IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received January 9, 2020, accepted January 14, 2020, date of publication January 20, 2020, date of current version January 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2968078

Efficient and Expressive Access Control With Revocation for Privacy of PHR Based on OBDD Access Structure

KENNEDY EDEMACU, BEAKCHEOL JANG, AND JONG WOOK KIM, (Member, IEEE)
Department of Computer Science, Sangmyung University, Seoul 03016, South Korea

Corresponding author: Jong Wook Kim (jkim@smu.ac.kr)

This work was supported in part by the National Research Foundation of Korea through the Basic Science Research Program, Ministry of Education, under Grant NRF-2017R1D1A1B03028097 and in part by Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea government (MSIT) (A Research on Safe and Convenient Big Data Processing Methods) under Grant 2018-0-00269.

ABSTRACT With the advancement of information and communication technology (ICT), the medical sector is undergoing a massive transformation. Health records are being digitized, stored remotely in the cloud and shared with different stakeholders. However, the use of the cloud for personal health record (PHR) storage presents data security and privacy challenges. Ciphertext-policy attribute-based encryption (CP-ABE) is being widely studied for fine-grained access control of PHRs in the cloud. Expressiveness, efficiency and attribute revocation, among others, are some key requirements of a cloud based health systems. But, many of the proposed CP-ABE schemes rely on access structures that are either restrictive or cumbersome and thus result in less expressive and efficient schemes. Many of the schemes also lack mechanisms for efficient and immediate attribute/user revocation. In this work, we propose an expressive and efficient access control scheme with attribute/user revocation based on ordered binary decision diagram (OBDD) access structure. We use the attribute group approach to achieve the attribute/user revocation in our work. Additionally, the ciphertexts and private keys are assigned version numbers to prevent the revoked group members from colluding with non-revoked members. Security and efficiency analysis show that our proposed scheme is secure, expressive and efficient.

INDEX TERMS Attribute-based encryption (ABE), ordered binary decision diagram (OBDD), personal health record (PHR), attribute/user revocation, security and privacy.

I. INTRODUCTION

The rapid development of information and communication technologies, in particular, the internet of things (IoT), wireless technologies and cloud computing in recent years have paved the way for interconnection of medical resources enabling improved delivery of healthcare services for patients. Digitized or electronic health records (EHR) (sometimes referred to as PHR) can now be collected from patients and sent to the cloud for analysis, diagnosis and sharing with different healthcare stakeholders. For example, suppose a patient is being simultaneously treated by two hospitals H-A and H-B for hypertension and a kidney disease, respectively. The recordings from the medical examinations conducted by H-A are stored in the cloud for sharing with H-B and vice versa. This practice of sharing health information between healthcare service providers not only reduces stress, but it also minimizes the need to carry out repeated and duplicated medical tests and examinations.

However, the use of third-parties for storage of PHRs presents multiple concerns, including security and privacy, that may hinder the wider adoption of this exciting paradigm if not addressed. One way to address the security and privacy concern is by encrypting the PHR before uploading it to the cloud. Secondly, access to the encrypted PHR should be regulated by using an access control method. Attribute-based encryption (ABE) which offers ingredients for encryption and access control, is being widely studied for fine-grained access control of PHR in the cloud [1]. ABE is a public-key encryption mechanism in which both the ciphertext and the decryption keys depend on attribute sets. There are two variants of ABE, ciphertext-policy attribute-based encryption (CP-ABE) [2] and key-policy attribute-based encryption (KP-ABE) [3]. In CP-ABE, the ciphertext is labeled with an access policy allowing the data owner to specify which users have access to his/her data while the user's key is associated with a set of

The associate editor coordinating the review of this manuscript and approving it for publication was Zheng Yan.

attributes. CP-ABE is suitable for fine-grained access control of data in cloud-based environments, including cloud-based health systems. Meanwhile, in KP-ABE, the ciphertext is labeled with attributes and the user's key is associated with an access policy. KP-ABE is suitable for pay-per-view channels.

Several studies suggesting the use of the attribute-based access control schemes for controlled access to patients' data in the cloud have been conducted [4]–[7], [34], [36]. However, most of these schemes rely on access structures that are either cumbersome or less expressive, hence, affecting the general efficiency and expressiveness of the proposed schemes. In [8], Li *et al.* suggested an efficient and expressive CP-ABE scheme based on ordered binary decision diagrams (OBDD) as an access structure. Though the scheme is expressive and efficient, it lacks a revocation mechanism. As stated in [4], [9], attribute/user revocation is one of the key security requirements for a cloud-based health system. Thus, the Li *et al.* scheme is unsuitable for the cloud-based ehealth systems.

Therefore, in this study, we construct a novel expressive access control scheme with immediate and efficient attribute/user revocation mechanism for regulating access to PHR in the cloud. We adapt and leverage the OBDD access structure [8] and the attribute revocation mechanism of [10] for the construction. As in [10], we create attribute groups whose members are users bearing the specific attributes. Each attribute group is assigned a unique group key. Whenever a user is revoked of an attribute, a new attribute group key is generated and distributed to all the group members except the revoked user. To prevent collusion between revoked and non-revoked users, version numbers are assigned to user secret keys and ciphertexts. The version number is incremented whenever there is a change in the attribute group key after an attribute/user revocation. We further carry out an experiment to show the efficiency of our scheme in terms of computation times for user key generation, encryption, re-encryption, decryption, ciphertext update, and user key update. The results show that our proposed scheme is efficient. Also, we present a security proof by reducing our scheme to the decisional bilinear diffie-hellman (DBDH) problem.

The rest of the paper is organized as follows: in section II we present the related works. Section III presents summaries of cryptographic fundamentals, access structure definitions and complexity assumptions used in this work. In section IV, we present the formal system definition and the security model. Section V presents the construction of our proposed access control scheme. The security proof and the performance evaluations are presented in sections VI and VII, respectively. Section VIII concludes the paper.

II. RELATED WORK

Regulating access to the patient's data is critical in cloud-based health systems. A patient's data is personal and sensitive, and its unregulated access can breach the security and the personal privacy of the patient (data owner). Thus, focus on the security and the personal privacy of the patient needs to be at the forefront if a wider acceptance of this paradigm is to be realized. ABE offers a promising solution to this problem.

ABE is primarily divided into two: KP-ABE and CP-ABE. Sahai and Waters did the first construction of KP-ABE [1] in the form of fuzzy identity-based encryption. Subsequently, Goyal *et al.* [3] constructed a KP-ABE scheme based on a monotonic access structure in 2006. The lack of a self-defensive mechanism in KP-ABE's ciphertexts prompted Bethencourt *et al.* [2] to propose the CP-ABE scheme.

Since then, CP-ABE scheme has attracted great attention, especially in cloud-based environments. Waters [11] proposed a CP-ABE scheme based on a flexible access structure but has a limitation on the number of occurrences of attributes. In a study by Cheung and Newport [12], AND-gate was proposed as an access structure for construction of CP-ABE schemes. The study further offered formal security proof for the CP-ABE. AND-gate access structure can represent both positive and negative attributes but prohibits repetition of attributes. Recently, Li *et al.* [8] proposed a CP-ABE scheme based on OBDD as an access structure. OBDD access structure is non-monotonic and supports the repetition of attributes. The authors in [8] further presented a CPA security proof under the DBDH security assumptions for their scheme. To improve efficiency, their scheme aggregates the attribute elements in the ciphertext and the user keys, thus resulting in difficulties in achieving efficient attribute/user revocation.

Hur and Noh [10] proposed an access control scheme with a revocation ability. They introduced the concept of attribute groups in attribute revocation. Users bearing the same attribute are made to belong to the same attribute group, and each group has a key. Once a user is revoked, a new group key is generated and sent out to all the group members except the revoked user. The ciphertext gets updated with the new group key. This makes it impossible for the revoked user to decrypt the ciphertext. However, their scheme does not prevent collusion between revoked and non-revoked users. The collusion problem was recently addressed by Li *et al.* [33]. Studies [13], [14] are some of the proposed CP-ABE schemes with revocation abilities constructed based on AND-gate as their access structures. Zhang *et al.* [7] proposed a CP-ABE scheme with hidden policies. The construction of their scheme is based on the linear secret sharing scheme (LSSS) access structure but it is computationally demanding. Additional policy-hidden studies were conducted in [15], [16] based on the AND-gate and LSSS as their access structures, respectively. Other variants of CP-ABE schemes for multiple-attribute authorities, producing ciphertexts with fixed-sizes and hierarchical attributes are suggested in [17]–[19], respectively.

From the perspective of security of PHRs, several studies have suggested the use of ABE for secure sharing of health data in cloud environments. The scheme in [4] categorizes PHR users into two domains, personal and public domains, and proposed a single authority KP-ABE scheme for the personal domain users while a multi-authority KP-ABE scheme is proposed for the public domain users. Wei *et al.* [34] proposed a revocable CP-ABE scheme with hierarchical delegation capabilities for sharing PHRs in the public cloud. However, their scheme only achieves periodic revocations. In [5], the author combined the techniques of CP-ABE and attribute-based signature [35] to construct an access control scheme that achieves unforgeability, confidentiality and authenticity of PHRs in cloud environments. However, their scheme lacks an immediate attribute/user revocation mechanism. Liang *et al.* [36] proposed a multi-authority ABE scheme in which the global identity (GID) of the user is hidden using the anonymous secret key issuing protocol for secure sharing of EHRs in the cloud. Their construction removes the random oracle from the security proof. However, it lacks an immediate attribute/user revocation mechanism and is based on a monotonic access structure. Additional CP-ABE schemes for secure sharing of health data are suggested in [20], [21]. These studies use the access matrix and threshold tree access structures, respectively, and they lack a revocation ability.

III. PRELIMINARIES

In this section, we summarize the definitions of access structure, security complexity assumption, OBDD and bilinear maps which are adapted for use in our construction.

A. ACCESS STRUCTURE

Definition 1 (Access Structure): Access structures are formal representations of access policies. There are numerous access structures being used in ABE today such as; LSSS [7], AND-gates [12], threshold gates [2], [3], OBDD [8], etc.

In relation to OBDD access structure (which is of interest in this work), an access structure is a rule $\mathbb{R}$ that satisfies a given set of attributes S, i.e., 1 is returned if S satisfies $\mathbb{R}$ ($S \models \mathbb{R}$). Otherwise 0 is returned.

B. BILINEAR MAPS

Definition 2 (Bilinear Maps): Let, $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ be two multiplicative cyclic groups of prime order p, and g be the generator of $\mathbb{G}$. A bilinear map e is defined as $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathbb{T}}$ and should satisfy the following conditions.

1) Bilinearity, i.e., $e(g^a, g^b) = e(g, g)^{ab}$ for all a and b.
2) Non-degenerate, i.e., $e(g, g) \neq 1$.
3) and is computationally feasible.

C. COMPLEXITY ASSUMPTION

Definition 3 (Decisional Bilinear Diffie-Hellman (DBDH) Assumption): Given two tuples $(g^a, g^b, g^c, e(g, g)^{abc})$ $\forall a, b, c$ random elements of $\mathbb{Z}_p$ and $(g^a, g^b, g^c, e(g, g)^z)$ $\forall a, b, c, z \in \mathbb{Z}_p$, the DBDH assumption [11] states that, the advantage of distinguishing the two tuples by a probabilistic polynomial-time algorithm $\mathcal{B}$ is negligible.

D. ORDERED BINARY DECISION DIAGRAMS (OBDD)

A binary decision diagram (BDD) [23], [24] is a data structure used to represent and manipulate boolean functions. Bryant in [25], introduced the transformation of BDD to OBDD by specifying the ordering of variables. OBDDs are computationally efficient and possess powerful descriptive abilities [26].

Definition 4 (Binary Decision Diagram (BDD)): Consider a boolean function $\mathfrak{f}(x_0, x_1, \cdots, x_{m-1})$ over a set of Boolean variables $x_m = \{x_0, x_1, \cdots, x_{m-1}\}$, a BDD over the variable set is a rooted, directed acyclic graph ($G = (V, E)$) with terminal nodes labeled as either 1 or 0 and has the following features.

1) The nodes in V are either terminal or non-terminal nodes.
2) Each terminal node in V is labeled as either 0 or 1 and has no child nodes.
3) Each non-terminal node is labeled with a variable $x_j \in x_m$ and has two out-going edges whose ends are denoted as $low(v)$ and $high(v)$. Provided, $(low(v), high(v) \in V)$.
4) Along a directed path from the root node to the terminal node, each variable appears once.

Definition 5 (Ordered Binary Decision Diagram (OBDD)): A boolean function $\mathfrak{f}(x_0, x_1, \cdots, x_{m-1})$ over the set of variables $x_m = \{x_0, x_1, \cdots, x_{m-1}\}$ is ordered, if and only if the variable order from the root node to the terminal node is fixed as π. The variable order permutation is of the form $\pi : \{0, \cdots, m-1\} \rightarrow \{x_0, \cdots, x_{m-1}\}$ [27].

E. OBDD ACCESS STRUCTURE

An OBDD access structure is generated from an access policy expressed in a natural language which is eventually transformed to a boolean function. The mathematical foundation for the construction of the OBDD is laid by Shannon's expansion theorem. Suppose an access policy is transformed to a boolean function $\mathfrak{f}(x_0, x_1, \cdots, x_{m-1})$, where m is the total number of attributes, using Shannon's expansion theorem, $\mathfrak{f}(x_0, x_1, \cdots, x_{m-1}) = x_i {\cdot} \mathfrak{f}_{|x_i=1} + x_i' {\cdot} \mathfrak{f}_{|x_i=0}$, where; $0 \leq i \leq m-1$, and $\mathfrak{f}_{|x_i=1}$ and $\mathfrak{f}_{|x_i=0}$ are co-factors of $\mathfrak{f}$. This Shannon's decomposition theorem is a recursive process having a pre-defined variable ordering $\pi : x_0 < x_1 \cdots < x_{m-1}$. This pre-defined order has direct implication on the number of nodes in the generated OBDD.

Each non-terminal node in the OBDD is a tuple with four(4) elements; $< i, id, low(v), high(v) >$, where i is the serial number of the attribute the node represents, id is a unique serial number of a node, $low(v)$ is the serial number of the $low(v)$ child node and $high(v)$ is the serial number of the $high(v)$ child node. Compacting the OBDD is done through the following conditions; 1) No more than one non-terminal node has the same name (id), and $low(v)$ and $high(v)$ vertices. 2) No node has identical $low(v)$ and $high(v)$ child nodes, i.e., $low(v) \neq high(v)$. The resulting OBDD access structure is $OBDD = \{Node^{i,low(v)}_{id,high(v)}|_{id \in ID, i \in \mathbb{A}}\}$, where, ID is the identity universe of the non-terminal nodes and $\mathbb{A}$ is

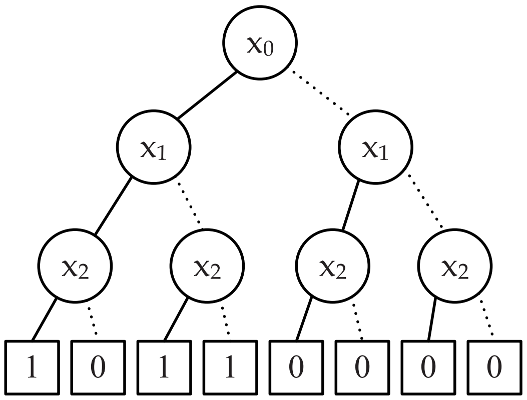


FIGURE 1. An OBDD access structure with redundancies. The pre-defined variable order used during the construction is: $x_0 < x_1 < x_2$.

the attribute universe. For terminal nodes, the elements $< i$, $low(v)$, $high(v) >$ are non-existent and thus, the nodes are removed to reduce storage cost.

OBDD Satisfiability: Given the attribute set S, starting from the root node, comparisons are made between S and the nodes in the OBDD access structure. If a value of S matches the current node's attribute value, S is forwarded to the $high(v)$ node. Otherwise, it is forwarded to the $low(v)$ node. This is done repeatedly until it is delivered to either terminal node 1 or 0. If the terminal node 1 is reached, S satisfies the OBDD access structure ($S \models OBDD$). But, if the terminal node 0 is reached, S does not satisfy the OBDD access structure ($S \not\models OBDD$).

F. EXAMPLES

An intuitive example: Suppose an access policy allows access to data for users who possess attributes x_0 and x_2 or attributes x_0 and x_1'. A boolean function $\mathfrak{f}(x_0, x_1, x_2) = x_0 \wedge (x_1' \vee x_2)$ is defined to represent the stated access policy. An initial construction of an OBDD access structure for the stated boolean function $\mathfrak{f}(x_0, x_1, x_2)$ is shown in Figure 1.

In Figure 2, we show a reduced version of Figure 1 after the elimination of redundancies. The computational and storage efficiency associated with OBDD access structure is based on this reduction. It can be seen that the number of nodes in the reduced version is much smaller as compared to the first construction in Figure 1.

A comparative example: To explicitly show the power and efficiency of the OBDD access structure, we compare the OBDD and threshold access tree access structures. The threshold access tree access structure is a monotonic access structure, and a negated variable can only be created if it is considered as an independent variable. However, the OBDD access structure integrates the AND, OR and NOT operations which prevents extra variables from being created. For instance, consider an access policy represented by the following boolean formula $\mathfrak{f}(x_0, x_1, x_2) = x_0x_1x_2 + x_0'x_2'$. Representing this access policy using the threshold access tree yields 8 nodes and 5 variables, out of which 2 variables

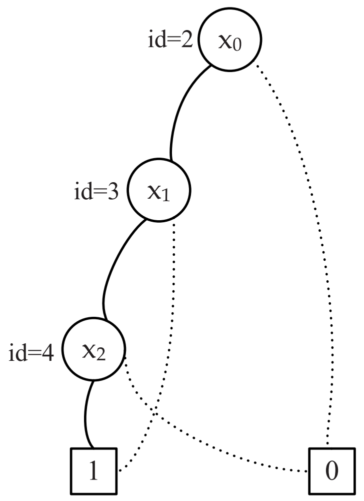


FIGURE 2. A reduced version of the OBDD access structure after removal of redundancies. The variable order is: $x_0 < x_1 < x_2$.

TABLE 1. Comparing threshold access tree and OBDD access structures.

Access structure	AND	OR	NOT	No. of variables	No. of nodes
Threshold	✓	✓	×	5	8
OBDD	✓	✓	✓	3	6

represent the negated x_0 and x_2 included in the formula. On the other hand, representing the formula using the OBDD access structure yields only 6 nodes with 3 variables. A summary of the comparisons is shown in Table 1.

IV. SYSTEM AND SECURITY MODELS

In this section, we present the system architecture, the definition of our access control scheme and the security model.

A. SYSTEM ARCHITECTURE

Depicted in Figure 3 is an illustration of our system architecture. The system consists of four entities; a trusted authority (TA), a data owner (EHR-O), data users (EHR-Us) and a cloud service provider (Cloud) which are described as follows.

- **TA:** The TA is the most trusted entity in the scheme. It initializes the system and in the process generates the necessary system parameters used by the other entities. It grants fine-grained access to data users by giving out user secret keys and key encryption keys. Similarly, it generates re-keys which are used for user key update after an attribute revocation. It also generates attribute group information which is used by the cloud for ciphertext re-encryption. Similar to the assumptions in [31], [32], we assume the TA is always online.
- **EHR-O:** The EHR-O is a data owner or a patient whose data is stored in the cloud. The data ranges from medical recordings performed in a health facility to health sensor recordings. The gathered health data is then out-sourced to the cloud inform of ciphertexts. The ciphertexts are generated based on policies defined by the EHR-O.

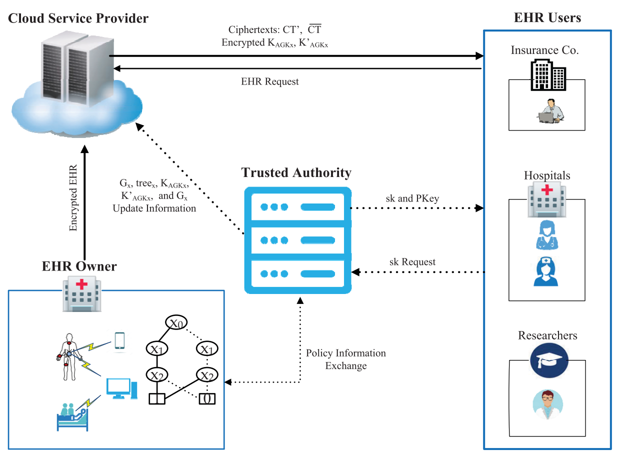

Cloud Service Provider
EHR Users
Ciphertexts: CT', $\overline{CT}$
Encrypted K_{AGKx}, K'_{AGKx}
EHR Request
Insurance Co.
Encrypted EHR
G_x, tree$_x$, K_{AGKx}, K'_{AGKx}, and G_x Update Information
Trusted Authority
sk and PKey
Hospitals
sk Request
EHR Owner
Policy Information Exchange
Researchers
X_0
X_1
X_2
0

FIGURE 3. System architecture for our access control showing the entities involved. The dashed arrows indicate the flow of key requests and responses between the entities, while the solid arrows indicate the flow of data requests and responses between entities in the access control.

- **EHR-U:** The EHR-U is a data user who accesses the encrypted data from the cloud. Examples of EHR-Us are; health practitioners (e.g., doctors, nurses, pharmacists, etc), researchers, insurance companies, relatives, friends, etc. For each EHR-U to have access to the encrypted data, his/her set of attributes need to satisfy the OBDD access policy defined by the EHR-O used during the encryption process. The EHR-U attributes are assigned by the TA and it is the same authority that can revoke the assigned attributes. Each attribute assigned to an EHR-U belongs to an attribute group and each attribute group has a unique key used in generating key encryption keys. Thus, apart from the user secret key, each user has a key encryption key for each attribute he/she possesses.
- **Cloud:** The cloud stores the ciphertexts received from the EHR-O and re-encrypts them before distributing them to the requesting EHR-Us. We assume the cloud is honest but curious i.e., it follows the algorithm the way it is but tries to gather secret information as much as possible. Also, we assume the cloud is always online.

B. THE PROPOSED ACCESS CONTROL SCHEME

In order to achieve expressiveness and efficiency with revocation, our proposed access control scheme possesses the following algorithms.

- $Setup(\lambda) \rightarrow (pp, mk)$: This algorithm is run by the TA and it only takes as input the security parameter (λ) and generates public keys (pp) and master key (mk) as its outputs.
- $UserKeyGen(pp, mk, S) \rightarrow sk$: The user key generation algorithm is run by the TA. It takes as input the public keys (pp), the master key (mk) and a set of data user attributes (S). It outputs a secret key (sk) for a data user associated with his/her attributes.
- $KEKeyGen(pp, i, K_{AGK_i}) \rightarrow KEK_i$: The algorithm is executed by the TA. It takes as input the public keys (pp), an attribute (i) managed by the TA and an attribute group key (K_{AGK_i}) associated with the attribute (i) and it outputs a key encryption key (KEK_i) for the attribute (i).
- $Encrypt(pp, M, OBDD) \rightarrow CT$: The encryption algorithm is executed by the EHR-O. It takes as input the public keys (pp), the data (M) and the $OBDD$ access structure over a given attribute set to generate a ciphertext (CT) as its output.
- $Re\text{-}Encrypt(pp, CT, GI_i) \rightarrow CT'$: The re-encryption algorithm is run by the cloud. It takes as input the public keys (pp), the ciphertext (CT) received from the EHR-O, and the attribute group information (GI_i) associated with each attribute in the OBDD access structure. It produces a re-encrypted ciphertext (CT') as its output.
- $Decrypt(CT', USK, pp) \rightarrow M/\perp$: The decryption algorithm is run by the EHR-U. It takes as input the re-encrypted ciphertext (CT'), the user key (USK) which comprises the user secret key (sk) and the key encryption key(s) (KEK_i(s)), and the public keys (pp). It recovers the encrypted data (M) if the set of data user attributes satisfy the OBDD access structure in the ciphertext. Otherwise it outputs $\perp$.
- $Re\text{-}KeyGen(pp, i, K_{AGK_i}, K'_{AGK_i}) \rightarrow (rk_{ci}, rk_{ki})$: The re-key generation algorithm is executed by the TA after a data user is revoked of an attribute (i). It takes as input the public keys (pp), the old attribute group key (K_{AGK_i}), a new attribute group key (K'_{AGK_i}) and an attribute (i) to generate re-keys (rk_{ci}, rk_{ki}) as its output.
- $CtxtUpdate(pp, i, CT', rk_{ci}) \rightarrow \overline{CT}$: This is the ciphertext update algorithm executed by the cloud whenever an attribute/user revocation happens. It takes as input the public keys (pp), the re-encrypted ciphertext (CT') and the re-key (rk_{ci}). It produces an updated ciphertext ($\overline{CT}$) as its output.
- $KeyUpdate(pp, KEK_i, sk, rk_{ki}) \rightarrow (\overline{sk}, \overline{KEK_i})$: The key update algorithm is run by the non-revoked EHR-Us possessing the revoked attribute (i). It takes as input the public keys (pp), the re-key (rk_{ki}), the user secret key (sk) and the key encryption key (KEK_i). It generates an updated user secret key ($\overline{sk}$) and key encryption key ($\overline{KEK_i}$) as its output.

C. SECURITY MODEL

In this subsection, a chosen plaintext attack (CPA) security of our scheme is defined using a security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Our scheme is secure if no probabilistic polynomial time adversary wins the CPA security game with a non-negligible advantage. The game proceeds as follows.

Init: The adversary $\mathcal{A}$ chooses a challenge access structure $OBDD^*$, attribute i^* and version ver^*, and submits to the challenger $\mathcal{C}$.

Setup: Using the security parameter (λ) as an input, the challenger executes the Setup algorithm and forwards the public keys pp to the adversary $\mathcal{A}$. The challenger also sets an initial system wide version $ver \neq ver^*$.

Phase I

Key Queries: The adversary $\mathcal{A}$ issues a series of secret and key encryption key queries by submitting user attribute sets to the challenger, on condition that the submitted attribute sets satisfy the challenge access structure, $OBDD^*$, but the attribute i^* is revoked. The adversary $\mathcal{A}$ can also request for re-key(s) rk_{ki}(s) by submitting revoked attributes of a user to the challenger. The challenger responds by running the Re-KeyGen algorithm and sends $< ver + 1, rk_{ki} >$ to the adversary $\mathcal{A}$. The increase in the version should not result in ver^*, i.e., $ver + 1 \neq ver^*$.

Challenge: Adversary $\mathcal{A}$ then submits two messages M_0 and M_1 of equal lengths to the challenger for encryption and sets the access structure as $OBDD^*$. The challenger $\mathcal{C}$ tosses a coin b and encrypts the message M_b by executing the Encrypt algorithm to generate the ciphertext CT. The challenger goes ahead to re-encrypt CT as CT' using the Re-Encrypt algorithm. The challenger then sets i^* as a revoked attribute and updates the CT' to CT^* by running the CtxtUpdate algorithm. The challenger sets the version as ver^* and sends $< ver^*, CT^* >$ to the adversary $\mathcal{A}$.

Phase II: Same as phase I.

Guess: Adversary $\mathcal{A}$ then puts a guess b' for b.

The advantage of adversary $\mathcal{A}$ in winning this security game is: $|Pr(b' = b) - \frac{1}{2}|$.

V. OUR ACCESS CONTROL CONSTRUCTION

In this section, we present a concrete construction of our access control scheme for secure sharing of PHR in cloud-based health systems. Our constructed scheme supports attribute revocation and is based on the expressive OBDD access structure.

We consider an attribute universe $\mathbb{A} = \{1, \cdots, N\}$, N being the number of attributes in the set $\mathbb{A}$. The construction of our access control scheme proceeds through the following phases.

1) System Setup

This phase uses the setup algorithm to output the system parameters.

Setup$(\lambda) \rightarrow (pp, mk)$: The algorithm chooses bilinear groups $\mathbb{G}, \mathbb{G}_{\mathbb{T}}$ of prime order p, a generator g of $\mathbb{G}$ and a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathbb{T}}$. It also randomly chooses $y \in \mathbb{Z}_p$. For each attribute in the attribute universe the algorithm chooses, $t_1, \cdots, t_N \in \mathbb{Z}_p$ and $t'_1, \cdots, t'_N \in \mathbb{Z}_p$. t_i is associated with the positive attribute value and t'_i is associated with the negative attribute value. For simplicity, we use $\underline{t}_i$ to represent both cases in this work. The algorithm then outputs the public keys (pp) as: $pp = (e, g, Y = e(g, g)^y, \underline{T}_i = g^{\underline{t}_i}|_{\forall i \in \mathbb{A}})$ and the master key (mk) as $(y, \underline{t}_i|_{\forall i \in \mathbb{A}})$. It sets a system wide version as (ver).

2) Key Generation

This phase is run by the TA and it has two algorithms: UserKeyGen and KEKeyGen algorithms.

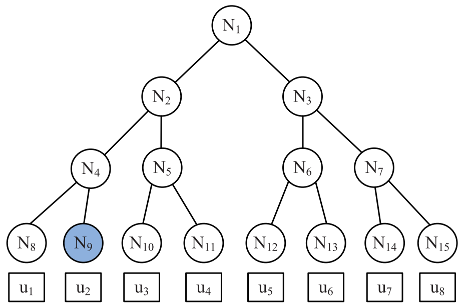


FIGURE 4. A binary tree to manage attribute group members. The blue node indicates a revoked user.

UserKeyGen$(pp, mk, S) \rightarrow sk$: Suppose a data user with an attribute set $(S \subseteq \mathbb{A})$ requests a secret key (sk) from the TA. For each user, the algorithm randomly chooses $r \in \mathbb{Z}_p$ and computes $\hat{\mathcal{D}} = g^{(y-r)}$. For each attribute $i \in S$ and $i \notin S \wedge i \in \mathbb{A}$, the algorithm randomly chooses $r_i \in \mathbb{Z}_p$ such that $r = \sum_1^N r_i$, and computes $\mathcal{D}_i = g^{(\frac{r_i}{\underline{t}_i})}|_{i \in \mathbb{A}}$. The algorithm then outputs a secret key (sk) as,

$$sk = (ver, \hat{\mathcal{D}}, \mathcal{D}_i|_{\forall i \in \mathbb{A}}).$$

KEKeyGen$(pp, i, K_{AGK_i}) \rightarrow KEK_i$: The TA first creates an attribute group (G_i) associated with the attribute (i) it manages. The members of G_i are users who possess the attribute i. The TA establishes a binary tree to manage the members of G_i as shown in Figure 4. The leaf nodes represent the users and each user can be a member of multiple attribute groups. Each attribute group has a minimum cover node set $(minCov)$ that covers all its members. For example, consider an attribute group G_i for attribute i which has members as $\{u_1, u_2, u_5, \cdots, u_8\}$ shown in Figure 4, the minimum cover node set covering all the members is $\{N4, N3\}$. In this case, the minimum cover node set is only used internally by the TA to manage the attribute groups and its members.

For each attribute group (G_i), the KEKeyGen algorithm randomly chooses a unique attribute group key $K_{AGK_i} \in \mathbb{Z}_p$ and computes the key encryption key for the data user as,

$$KEK_i = g^{r_i . K_{AGK_i}}.$$

Then, the TA sends to the data user, the computed sk and KEK_i(s) as the user key (USK).

3) Outsourcing

In this phase, the EHR-O begins by defining an access policy as described in section III and then encrypts his/her data under this policy using the Encrypt algorithm as follows.

Encrypt$(pp, M, OBDD) \rightarrow CT$: This algorithm is run by the EHR-O. It takes as input the public keys (pp), the data M and the $OBDD$ access structure, and outputs a ciphertext (CT). The Encrypt algorithm first randomly chooses $s \in \mathbb{Z}_p$, and computes $\tilde{\mathcal{C}} = M.Y^s$ and $\hat{\mathcal{C}} = g^s$. Additionally, it computes $\mathcal{C}_i = (g^{\underline{t}_i . s}|_{\forall i \in I})$. Where I is the attribute set in the access structure. The algorithm outputs CT as:

$$CT = (ver, OBDD, \tilde{\mathcal{C}}, \hat{\mathcal{C}}, \mathcal{C}_i)$$

which is then outsourced to the cloud.

4) **Re-encryption**

This phase associates the attribute elements in the ciphertext with their respective attribute groups through the attribute group information (GI_i). It is performed by the cloud which helps to lessen the computational burden on the data owner. Upon receiving CT from the EHR-O, the cloud immediately requests for the GI_i for each attribute embedded in CT from the TA. The TA computes $GI_i = g^{\underline{t}_i . K_{AGK_i}}$ for each attribute and sends it to the cloud. After receiving the GI_i from the TA, the cloud re-encrypts CT using the Re-Encrypt algorithm.

Re-Encrypt$(pp, CT, GI_i) \rightarrow CT'$: The algorithm re-encrypts the $\mathcal{C}_i$ element of CT to output the re-encrypted ciphertext CT' as:

$$CT' = (ver, OBDD, \tilde{\mathcal{C}}, \hat{\mathcal{C}}, \mathcal{C}'_i = g^{\underline{t}_i.(s+K_{AGK_i})}|_{\forall i \in \mathbb{A} \wedge i \in I})$$

CT is then replaced by CT'.

5) **Decryption**

In case a data user's secret key version and ciphertext version match, and his/her set of attributes satisfy the OBDD access structure included in the CT', the Decrypt algorithm is used to recover the data.

Decrypt$(CT', USK, pp) \rightarrow M/\perp$: In this case, the Decrypt algorithm is run by the EHR-U and it takes as inputs the re-encrypted ciphertext (CT'), the user key (USK) and the public key (pp) to recover the encrypted data M or output $\perp$. The OBDD access structure satisfaction is done recursively as:

(a) Find the root node and make it as the current node.

(b) Unpack the information in the current node $Node^i_{id}$. If, $i \in S$ and i is present, proceed to (c), otherwise, proceed to (d).

(c) Search the $high(v)$ of the current node.
   1) If the $high(v)$ is terminal node 0, stop.
   2) If the $high(v)$ is terminal node 1, proceed to (e).
   3) If the $high(v)$ is a non-terminal node, make it as the current node and return to (b).

(d) Search the $low(v)$ of the current node.
   1) If the $low(v)$ is terminal node 0, stop.
   2) If the $low(v)$ is terminal node 1, proceed to (e).
   3) If the $low(v)$ is a non-terminal node, make it as the current node and return to (b).

(e) Store the path P_j that satisfies the OBDD (i.e., *root node* $\rightarrow 1$).

The Decrypt algorithm computes:

$$\begin{aligned} &e(\mathcal{D}_i, \mathcal{C}'_i)/e(KEK_i, g) \\ &\quad = e(g^{r_i/\underline{t}_i}, g^{s.\underline{t}_i}.g^{\underline{t}_i.K_{AGK_i}})/e(g^{r_i.K_{AGK_i}}, g) \\ &\quad = e(g, g)^{r_i.s}.e(g, g)^{r_i.K_{AGK_i}}/e(g, g)^{r_i.K_{AGK_i}} \\ &\quad = e(g, g)^{r_i.s} \end{aligned}$$

Finally, M is recovered as: $M = \frac{\tilde{\mathcal{C}}}{e(\hat{\mathcal{C}}, \hat{\mathcal{D}}).\Pi^n_{i=1} e(g,g)^{r_i.s}}$.

6) **Attribute Revocation**

In the circumstances a data user is revoked of an attribute i, the TA generates an updated membership information for the attribute group G_i. A new minimum cover set is chosen for the non-revoked users. For example, suppose a user u_2 (the blue node in Figure 4) is revoked of the attribute i, the non-revoked users are $\{u_1, u_5, u_6, u_7, u_8\}$ and the new minimum cover set for the non-revoked users is $\{N_8, N_3\}$. The TA then randomly chooses a new attribute group key $K'_{AGK_i} \in \mathbb{Z}_p$ for G_i. Three algorithms: Re-KeyGen, KeyUpdate and CtxtUpdate are then used to complete the attribute revocation task as follows.

Re-KeyGen$(pp, i, K_{AGK_i}, K'_{AGK_i}) \rightarrow (rk_{ci}, rk_{ki})$: The algorithm takes as input the public keys (pp), the old attribute group key K_{AGK_i}, the new attribute group key K'_{AGK_i} and the revoked attribute i, and it produces as output two categories of re-keys: rk_{ci} for ciphertext update and rk_{ki} for user key update. rk_{ci} is generated as:

$$\forall i = revoked : rk_{ci} = g^{\underline{t}_i.(K'_{AGK_i} - K_{AGK_i})}.$$

The version is then increased by one. The TA sends the outcome $(ver + 1, rk_{ci})$ to the cloud. rk_{ki} is generated for each non-revoked data user bearing the revoked attribute (i) as:

$$\forall i = revoked : rk_{ki} = g^{r_i(K'_{AGK_i} - K_{AGK_i})}.$$

Similarly, the version is increased by one. TA then sends the outcome $(ver + 1, rk_{ki})$ to each non-revoked data user.

CtxtUpdate$(pp, CT', rk_{ci}) \rightarrow \overline{CT}$: This algorithm is run by the cloud to update the CT' after an attribute revocation. It takes as input the CT' and the re-key(s) rk_{ci}(s), and outputs an updated ciphertext $\overline{CT}$. The algorithm chooses a random $s' \in \mathbb{Z}_p$ and does the update as:

$$\begin{aligned} &\overline{CT} = \Big(ver + 1, OBDD, \overline{\tilde{\mathcal{C}}} = M.Y^{(s+s')}, \overline{\hat{\mathcal{C}}} = g^{(s+s')}, \\ &\quad \forall i = revoked : \overline{\mathcal{C}_i} = g^{(s+s').\underline{t}_i}.g^{K_{AGK_i}}.rk_{ci} = g^{(s+s').\underline{t}_i}.g^{K'_{AGK_i}}, \\ &\quad \forall i \neq revoked : \overline{\mathcal{C}_i} = g^{(s+s').\underline{t}_i}.g^{K_{AGK_i}}\Big) \end{aligned}$$

Note that the ciphertext version is increased by 1.

KeyUpdate$(pp, KEK_i, sk, rk_{ki}) \rightarrow (\overline{sk}, \overline{KEK_i})$: This algorithm is run by the non-revoked data users who possess the revoked attribute i. The algorithm takes as input the public keys (pp), the user secret key (sk), the key encryption key(s) (KEK_i(s)) and the re-key(s) (rk_{ki}(s)) and outputs updated user secret key $\overline{sk}$ and key encryption key $\overline{KEK_i}$. The keys are updates as:

$$\begin{aligned} &\overline{sk} = \Big(ver + 1, \overline{\hat{\mathcal{D}}} = g^{y-r}, \forall i : \overline{\mathcal{D}_i} = \mathcal{D}_i\Big) \\ &\forall i = revoked : \overline{KEK_i} = KEK_i.rk_{ki} = g^{r_i.K'_{AGK_i}}, \\ &\forall i \neq revoked : \overline{KEK_i} = g^{r_i.K_{AGK_i}}, \end{aligned}$$

Take note of the increase in the version number in the updated user secret key ($\overline{sk}$).

CORRECTNESS AFTER REVOCATION:

After an attribute revocation, the decryption proceeds in the same way by first computing $e(\overline{\mathcal{D}_i}, \overline{\mathcal{C}_i})/e(\overline{KEK}_i, g) =$

$e(g,g)^{r_i(s+s')}$ for each attribute as:

$$\forall i = revoked : e(g^{r_i/\underline{t}_i}, g^{(s+s').\underline{t}_i}.g^{\underline{t}_i.K'_{AGK_i}})/e(g^{r_i.K'_{AGK_i}}, g)$$
$$\forall i \neq revoked : e(g^{r_i/\underline{t}_i}, g^{(s+s').\underline{t}_i}.g^{\underline{t}_i.K_{AGK_i}})/e(g^{r_i.K_{AGK_i}}, g)$$

Then, M is recovered as $M = \frac{\bar{\tilde{\mathcal{C}}}}{e(\bar{\hat{\mathcal{C}}},\bar{\hat{\mathcal{D}}}).\Pi_{i=1}^{n} e(g,g)^{r_i.(s+s')}}$.

VI. SECURITY ANALYSIS

In this section, we present the security analysis of our our access control scheme in terms of a security proof, and forward and backward security.

A. FORWARD SECURITY

After an attribute revocation, the attribute group key associated with the revoked attribute and the system version get updated. These become the group key and system version assigned to the new data users joining the system. Note that whenever an attribute is revoked, the re-keying and the ciphertext update algorithms in our scheme are used to keep the ciphertext updated as in [30]. This way, the forward security is guaranteed.

B. BACKWARD SECURITY

Similarly, after an attribute revocation, the TA generates a new attribute group key. The TA then generates a re-key for each non-revoked data user which it forwards to the respective data users. This prevents the revoked data user from updating his/her user key. However, in the circumstances a revoked data user obtains a re-key intended for another non-revoked data user, he/she is still prevented from successfully updating his/her keys because of the r_i component included in the re-key. Note that the r_i of an attribute is different for different users. This achieves backward security.

C. SECURITY PROOF

A security proof of our access control under the selective model is presented in this subsection.

Theorem 1: If there is a probabilistic polynomial-time adversary $\mathcal{A}$ that can win our CPA game with a non-negligible advantage ε, then we can construct a simulator $\mathcal{B}$ that can distinguish a DBDH tuple from a random tuple with a non-negligible advantage.

Proof: The challenger generates a tuple $\left(A = g^a, B = g^b, C = g^c, Z = e(g,g)^z\right)$. Where $a, b, c, z \in_R \mathbb{Z}_p$. The challengers flips a coin b and if $b = 0$, the challenger sets $Z = e(g,g)^{abc}$. Otherwise, $Z \in_R \mathbb{G}_{\mathbb{T}}$. $\mathcal{B}$ is then tasked to guess b through playing the CPA game. During the game, $\mathcal{B}$ acts as the challenger and it proceeds as follows:

Init: The adversary $\mathcal{A}$ chooses a challenger access structure $OBDD^*$, an attribute i^* and a version ver^* and submits to $\mathcal{B}$.

Setup: $\mathcal{B}$ simulates the public parameters by running the Setup algorithm. $\mathcal{B}$ defines Y as $e(A,B) = e(g,g)^{ab}$. $\mathcal{B}$ then generates the attribute universe $\mathbb{A}$ and for each attribute in the universe, $\mathcal{B}$ randomly chooses $\underline{t}_i$. $\mathcal{B}$ then computes $\underline{T}_i = g^{\underline{t}_i}$. $\mathcal{B}$ publishes the public parameters as $< e, \mathbb{G}, g, Y, \underline{T}_i|_{\forall i \in \mathbb{A}} >$. $\mathcal{B}$ then sets a system version $ver \neq ver^*$.

Phase I: Adversary $\mathcal{A}$ issues a series of queries for secret keys, key encryption keys and re-keys by submitting attribute sets to $\mathcal{B}$. $\mathcal{A}$ submits the attribute set S to $\mathcal{B}$. S satisfies the challenge access structure $OBDD^*$ but the attribute i^* is revoked and version is not ver^*. $\mathcal{B}$ generates a user secret key sk as follows: For each attribute set S, it chooses $r \in_R \mathbb{Z}_p$ and for each $i \in S \wedge i \neq i^*$, it chooses $r_i \in \mathbb{Z}_p$. It computes $\hat{\mathcal{D}} = g^{ab-y}$ and $\mathcal{D}_i = g^{r_i/\underline{t}_i}$. For i^* it randomly chooses $r^* \in_R \mathbb{Z}_p$ and computes $\mathcal{D}_{i^*} = g^{r^*/\underline{t}^*}$. $\underline{t}^*$ is the $\underline{t}_i$ associated with i^*. $\mathcal{B}$ sets sk as:

$$sk = (ver, \hat{\mathcal{D}}, \mathcal{D}_{i^*}|_{\forall i^*}, \mathcal{D}_i|_{\forall i \in S \wedge i \neq i^*, ver \neq ver^*}).$$

$\mathcal{B}$ also generates attribute groups G_i. For each attribute group associated with i, $\mathcal{B}$ chooses a random $k_i \in_R \mathbb{Z}_p$ as a group key. For i^*, $\mathcal{B}$ randomly chooses $k_{i^*} \in_R \mathbb{Z}_p$ and sets the group key associated with i^* as $b.k_{i^*}$. $\mathcal{B}$ computes the key encryption key(s) KEK_i(s) as follows:

$$\forall i \neq i^* : KEK_i = g^{r_i.k_i}, i = i^* : KEK_{i^*} = g^{r^*.b.k_{i^*}}$$

$\mathcal{B}$ then sends the user secret key and the key encryption keys to $\mathcal{A}$. $\mathcal{A}$ can also request for re-key(s) rk_{ki}(s) by submitting revoked attribute $i \neq i^*$. $\mathcal{B}$ generates a new group key for the revoked attribute i as $k'_i \in_R \mathbb{Z}_p$ and using the Re-KeyGen algorithm, $\mathcal{B}$ computes rk_{ki} as $g^{r_i(k'_i - k_i)}$. $\mathcal{B}$ increases the version and sends to $\mathcal{A}$ the result of the Re-KeyGen algorithm as:

$$\left(ver + 1, rk_{ki}|_{\forall i \neq i^*, ver+1 \neq ver^*}\right)$$

Challenge: $\mathcal{A}$ then submits two messages M_0 and M_1 of equal lengths to $\mathcal{B}$, and sets the access structure as $OBDD^*$. $\mathcal{B}$ initially sets $ver^* - 1$ as the current version. $\mathcal{B}$ then flips a coin $b \in \{0, 1\}$ and encrypts M_b by executing the Encrypt algorithm to generate the ciphertext CT as follows:

$$\tilde{C} = M_b.Y^c = M_b.e(g,g)^{abc} = M_b.Z,$$
$$\hat{C} = g^c,$$
$$\forall i \in I^* \wedge i \neq i^* : C_i = g^{c\underline{t}_i}, i = i^* : C_{i^*} = g^{c.\underline{t}^*}$$

where I^* is the attribute set in $OBDD^*$ including the i^* attribute.

$$CT = (ver^* - 1, OBDD^*, \tilde{C}, \hat{C}, C_i, C_{i^*}).$$

$\mathcal{B}$ then re-encrypts CT to generate CT' using the Re-Encrypt algorithm as:

$$CT' = \Big(ver^* - 1, OBDD^*, \tilde{C}' = \tilde{C}, \hat{C}' = \hat{C},$$
$$i = i^* : C'_{i^*} = (C_{i^*}).g^{\underline{t}^*.b.k_{i^*}}$$
$$i \neq i^* : C'_i = (C_i).g^{\underline{t}_i.k_i}\Big)$$

$\mathcal{B}$ then sets i^* as a revoked attribute and updates CT' by running the CtxtUpdate algorithm to generate $\overline{CT}$ as follows:

First $\mathcal{B}$ randomly choose $s' \in \mathbb{Z}_p$. For i^*, $\mathcal{B}$ generates rk_{ci^*} as $g^{\underline{t}^*.(k'_{i^*}-b.k_{i^*})}$. Where $k'_{i^*} \in_R \mathbb{Z}_p$. $\mathcal{B}$ updates CT' as:

$$\overline{\tilde{C}} = M_b.Y^{c+s'} = M_b.Z.e(g,g)^{abs'},$$

$$\overline{\hat{C}} = g^{(c+s')},$$

$$\forall i = i^* : \overline{C_{i^*}} = g^{(c+s')\underline{t}^*}.g^{\underline{t}^*.b.k_{i^*}}.rk_{ci^*} = g^{(c+s')\underline{t}^*}.g^{\underline{t}^*.k'_{i^*}},$$

$$\forall i \neq i^* : \overline{C_i} = g^{(c+s')\underline{t}_i}.g^{\underline{t}_i.k_i}$$

$\mathcal{B}$ then increases the version $ver^* - 1$ by 1 to obtain the challenge version ver^*.

$$\overline{CT} = (ver^*, OBDD^*, \overline{\tilde{C}}, \overline{\hat{C}}, \overline{C_{i^*}}, \overline{C_i})$$

$\mathcal{B}$ sets $\overline{CT}$ as the challenge ciphertext CT^* and sends it to $\mathcal{A}$.

Phase II: Is the same as phase I.

Guess: $\mathcal{A}$ then outputs a guess b' for b. If $b' = b$, $Z = e(g,g)^{abc}$. Otherwise Z is a random value in $\mathbb{G}_{\mathbb{T}}$. Hence, the advantage of $\mathcal{B}$ in winning this game is,

$$\frac{1}{2}Pr[b' = b|b = 0] + \frac{1}{2}Pr[b' = b|b = 1] - \frac{1}{2} = \frac{\varepsilon}{2}.$$

VII. PERFORMANCE EVALUATION

In this section, we compare our scheme with some similar access control schemes [7], [8], [10], [13], [28], [30].

Table 2 shows feature comparison between our scheme and the similar schemes. We carried out the comparison in terms of expressivess, revocation ability, unboundedness, forward security and backward security. It can be seen that the LGCXLQ [8] scheme and our scheme use the OBDD access structure to achieve their expressivenesses. The WGZ [28], ZZD [7] and YJ [30] schemes use the LSSS access structure to achieve their expressivenesses. Meanwhile, the Hur-Nor [10] and YWRL [13] schemes use the threshold gates and AND-gate as their access structures, respectively. The WGZ [28], Hur-Nor [10], YWRL [13], YJ [30] schemes and our scheme all possess immediate attribute revocation ability while the LGCXLQ [8] and ZZD [7] schemes do not revoke. The LGCXLQ [8], YWRL [13] schemes and our scheme are bounded, i.e., the size of their attribute universe is polynomially bounded to the security parameter. Meanwhile, the WGZ [28], Hur-Nor [10] and YJ [30] schemes are unbounded. Apart from the setup algorithm, the boundedness has no additional computational effect on the simulation results shown later in this section. Among the schemes that possess revocation abilities, the WGZ [28] and YJ [30] schemes, and our scheme achieve both forward and backward security. The Hur-Nor [10] and YWRL [13] schemes achieve forward security but do not guarantee backward security.

We further performed theoretical computational cost analysis of our scheme in comparison with the related schemes as shown in Table 3. In the analysis, we let: I_{aa} be the number of attribute authorities whose attributes are held by a data user, I_{na} be the number of attribute authorities in the system whose attributes are not held by the data user, $|I_{aa}|$ be the number of attribute authorities whose attributes are included

TABLE 2. Feature comparison of our scheme with other related schemes.

Scheme	Express	Revoc	Unbounded	Forward	Backward
WGZ [28]	LSSS	✓	✓	✓	✓
Hur-Nor [10]	Threshold	✓	✓	✓	×
LGCXLQ [8]	OBDD	×	×	N/A	N/A
ZZD [7]	LSSS	×	✓	N/A	N/A
YWRL [13]	AND	✓	×	✓	×
YJ [30]	LSSS	✓	✓	✓	✓
Our Scheme	OBDD	✓	×	✓	✓

Express is expressivness, *Revoc* is revocation, *Forward* is forward security and *Backward* is backward security.

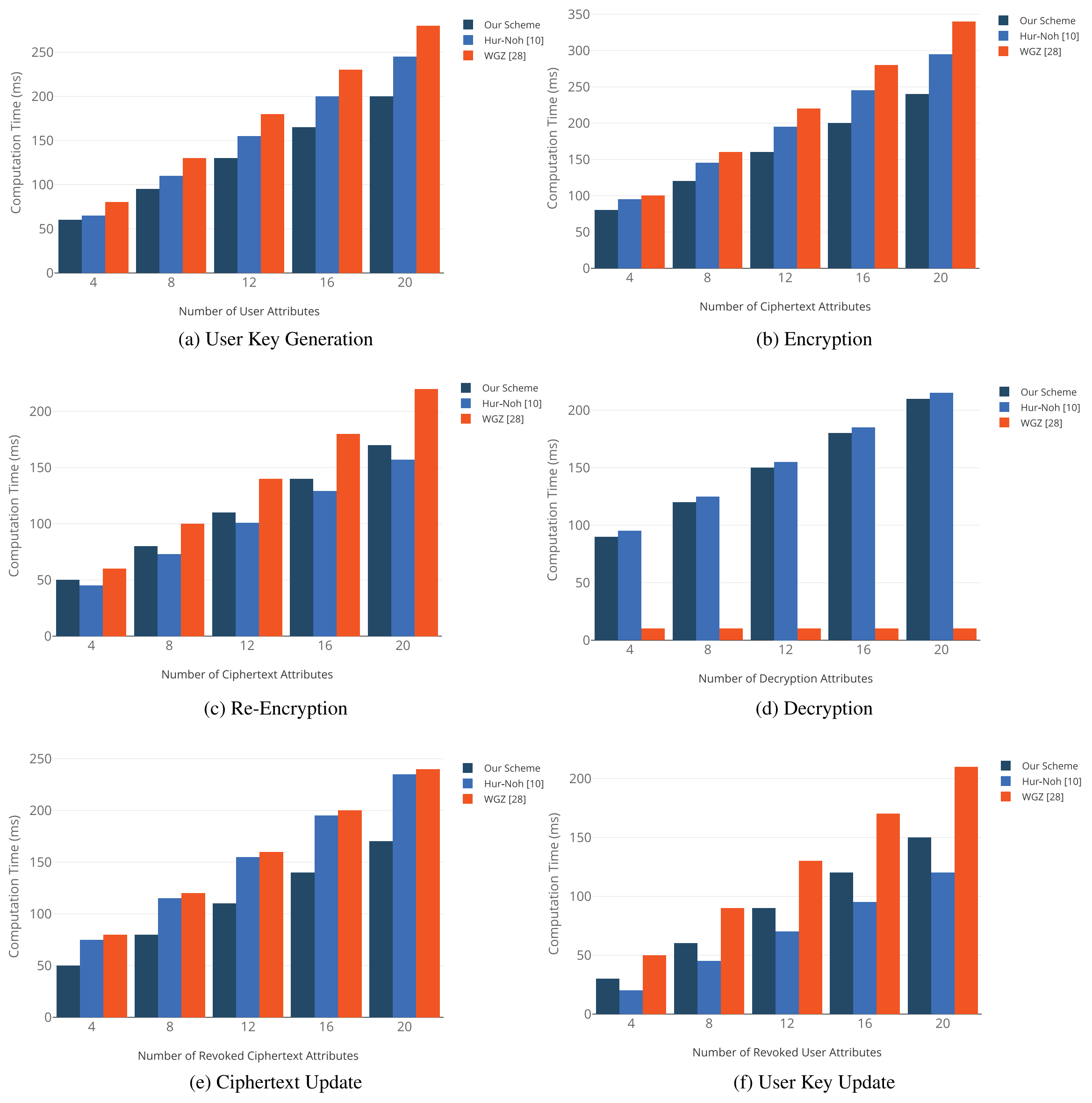
in the ciphertext, $|k|$ be the number of the data user attributes, $|l|$ be the number of ciphertext attributes, $|R|$ be the number of the satisfied paths in the OBDD access structure, $|d|$ be the number of the attributes involved in decryption, $|r_a|$ be the number of the revoked attributes, $|r_n|$ be the number of unrevoked attributes and u_r be the number of data users who possess a revoked attribute. Table 3 shows the computational cost analysis results.

From Table 3, during key generation, the YJ [30] scheme has the highest computational cost and the LGCXLQ [8] scheme has the least computational cost. The YJ [30] scheme is a multi-authority scheme and each user key contains components from all the attribute authorities and hence the high computational cost. The LGCXLQ [8] scheme combines the attribute elements in the user secret key as one and thus, the low computational cost. Our scheme has optimal computation cost in key generation as performs only $2.|k| + 1$ exponentiation operations during user key generation. During encryption, the ZZD [7] scheme incurs the highest computational cost while the YWRL [13] scheme and our scheme incur the least computational cost which includes one multiplication and $|l| + 2$ exponentiation operations only. This is mainly because of having no hashing operations and only one attribute element in the ciphertext. Re-encryption which associates the attribute elements with the attribute groups is only performed by the Hur-Nor [10] and WGZ [28] schemes, and our scheme. However, the computational cost incurred during re-encryption is low in the Hur-Nor [10] scheme and our scheme as compared with that incurred in the WGZ [28] scheme. This is mainly attributed to having only one attribute element that requires modification in the Hur-Nor [10] scheme and our scheme. During decryption, our scheme performs better than the YJ [30], Hur-Nor [10] and ZZD [7] schemes, as it has less exponentiation, multiplication and pairing operations compared with the stated schemes. The WGZ [28] incurs the least computational cost. This is mainly because of the outsourcing of the computationally demanding attribute operations to the cloud. After a revocation, the Hur-Nor [10] and WGZ [28] schemes, and our scheme update the entire ciphertext. Meanwhile, the YJ [30] and YWRL [13] schemes update fewer ciphertext elements. However, amongst the schemes that update the entire ciphertext, our scheme due to its few attribute elements, incurs the least computational cost. During key update, the YJ [30] and WGZ [28] schemes, and our scheme generate a key update

TABLE 3. Computation cost comparison between our scheme and related CP-ABE schemes.

Scheme	Operation	YJ [30]	LGCXLQ [8]	Hur-Nor [10]	WGZ [28]	ZZD [7]	YWRL [13]	Our Scheme
Key Gen	Mult	$I_{aa}.\|k\| + 2.I_{aa} + 2.I_{na}$	N/A	$\|k\|$	1	$2.\|k\| + 3$	N/A	N/A
	Expo	$2.I_{aa}.\|k\| + 4.I_{aa} + 4.I_{na}$	2	$3.\|k\| + 1$	$4.\|k\| + 4$	$2.\|k\| + 3$	$\|k\| + 1$	$2.\|k\| + 1$
Encrypt	Mult	$\|l\| + \|I_{aa}\|$	$\|R\|.(\|l\| - 1) + 1$	1	1	$7.\|l\| + 2$	1	1
	Expo	$5.\|l\| + 3$	$\|R\| + 2$	$2.\|l\| + 2$	$3.\|l\| + 3$	$7.\|l\| + 4$	$\|l\| + 2$	$\|l\| + 2$
Re-Encrypt	Mult	N/A	N/A	N/A	$\|l\|$	N/A	N/A	$\|l\|$
	Expo	N/A	N/A	$\|l\|$	$2.\|l\|$	N/A	N/A	$\|l\|$
Decrypt	Mult	$3.I_a + 3.\|d\| + I_a.\|d\| - 2$	2	$\geq \|d\| + 2$	1	$4.\|d\| - 1$	$\|d\| + 1$	$2.\|d\| + 1$
	Expo	$2.I_a + 2.\|d\| + I_a.\|d\|$	N/A	$\|k\|$	1	$3.\|d\| + 1$	N/A	N/A
	Pair	$3.I_a + 4.\|d\| + I_a.\|d\|$	1	$\geq 2.\|d\| + 1$	N/A	$2.\|d\| + 3$	$\|d\| + 2$	$2.\|d\| + 1$
CT Update	Mult	$4.\|r_a\|$	N/A	$2.\|l\| + 2$	$2.\|r_a\| + \|r_n\| + 3$	N/A	N/A	$2.\|r_a\| + \|r_n\| + 2$
	Expo	$3.\|r_a\|$	N/A	$3.\|l\| + 2$	$3.\|r_a\| + \|r_n\| + 3$	N/A	$\|l\|$	$2.\|r_a\| + \|r_n\| + 2$
Key Update	Mult	$\|r_a\|$	N/A	N/A	$\|r_a\|$	N/A	$\|k\|$	$\|r_a\|$
	Expo	$\|r_a\|.u_r$	N/A	$\|r_a\|$	$3.\|r_a\|.u_r$	N/A	$\|k\|$	$\|r_a\|.u_r$

Mult is multiplication operation, *Expo* is exponentiation operation and *pair* is pairing operation.



(a) User Key Generation (b) Encryption (c) Re-Encryption (d) Decryption (e) Ciphertext Update (f) User Key Update

FIGURE 5. Experimental results of our scheme in comparison with the Hur-Nor [10] and WGZ [28] schemes.

for each non-revoked user possessing the revoked attribute. However, the YJ [30] scheme and our scheme incur the least computation cost as compared with the WGZ [28] scheme. They only perform $|r_a|.u_r$ exponentiation operations as compared to $3.|r_a|.u_r$ exponentiation operations performed by the WGZ [28] scheme.

To explicitly demonstrate the computational efficiency of our scheme, we further performed an experiment in comparison with the Hur-Nor [10] and WGZ [28] schemes. The experimental setup included: a desktop computer with 4GB of RAM and Intel core processor with a speed of 2.0GHz, running Ubuntu 12.04 operating system. We installed PBC and OpenSSL to provide the necessary cryptographic functionalities. The implementation was done using the Charm tool [29] and python 2.7. We present the results in Figure 5 and each result is an average of 20 independent trials.

Figure 5(a) shows the variation of computation time for the user key generation against the number of user attributes. It can be observed that the computation time increases with an increase in the number of user attributes. However, the increase is more for the Hur-Nor [10] and WGZ [28] schemes as compared to ours. This is because our scheme has fewer user key elements that require exponentiation operations as compared to the Hur-Nor [10] and WGZ [28] schemes and there is no need for hashing. In Figure 5(b), we present the variation of the computation time against the number of ciphertext attributes during encryption. The computation time increases with an increase in the number of ciphertext attributes. Similarly, our scheme performs better than the Hur-Nor [10] and WGZ [28] schemes due to the fewer exponentiation operations it performs. Figure 5(c) shows the variation of the computation time against the number of ciphertext attributes during re-encryption. The computation time also increases with an increase in the number of ciphertext attributes. The increase is more for the WGZ [28] scheme as compared to ours. However, in this case, the Hur-Nor [10] scheme performs better than our scheme. The variation of computation time for decryption against the number of attributes involved in decryption is presented in Figure 5(d). It can be seen that the computation time increases with an increase in the number of attributes involved in decryption for the Hur-Nor [10] scheme and our scheme, while it remains constant for the WGZ [28] scheme. This is because the WGZ [28] scheme outsources the computationally demanding attribute operations to the cloud. In Figure 5(e), we present the variation of computation time for ciphertext update against the number of revoked attributes. Similarly, the computation time increases with an increase in the number of revoked attributes. The increase is more for the Hur-Nor [10] and WGZ [28] schemes as compared to ours. Figure 5(f) shows the variation of computation time for user key update against the number of revoked user attributes. It can also be observed that the computation time increases with the number of revoked user attributes. In this case, our scheme performs better than the WGZ [28] scheme but worse as compared with the Hur-Nor [10] scheme. In summary, the experimental results confirm the theoretical efficiency analysis and our scheme is optimally efficient in key generation, encryption, re-encryption, decryption and attribute revocation as compared to the Hur-Nor [10] and WGZ [28] schemes.

VIII. CONCLUSION

In this study, we focused on addressing the privacy issues of PHR in cloud based health systems. We proposed and constructed an expressive, efficient and revocable access control for fine-grained access to health data based on OBDD access structure. In our construction, we leveraged attribute groups and assigned version numbers to ciphertexts and user keys to achieve attribute/user revocation while preventing collusion between revoked and non-revoked users. Security and performance analysis show that our proposed access control scheme is expressive, efficient and secure. Additionally, the proposed access control scheme guarantees forward and backward security.

REFERENCES

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

[2] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, Berkeley, CA, USA, May 2007, pp. 321–334.

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

[4] 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.

[5] Y. S. Rao, "A secure and efficient ciphertext-policy attribute-based signcryption for personal health records sharing in cloud computing," *Future Gener. Comput. Syst.*, vol. 67, pp. 133–151, Feb. 2017.

[6] A. Michalas and N. Weingarten, "HealthShare: Using attribute-based encryption for secure data sharing between multiple clouds," in *Proc. IEEE 30th Int. Symp. Comput.-Based Med. Syst. (CBMS)*, Jun. 2017, pp. 811–815.

[7] 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.

[8] L. Li, T. Gu, L. Chang, Z. Xu, Y. Liu, and J. Qian, "A ciphertext-policy attribute-based encryption based on an ordered binary decision diagram," *IEEE Access*, vol. 5, pp. 1137–1145, 2017.

[9] K. D. Mandl, W. W. Simons, W. C. Crawford, and J. M. Abbett, "Indivo: A personally controlled health record for health information exchange and communication," *BMC Med. Inform. Decis. Making*, vol. 7, no. 1, p. 25, 2007.

[10] 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.

[11] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptogr.* Berlin, Germany: Springer, 2011, pp. 53–70.

[12] L. Cheung and C. Newport, "Provably secure ciphertext policy ABE," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, 2007, pp. 456–465.

[13] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Secur. (ASIACCS)*, 2010, pp. 261–270.

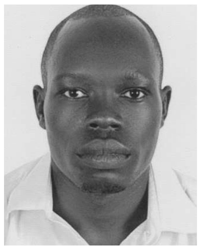
[14] Y. Zhang, X. Chen, J. Li, H. Li, and F. Li, "FDR-ABE: Attribute-based encryption with flexible and direct revocation," in *Proc. 5th Int. Conf. Intell. Netw. Collaborative Syst.*, Sep. 2013, pp. 38–45.

[15] Y. Zhang, X. Chen, J. Li, D. S. Wong, and H. Li, "Anonymous attribute-based encryption supporting efficient decryption test," in *Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur., (ASIACCS)*, New York, NY, USA, 2013, pp. 511–516.

[16] J. Lai, R. H. Deng, Y. Li, "Expressive CP-ABE with partially hidden access structures," in *Proc. 7th ACM Symp. Inf., Comput. Commun. Secur.*, Seoul, South Korea, May 2012, pp. 1–12.

[17] X. Li and M. Zhang, "Cloud storage access control policy based on ma-abe," *J. Lanzhou Univ. Technol.*, vol. 42, no. 10, pp: 133–140, 2015.

[18] Z. Zhou, D. Huang, and Z. Wang, "Efficient privacy-preserving ciphertext-policy attribute based-encryption and broadcast encryption," *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 126–138, Jan. 2015.
[19] L. You and L. Wang, "Hierarchical authority key-policy attribute-based encryption," in *Proc. IEEE 16th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2015, pp. 868–872.
[20] A. Samydurai, K. Revathi, P. Prema, D. S. Arulmozhiarasi, J. Jency, and S. Hemapriya, "Secured Health Care Information exchange on cloud using attribute based encryption," in *Proc. 3rd Int. Conf. Signal Process., Commun. Netw. (ICSCN)*, Mar. 2015, pp. 1–5.
[21] S. Maheswari and U. Gudla, "Secure sharing of personal health records in Jelastic cloud by attribute based encryption," in *Proc. 4th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Jan. 2017, pp. 1–4.
[22] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. 21st Annu. Int. Cryptol. Conf.* London-U.K.: Springer-Verlag, 2001, pp. 213–229.
[23] C. Y. Lee, "Representation of switching circuits by binary-decision programs," *Bell Syst. Tech. J.*, vol. 38, no. 4, pp. 985–999, Jul. 1959.
[24] S. B. Akers, "Binary decision diagrams," *IEEE Trans. Comput.*, vol. C-27, no. 6, pp. 509–516, Jun. 1978.
[25] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Comput.*, vol. C-35, no. 8, pp. 677–691, Aug. 1986.
[26] G. Wei, "Reliability analysis of the Internet of Things based on ordered binary decision diagram," *Int. J. Online Eng.*, vol. 14, no. 8, pp. 20–34, 2018.
[27] N. Jansen. (Mar. 2018). *Binary Decision Diagrams*. Inst. for Computing Inf. Sci., Radboud Univ. Nijmegen. [Online]. Available: https://ocw.cs.ru.nl/
[28] S. Wang, K. Guo, and Y. Zhang, "Traceable ciphertext-policy attribute-based encryption scheme with attribute level user revocation for cloud storage," *PLoS ONE*, vol. 13, no. 9, Sep. 2018, Art. no. e0203225.
[29] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: A framework for rapidly prototyping cryptosystems," *J. Cryptograph. Eng.*, vol. 3, no. 2, pp. 111–128, Jun. 2013.
[30] 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.
[31] R. Ahuja and S. K. Mohanty, "A scalable attribute-based access control scheme with flexible delegation cum sharing of access privileges for cloud storage," *IEEE Trans. Cloud Comput.*, to be published.
[32] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[33] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, "User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1767–1777, Jun. 2018.
[34] J. Wei, X. Chen, X. Huang, X. Hu, and W. Susilo, "RS-HABE: Revocable-storage and hierarchical attribute-based access scheme for secure sharing of e-health records in public cloud," *IEEE Trans. Dependable Secure Comput.*, to be published.
[35] Y. S. Rao and R. Dutta, "Efficient attribute-based signature and signcryption realizing expressive access structures," *Int. J. Inf. Secur.*, vol. 15, no. 1, pp. 81–109, Feb. 2016.
[36] P. Liang, L. Zhang, L. Kang, and J. Ren, "Privacy-preserving decentralized ABE for secure sharing of personal health records in cloud storage," *J. Inf. Secur. Appl.*, vol. 47, pp. 258–266, Aug. 2019.



KENNEDY EDEMACU received the B.S. degree in computer science from Gulu University, in 2011, and the M.S. degree in data communication and software engineering from Makerere University, in 2014. He is currently pursuing the Ph.D. degree in computer science with Sangmyung University. His current research interests include privacy in cloud computing, cryptography, and artificial intelligence.

BEAKCHEOL JANG received the B.S. degree from Yonsei University, in 2001, the M.S. degree from the Korea Advanced Institute of Science and Technology, in 2002, and the Ph.D. degree from North Carolina State University, in 2009, all in computer science. He is currently an Associate Professor with the Department of Computer Science, Sangmyung University. His primary research interests include wireless networking, big data, the Internet of Things, and artificial intelligence.

JONG WOOK KIM (Member, IEEE) received the Ph.D. degree from the Computer Science Department, Arizona State University, in 2009. He was a Software Engineer with the Query Optimization Group, Teradata, from 2010 to 2013. He is currently an Associate Professor of computer science with Sangmyung University. His primary research interests include data privacy, distributed databases, and query optimization.