{S}_u}(0)}$ until root node R,where $i = \mathtt{index}(z) + \mathtt{index}'(z)$, $\widehat{S}_u = \{\mathtt{index}(z) + \mathtt{index}'(z) : z \in S'_u, \mathtt{index}'(z) = 0\}$ if z is the nonleaf node and $\widetilde{S}_u = \{\mathtt{index}'(z) + \mathtt{index}(z) : z \in S''_u, \mathtt{index}(z) = 0\}$ if z is the leaf node.

Compute

$$C_1 \cdot (e(C_2, K)/f_R)^{-1} = M$$

return M.

polynomial is $d_R = k_R - 1 = 1$. Since F is a nonleaf node and C is a leaf node, we have $\mathtt{index}(F) = 1$, $\mathtt{index}'(F) = 0$ and $\mathtt{index}'(C) = \pi(u_3) = 3, \mathtt{index}(C) = 0$ such that $q_F(0) = q_{parent(F)}(\mathtt{index}(F)+\mathtt{index}'(F)) = q_R(1) = \mu+a_1$ and $q_C(0) = q_{parent(C)}(\mathtt{index}'(C) + \mathtt{index}(C)) = q_R(3) = \mu + 3a_1$. We define the complete polynomials for F and C as $q_F(x) = \mu + a_1 + rx$ and $q_C(x) = \mu + 3a_1$, where r is a random value. Let $f_F = e(g, g)^{q_F(0)\gamma}$ [calculating from nodes A and B by (2)] and $f_C = e(g, g)^{q_C(0)\gamma}$ [calculating by (1)]. We compute

$$\begin{aligned}
f_R &= \prod_{z\in\{F\}\cup\{C\}} f_z^{\Delta_{i,\{\mathtt{index}(F)+\mathtt{index}'(F)\}\cup\{\mathtt{index}(C)+\mathtt{index}'(C)\}}(0)} \\
&= \prod_{z\in\{F,C\}} \left(e(g, g)^{\gamma\cdot q_z(0)}\right)^{\Delta_{i,\{\mathtt{index}(F),\mathtt{index}'(C)\}}(0)} \\
&= \prod_{z\in\{F,C\}} \left(e(g, g)^{\gamma\cdot q_R(\mathtt{index}(z)+\mathtt{index}'(z))}\right)^{\Delta_{i,\{1,3\}}(0)} \\
&= \prod_{z\in\{F,C\}} \left(e(g, g)^{\gamma\cdot q_R(i)}\right)^{\Delta_{i,\{1,3\}}(0)} \\
&= e(g, g)^{\gamma\cdot(q_R(1)\cdot(-3)/(1-3)+q_R(3)\cdot(-1)/(3-1))} \\
&= e(g, g)^{\gamma\cdot((\mu+a_1)\cdot 3/2-(\mu+3a_1)\cdot 1/2)} \\
&= e(g, g)^{\gamma\cdot\mu} = e(g, g)^{\gamma\cdot q_R(0)}. \qquad (3)
\end{aligned}$$

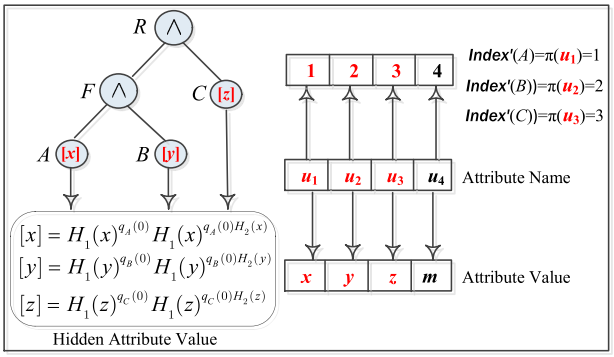


Fig. 5. Example of the decryption computation.

B. Security Analysis

Essentially, our scheme uses the same encryption framework for a message as the original construction in [8] other than a real attribute value is hidden in our special ciphertext component. The security guarantee is that the attacker has no capacity to recover the indispensable decryption element $e(g, g)^{\gamma\mu}$ even in the collusion attack environment. Therefore intuitively, our scheme is sucre under the generic bilinear group model, where the hash function H_1 is modeled as the random oracle.

To achieve the protection of attributes and ensure the correctness of ABE scheme, we derive a special ciphertext component mentioned above, for example, which can be written as $C_u = H_1(v)^{q_u(0)}H_1(v)^{q_u(0)H_2(v)}$, where u is a leaf node in an access policy T and v is the attribute value hidden in u. As a result, we must answer the following question: Can a polynomial-time adversary recover attribute information v from the ciphertext component C_u? We give the formal security proof to demonstrate that any polynomial-time adversary cannot obtain v from the ciphertext construction C_u if decisional Diffie-Hellman (DDH) assumption holds.

Theorem 1: Our proposed ciphertext component C_u hiding the attribute information v is semantically secure against chosen-plaintext keyword attack (CPA) if DDH assumption

holds, where H_1 is modeled as the random oracle and H_2 is a one-way hash function.

Proof: Assuming a PPT adversary can break C_u to recover the underlying attribute information v with a nonnegligible advantage ε, we can construct a simulator $\mathcal{B}$ who can solve the DDH problem with a nonnegligible advantage $(\epsilon/2)$.

The challenger $\mathcal{C}$ first flips a binary coin μ. If $\xi = 0$, $\mathcal{C}$ sets tuple $t_0 : (g, A = g^a, B = g^b, C = g^{ab})$; if $\xi = 1$, he sets tuple $t_1 : (g, A = g^a, B = g^b, C = g^c)$, where $a, b, c,$ are chosen from $\mathbb{Z}_q$ at random uniformly. Tuple t_μ is sent to simulator $\mathcal{B}$. The simulator $\mathcal{B}$ plays the following game with adversary $\mathcal{A}$ on behalf of challenger $\mathcal{C}$.

Setup: $\mathcal{B}$ sends the public parameter $(\mathbb{G}, \mathbb{G}_T, g, q, e, H_1, H_2)$ to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ accesses the encryption construction C_u many times using arbitrary attribute values to ask corresponding ciphertext. When $\mathcal{A}$ calls for the evaluation of H_1 on any attribute value v, a new random value v' is chosen (unless it has already been used), and a value $g^{v'}$ as the response to $H_1(v)$ [8]. Finally, he outputs two values v_1 and v_2 corresponding to attribute name u_1 and u_2 and sends them to $\mathcal{B}$.

Challenge: $\mathcal{B}$ flips a binary coin γ and encrypts v_γ as $C_{u_\gamma} = (C)^{H_2(v_\gamma)}$.

If $\mu = 0$, $C = g^{ab}$. According to the polynomial definition in the encryption, for each node t in the subtree of the root node R, $q_t(0)$ must contain the term μ for each node, including any leaf node. Recall that since μ is chosen in random at the rood node R, $q_t(0)$ is also random. We use notation $q'_{u_\gamma}(0)$ to denote the random value for $q_{u_\gamma}(0)$. As $q'_{u_\gamma}(0)$ is random, the value $([q'_{u_\gamma}(0)]/[H_2(v_\gamma)])+q'_{u_\gamma}(0)$ is also random. Since a, b are chosen from $\mathbb{Z}_q$ at random uniformly, we let $([q'_{u_\gamma}(0)]/[H_2(v_\gamma)]) + q'_{u_\gamma}(0) = ab$, Thus, we have

$$C_{u_\gamma} = (C)^{H_2(v_\gamma)} = \left(g^{\frac{q'_{u_\gamma}(0)}{H_2(v_\gamma)}+q'_{u_\gamma}(0)}\right)^{H_2(v_\gamma)} = g^{q'_{u_\gamma}(0)} g^{q'_{u_\gamma}(0)H_2(v_\gamma)}.$$

As both $q'_{u_\gamma}(0)$ and $q_{u_\gamma}(0)$ is random, let $q'_{u_\gamma}(0) = v'_\gamma q_{u_\gamma}(0)$. Recall that when accessing the random oracle H_1 for v_r, the simulation outputs $g^{v'_\gamma}$ as response of $H_1(v_\gamma)$. Therefore, we have $C_{u_\gamma} = H_1(v_\gamma)^{q_{u_\gamma}(0)} H_1(v_\gamma)^{q_{u_\gamma}(0)H_2(v_\gamma)}$, which is a valid ciphertext for attribute value v_γ.

If $\mu = 1$, $C = g^c$. Then we have $C_{u_\gamma} = g^{cH_2(v_\gamma)}$. Since c is a random element, therefore C_{u_γ} is a random element in $\mathbb{G}$ from $\mathcal{A}$'s perspective and contains no information about v_γ.

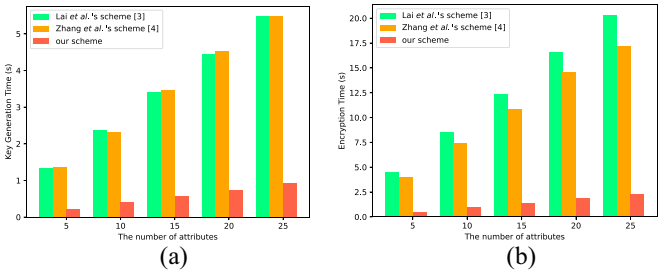
Phase 2: Repeat Phase 1.

Guess: $\mathcal{A}$ outputs a guess γ' of γ. If $\gamma' = \gamma$, then $\mathcal{B}$ outputs the guess $\xi' = 0$ of ξ. This means $\mathcal{C}$ sent the valid encryption tuple $t_0 : (g, A = g^a, B = g^b, C = g^{ab})$ to $\mathcal{B}$. Since $\mathcal{A}$ has advantage ϵ to break C_u, therefore, the probability $\mathcal{A}$ outputs guess γ' of γ satisfying $\gamma' = \gamma$ is $(1/2)+\epsilon$. Correspondingly, the probability that $\mathcal{B}$ outputs guess ξ' of ξ satisfying $\xi' = \xi = 0$ is $(1/2)+\epsilon$. If $\gamma' \neq \gamma$, then $\mathcal{B}$ outputs the guess $\xi' = 1$ of ξ. This means random tuple t_1 was sent to $\mathcal{B}$. Therefore, the probability $\mathcal{A}$ outputs guess γ' of γ satisfying $\gamma' = \gamma$ is $(1/2)$. Correspondingly, the probability that $\mathcal{B}$ outputs guess ξ' of ξ satisfying $\xi' = \xi = 1$ is $(1/2)$.

Hence, the overall advantage that $\mathcal{B}$ solves the DDH problem can be computed:

$$\left|\frac{1}{2}Pr[\xi = \xi'|\xi = 0] + \frac{1}{2}Pr[\xi = \xi'|\xi = 1] - \frac{1}{2}\right| = \left|\left[\frac{1}{2}\left(\frac{1}{2}+\epsilon\right) + \frac{1}{2}\cdot\frac{1}{2}\right] - \frac{1}{2}\right| = \frac{\epsilon}{2}$$

Since ϵ is nonnegligible, therefore $(\epsilon/2)$ is also nonnegligible. This conclusion means $\mathcal{B}$ is able to solve the DDH problem with a nonnegligible advantage, which contradicts DDH problem assumption. ■



(a) (b)

Fig. 6. (a) Key generation time and (b) encryption time when varying the number of attributes.

VIII. Experimental Evaluation

We implement our construction as well as the similar schemes in [3] and [4] for the convictive performance compares. All programs are implemented in Java environment relying on the Java pairing-based cryptography library[1] in which *Type A* pairing is used for our scheme and *Type A1* for the other two schemes. All experiments are tested at a Windows 10 System with 3.30-GHZ Inter Core i7-11370H CPU and 16-GB memory.

Fig. 6(a) and (b) illustrates the experimental evaluate results of *Key Generation* algorithm and *Encryption* algorithm. we can see that time cost of both of them linearly increases with the number of attributes. However, our scheme requires far less time in the same experiment parameters than other two works. For example, when we set the number of attributes to be 20, the key generation and encryption time of our scheme is about 0.74 and 1.79 s, while Lai et al.'s scheme needs about 4.45 and 16.8 s, and Zhang et al.'s scheme needs 4.52 and 14.2 s. Fig. 7(a) and (d) presents the time cost of running the decryption algorithm in the three schemes when varying the number of attributes in the authorized subset under the different size of the authorized subset set. The experimental results reveal that the time cost on decryption of all of the three schemes is closely related to those two parameters. Given an access policy T, the more the number of T's authorized subsets and the number of attributes in one authorized subset are, the more time cost is consumed on running decryption algorithm. However, our scheme is more suitable to deploy in the real smart health application due to the highly desirable performance cost.

[1]http://gas.dia.unisa.it/projects/jpbc/index.html

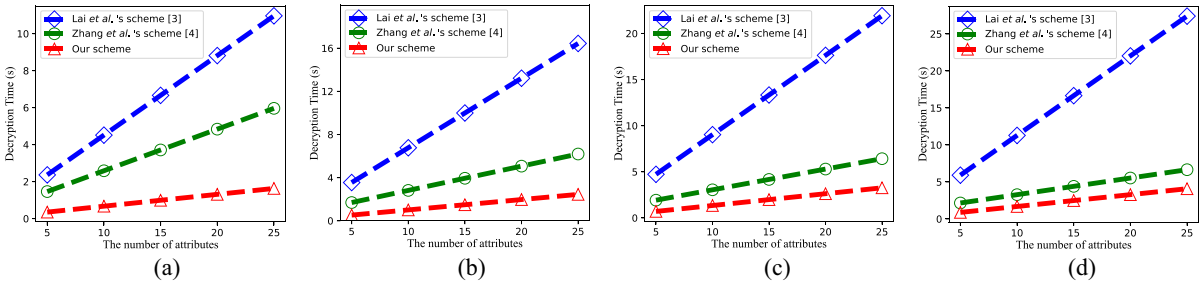


Fig. 7. Decryption time when varying the number of attributes in the authorized subset and setting different size of authorized subset set of the access policy T as (a) $|\Psi_T| = 1$, (b) $|\Psi_T| = 2$, (c) $|\Psi_T| = 3$, and (d) $|\Psi_T| = 4$.

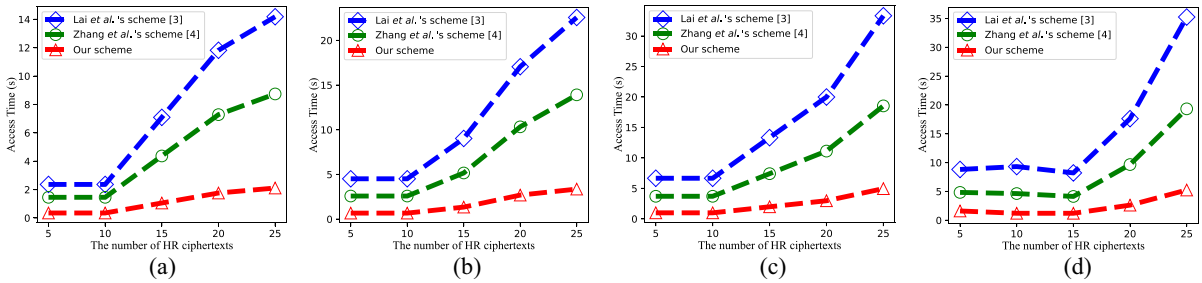


Fig. 8. HRs access time when varying the number of HR ciphertexts and setting different number of attributes in access policies I as (a) $|I| = 5$, (b) $|I| = 10$, (c) $|I| = 15$, and (d) $|I| = 20$.

We put our implementations on the real HR dataset to run experiment cases, where each symmetric key (using to encrypt HRs) is encrypted under a policy T of form "$attr_1$ AND $attr_2$ AND $\cdots$ AND $attr_n$" such that we have $|\Psi_T| = 1$. The experiment results in Fig. 8 demonstrate that the ciphertext access time cost of all of the three schemes is closely affected by the complexity of the access policy. Moreover, the more the number of attributes in access policies is, the less the number of HRs can be accessed. For instance, when the number of HR ciphertexts is 25 and the number of attributes in access policies is set as 5, 10, 15, and 20, respectively, the number of the ciphertexts can be accessed (decrypted) is 6, 5, 5, and 4, as shown in Fig. 8(a)–(d). As a comparison, our proposed scheme has more practical access performance in the real application. For example, when the number of RH ciphertexts achieves 25 and setting the number of attributes in the access policy to be 15, the access time of our scheme is 4.96 s, and Lai et.'s and Zhang et.'s schemes are 33.37 and 18.72 s, respectively, as shown in Fig. 8(c), where five RH ciphertexts are accessed under this experiment parameter.

IX. Conclusion

In this article, we propose an expressive, policy fully hiding, and efficient CP-ABE construction. Compared to the existing related works, our CP-ABE scheme bears three critical advantages that simultaneously: 1) does not leak any attribute information, including both attribute categories and attribute values; 2) is based on the faster prime order pairing group; and 3) supports any monotone access structure. We implement our construction, which can be deployed the real smart health-care application scenario for privacy-preserving and fine-grained access control on the encrypted patient HR. However, similar to the existing related works, our scheme needs also a relatively expensive test to find an exact attribute set in the private key satisfying the current access policy from all authorized subsets (Ψ_T) of the access policy by a redundant ciphertext. How to eliminate the test computation and the redundant ciphertext making it more practical will be our future work.

Acknowledgment

The authors would like to express their gratitude to the anonymous reviewers whose constructive comments have helped to improve the quality of the manuscript.

References

[1] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptogr.*, 2011, pp. 53–70.
[2] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encryption data," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.
[3] J. Lai, R. H. Deng, and Y. Li, "Expressive CP-ABE with partially hidden access structures," in *Proc. ACM Symp. Inf., Comput. Commun. Secur.*, 2012, pp. 18–19.
[4] Y. Zhang, D. Zheng, and R. H. Deng, "Security and privacy in smart health: Efficient policy-hiding attribute-based access control," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2130–2145, Jun. 2018.
[5] T. Nishide, K. Yoneyama, and K. Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structure," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2008, pp. 111–129.
[6] J. Li, K. Ren, B. Zhu, and Z. Wan, "Privacy-aware attribute-based encryption with user accountability," in *Proc. Int. Conf. Inf. Secur.*, 2009, pp. 347–362.
[7] T. V. X. Phuong, G. Yang, and W. Susilo, "Hidden ciphertext policy attribute-based encryption under standard assumptions," *IEEE Trans. Inf. Forensics Security*, vol. 11, pp. 35–45, 2016.
[8] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 321–334.
[9] A. Sahai and B. Waters, "Fuzzy identity-base encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2005, pp. 457–473.

[10] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2007, pp. 195–203.
[11] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2010, pp. 62–91.
[12] A. Lewko and B. Waters, "Unbounded HIBE and attribute-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2011, pp. 547–567.
[13] T. Okamoto and K. Takashima, "Fully secure unbounded inner-product and attribute-based encryption," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2012, pp. 349–366.
[14] Q. M. Malluhi, A. Shikfa, and V. Trinh, "A Ciphertext-policy attribute-based encryption scheme with optimized ciphertext size and fast decryption," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 230–240.
[15] S. Agrawal and M. Chase, "FAME: Fast attribute-based message encryption," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2017, pp. 665–682.
[16] D. Riepel and H. Wee, "FABEO: Fast attribute-based encryption with optimal security," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2022, pp. 2491–2504.
[17] L. Cheng and C. Newport, "Provably secure ciphertext policy ABE," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2007, pp. 456–465.
[18] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2011, pp. 547–567.
[19] M. Chase and S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2009, pp. 121–130.
[20] N. Chen, J. Li, Y. Zhang, and Y. Guo, "Efficient CP-ABE scheme with shared decryption in cloud storage," *IEEE Trans. Comput.*, vol. 71, no. 1, pp. 175–184, Jan. 2022.
[21] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh, and D. Wang, "Attribute based encryption with privacy protection and accountability for CloudIoT," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 762–773, Apr.–Jun. 2022.
[22] S. Chen, J. Li, Y. Zhang, and J. Han, "Efficient revocable attribute-based encryption with verifiable data integrity," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 10441–10451, Mar. 2024.
[23] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2008, pp. 146–162.
[24] A. D. Caro, V. Iovino, and G. Persiano, "Fully secure hidden vector encryption," in *Proc. Int. Conf. Pairing-Based Cryptogr.*, 2012, pp. 101–121.
[25] T. Okamoto and K. Takashima, "Hierarchical predicate encryption for inner-products," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2009, pp. 214–231.

Hui Yin received the M.S. degree in computer software and theory from Central South University, Changsha, China, in 2008, and the Ph.D. degree from the College of Information Science and Engineering, Hunan University, Changsha, in 2018.

He is currently a Professor with the College of Computer Science and Engineering, Changsha University, Changsha. His interests are data security, privacy protection, and applied cryptography.

Yin Zhu is currently pursuing the bachelor's degree with the College of Computer Science and Engineering, Changsha University, Changsha, China.

Her interests are attribute-based encryption and searchable encryption.

Dr. Zhu has won the First Prize and Innovation Value Award of the 15th National College Students Information Security Competition in 2022.

Hua Deng received the M.S. degree in cryptography from Southwest Jiaotong University, Chengdu, China, in 2010, and the Ph.D. degree in information security from Wuhan University, Wuhan, China, in 2015.

He is currently a Professor with the College of Computer Science and Engineering, Changsha University, Changsha, China. His research interests include applied cryptography, data security and privacy, and cloud security.

Lu Ou received the Ph.D. degree in software engineering from Hunan University, Changsha, China, in 2018.

From 2015 to 2016, she was a visiting student with the Department of Computer Science, University of Texas at Arlington, Arlington, TX, USA. She was a Postdoctoral Fellow with the College of Computer Science and Electronic Engineering, Hunan University, where she is currently an Associate Professor with the School of Journalism and Communication. Her research focuses on data security, privacy and big data, as well as signal, image, and video analysis.

Zheng Qin received the Ph.D. degree in computer software and theory from Chongqing University, Chongqing, China, in 2001.

From 2010 to 2011, he served as a Visiting Scholar with the Department of Computer Science, Michigan University, Ann Arbor, MI, USA. He is currently a Professor with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. His main interests are network and data security, privacy, machine learning, and applied cryptography.

Keqin Li (Fellow, IEEE) received the B.S. degree in computer science from Tsinghua University, Beijing, China, in 1985, and the Ph.D. degree in computer science from the University of Houston, Houston, TX, USA, in 1990.

He is a SUNY Distinguished Professor of Computer Science with the State University of New York at New Paltz, New Paltz, NY, USA. He is also a National Distinguished Professor with Hunan University, Changsha, China. He holds over 70 patents announced or authorized by the Chinese National Intellectual Property Administration. He has authored or co-authored over 850 journal articles, book chapters, and refereed conference papers. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPUCGPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent, and soft computing.

Dr. Li has received several best paper awards. He is among the World Top Five Most Influential Scientists in parallel and distributed computing in terms of both single-year impact and career-long impact based on a composite indicator of Scopus citation database. He has chaired many international conferences. He is currently an Associate Editor of the *ACM Computing Surveys* and the *CCF Transactions on High Performance Computing*. He has served on the editorial boards for the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON SERVICES COMPUTING, and the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING. He is an AAIA Fellow. He is also a member of Academia Europaea (Academician of the Academy of Europe).