Received January 10, 2020, accepted January 23, 2020, date of publication January 28, 2020, date of current version February 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2969957

Lightweight Revocable Hierarchical Attribute-Based Encryption for Internet of Things

MOHAMMAD ALI[1], MOHAMMAD-REZA SADEGHI[1], AND XIMENG LIU[2,3], (Member, IEEE)
[1]Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran 15875-4413, Iran
[2]Guangdong Provincial Key Laboratory of Data Security and Privacy Protection, Guangzhou 510632, China
[3]Shaanxi Key Laboratory of Network and System Security, Xidian University, Xi'an 710071, China

Corresponding author: Ximeng Liu (snbnix@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant U1804263 and Grant 61702105, and in part by the Opening Project of Guangdong Provincial Key Laboratory of Data Security and Privacy Protection under Grant 2017B030301004-12.

ABSTRACT The Internet of Things (IoT) is an emerging technology that can benefit from cloud infrastructure. In a cloud-based IoT network, a variety of data is collected by smart devices and transmitted to a cloud server. However, since the data may contain sensitive information about individuals, providing confidentiality and access control is essential to protect the users' privacy. Attribute-based encryption (ABE) is a promising tool to provide these requirements. However, most of ABE schemes neither provide efficient encryption and decryption mechanisms nor offer flexible and efficient key delegation and user revocation approaches. In this paper, to address these issues, we propose a lightweight revocable hierarchical ABE (**LW-RHABE**) scheme. In our scheme, computation overhead on the user side is very efficient, and most of the computational operations are performed by the cloud server. Also, using the hierarchical model, our scheme offers flexible and scalable key delegation and user revocation mechanisms. Indeed, in our scheme, key delegation and user revocation associated with each attribute can be handled by several key authorities. We provide the security definition for **LW-RHABE**, and we prove its security in the standard model and under the hardness assumption of the decisional bilinear Diffie-Hellman (DBDH) problem.

INDEX TERMS Internet of Things, cloud computing, fine-grained access control, attribute-based encryption, light weight computation.

I. INTRODUCTION

The recent proliferation of the Internet of Things (IoT) technology has facilitated the improvement of several systems ranging from the current healthcare and assisted living systems to smart city systems [1]. According to the report published by Gartner [2], it is expected that the number of machine to machine (M2M) connections will increase from 5.6 billion in 2016 to more than 27 billion in 2024. Also, the report states that the number of smart devices connected to the internet network will be dramatically increased from 8.4 billion in 2020 to 20 billion in 2022. Moreover, it is predicted that the revenue of the IoT industry will significantly increase from 892 billion in 2018 to four trillion by 2025 [3]. Observing the surges in the numbers, one expects that IoT will become one of the major forthcoming markets making a cornerstone of the expanding digital economy.

However, smart devices and sensors in IoT networks usually suffer from extremely limited processing and storage resources. To address these constraints, a promising way is to interconnect IoT networks with cloud servers to benefit from the elastically scalable and always available storage and computational services offered by the cloud computing paradigm [4], [5]. The cloud-based IoT systems facilitate storage and processing of the collected data, provide user mobility, and make using the same data in multiple services possible.

However, outsourcing the collected data to a cloud server raises concerns over data confidentiality and fine-grained access control, since the cloud cannot be considered as a trusted party. Attribute-based encryption (ABE) [6]–[8] is one of the most promising methods to the mentioned problems. Generally, ABE schemes can be divided into three

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

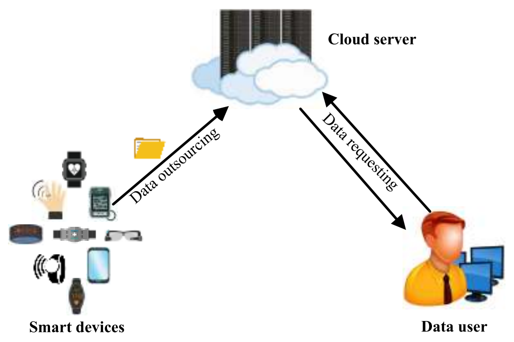

FIGURE 1. A typical cloud-based system.

categories key-policy ABE (KP-ABE) [9], ciphertext-policy ABE (CP-ABE) [10], and dual policy ABE (DP-ABE) [11]. In a KP-ABE scheme, a data user's secret-key is associated with an access control policy defined by the central authority, and each ciphertext is labeled by a set of descriptive attributes determined by a data owner. A data user can recover the data file embedded in a ciphertext only if the attribute set corresponding to the ciphertext satisfies the data user's access control policy. However, in a CP-ABE scheme, a data user's secret-key is associated with an attribute set, and each ciphertext is associated with an access control policy defined by a data owner. A data user can recover the data corresponding to a ciphertext only if its attributes satisfy the access control policy of the ciphertext. DP-ABE schemes can be considered as a combination of KP-ABE and CP-ABE schemes. In such schemes, each ciphertext is associated with both a set of descriptive attributes and an access control policy. Also, each data user's secret-key corresponds to an attribute set and an access control policy defined by the central authority. It seems that CP-ABE schemes offer more flexibility for data owners, as in such schemes, the central authority does not have any role in determining the access rights of data users, and data owners can independently determine the authorized users to access their data.

However, since the sensors and smart devices in IoT networks have limited processing and storage resources, most of the existing ABE schemes are not suitable for providing confidentiality and fine-grained access control in these networks. Also, since in the most of ABE schemes, there is a single key generator authority for providing key delegation and user revocation services, these schemes may get into trouble when many users make queries for their secret-keys. Thus, such schemes seem not to be appropriate for large networks. In this paper, to address the mentioned problems, we put forward a lightweight revocable hierarchical ABE scheme called **LW-RHABE**. Our scheme provides lightweight encryption and decryption approaches. Also, it offers flexible and scalable key delegation and user revocation mechanisms. In the following, we list our main contributions in this work:

- *Lightweight encryption mechanism*: In **LW-RHABE**, the cloud server performs almost all expensive computational operations in the encryption phase without learning any information about the underlying data file, and data owners can encrypt their data, by performing lightweight computations.
- *Lightweight decryption mechanism*: Similar to the encryption phase, most of the expensive operations in the decryption phase can be offloaded onto the cloud server without leaking any information about the underlying data and data users' secret-keys.
- *Flexible access control*: **LW-RHABE** supports access tree as the access control policy. It is known that it provides a high level of flexibility in determining the access rights of data users.
- *Flexible and scalable key delegation and user revocation mechanisms*: In **LW-RHABE**, there are several key generator authorities that each of them can independently handle the key delegation and user revocation phases, and users can request their secret-keys from an arbitrary key authority without any constraints. Also, whenever the system needs more computational resources, the main authority can add some new key generator authorities.
- *Security definition and security proof*: We formalize the system model and the security definition for an **LW-RHABE** scheme. Moreover, we prove that our scheme is secure in the standard model and under the hardness assumption of the DBDH problem.

The remainder of this paper is organized as follows. Section II reviews some related work. In Section III, we describe our system model and threat model. Section IV introduces some required background. The system definition and the security model are presented in Section V. Section VI describes the proposed **LW-RHABE** scheme in detail. In Sections VII and VIII, we present our security and performance analysis. Section IX summarizes our results and concludes this paper. We also prove the correctness of our **LW-RHABE** scheme in Appendix.

II. RELATED WORK

The concept of attribute-based encryption (ABE) was first introduced by Sahai and Waters [12]. In an ABE scheme, a sender can share its data with several expected users without knowing their public-keys. Afterward, three schemes [11], [13], [14] divided ABE schemes into three categories key-policy ABE (KP-ABE), ciphertext-policy ABE (CP-ABE), and dual-policy ABE (DP-ABE). Lewko and Waters [15] proposed a decentralized ABE scheme. In their scheme, there is no requirement for any global authority, and any user can be considered as an authority. Li *et al.* [16] proposed an ABE scheme with verifiable outsourced decryption. In their scheme, the correctness of transformed ciphertexts to authorized users and unauthorized users can be simultaneously verified. Ostrovsky *et al.* [17] designed an ABE scheme supporting non-monotonic access control policies. Lewko *et al.* [18] designed a fully secure functional ABE scheme. Li *et al.* [19] proposed a KP-ABE scheme against continual auxiliary input leakage. Belguith *et al.* [20] designed a multi-authority ABE scheme with hidden access control policies. Li *et al.* [21] designed a revocable user collusion avoidance ABE scheme for cloud storage. Qian *et al.* [22] proposed a revocable attribute-based health record system. Their scheme provides an efficient on-demand user and attribute revocation mechanism. Also, their proposed scheme supports dynamic policy updates. Recently, Li *et al.* [23] proposed an attribute-based file hierarchy access control scheme. Their scheme greatly saves computational resources as it enables a data owner to encrypt multiple data files on the same access level.

However, most of ABE schemes suffer from heavy storage and computational overhead on the user side. It is known that such schemes are not suitable for IoT applications. To address these issues, by employing the elliptic curve cryptography (ECC) algorithm, Yao *et al.* [24] designed a lightweight ABE scheme. Using CP-ABE schemes, Li *et al.* designed a lightweight data sharing system for mobile cloud computing [25]. Guo *et al.* [26] proposed a lightweight CP-ABE scheme with constant secret-key size. Rasori *et al.* [27] designed a lightweight and scalable ABE scheme for smart cities. Li and Jing [28] designed a lightweight searchable ABE scheme for fog-based IoT networks. Miao *et al.* [29] designed a lightweight searchable ABE scheme for fog computing. In [30], Chaudhary *et al.* proposed a software-defined network-enabled multi-attribute secure communication model for an industrial IoT environment. Using a proxy service architecture and a novel CP-ABE scheme, He *et al.* [31] designed a lightweight ABE scheme for mobile cloud-assisted cyber-physical systems. Hao *et al.* [32] proposed an attribute-based fine-grained access control with attribute-hiding policy for cloud-based IoT networks. In their work, by employing randomizable and the garbled Bloom filter techniques, the attribute information is totally hidden, and a fuzzy attribute positioning mechanism is designed to help recipients to locate their attributes and to decrypt ciphertexts successfully. Yang *et al.* [33] proposed a lightweight fine-grained access control system for mobile health systems.

However, in most of the current ABE schemes, there is a single key authority for issuing and revoking users' secret-keys. Therefore, they seem not to be appropriate for large networks. To address this problem, combining two concepts hierarchical identity-based encryption and ABE, Wang *et al.* put forward the concept of hierarchical ABE (HABE) schemes [34], [35]. Afterward, the HABE approach was applied to several cryptographic schemes. Liu *et al.* [36] proposed a time-based HABE scheme. Deng *et al.* [37] designed a CP-HABE scheme with short length ciphertexts. Using the hierarchical method in the key delegation phase, Huang *et al.* [38] designed a cloud-based data collaboration system. Wei *et al.* [39] proposed a revocable hierarchical attribute-based access control system for secure sharing of smart-health records in the public cloud storage. By applying the CP-HABE technique, Wang and Gao [40] designed a regulation scheme for the bitcoin system. In their work, they used the HABE approach to provide user anonymity and also traceability of dishonest users. Guo *et al.* [41] proposed an HABE scheme with continuous auxiliary input leakage. Li *et al.* [42] designed an HABE scheme with continuous leakage-resilience. In their work, they provided the formal definition and the security model for HABE schemes with continuous leakage-resilience. Also, they presented a CP-HABE scheme with continuous leakage-resilience. Luo *et al.* [43] designed an HABE friend discovery scheme for mobile social networks.

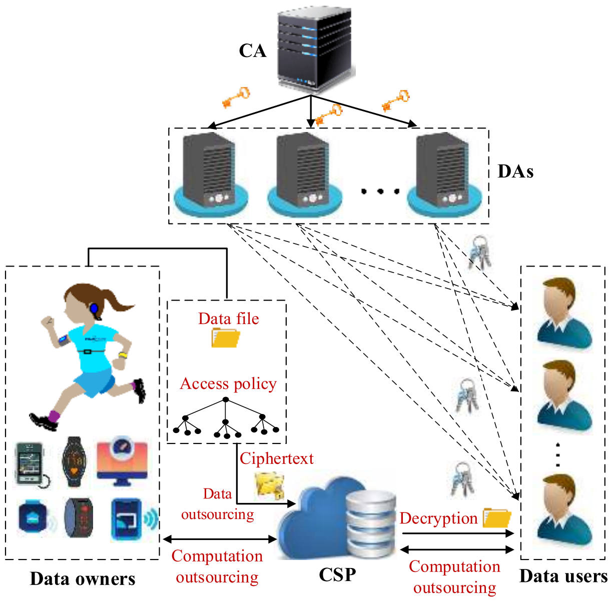

FIGURE 2. Architecture of LW-RHABE.

However, to the best of the authors' knowledge, there is no lightweight ABE providing flexible and scalable key delegation and user revocation mechanisms. The interested reader is referred to [44]–[50] for further information about attribute-based systems and IoT networks.

III. SYSTEM ARCHITECTURE

In this section, we present the system model and threat model of our **LW-RHABE** scheme.

A. SYSTEM MODEL

As shown in Fig. 2, our system consists of five generic entities Central Authority (CA), several Domain Authorities (DAs), the Cloud Service Provider (CSP), several data owners, and several data users. In the following, we describe the mentioned entities:

- CA: This entity is responsible for generating system parameters and initializing the DAs.
- DAs: These entities are responsible for generating secret-keys for data users according to their attributes and revoking them when they miss some attributes. Each DA provides key delegation and user revocation services associated with a specified attribute set. When a data

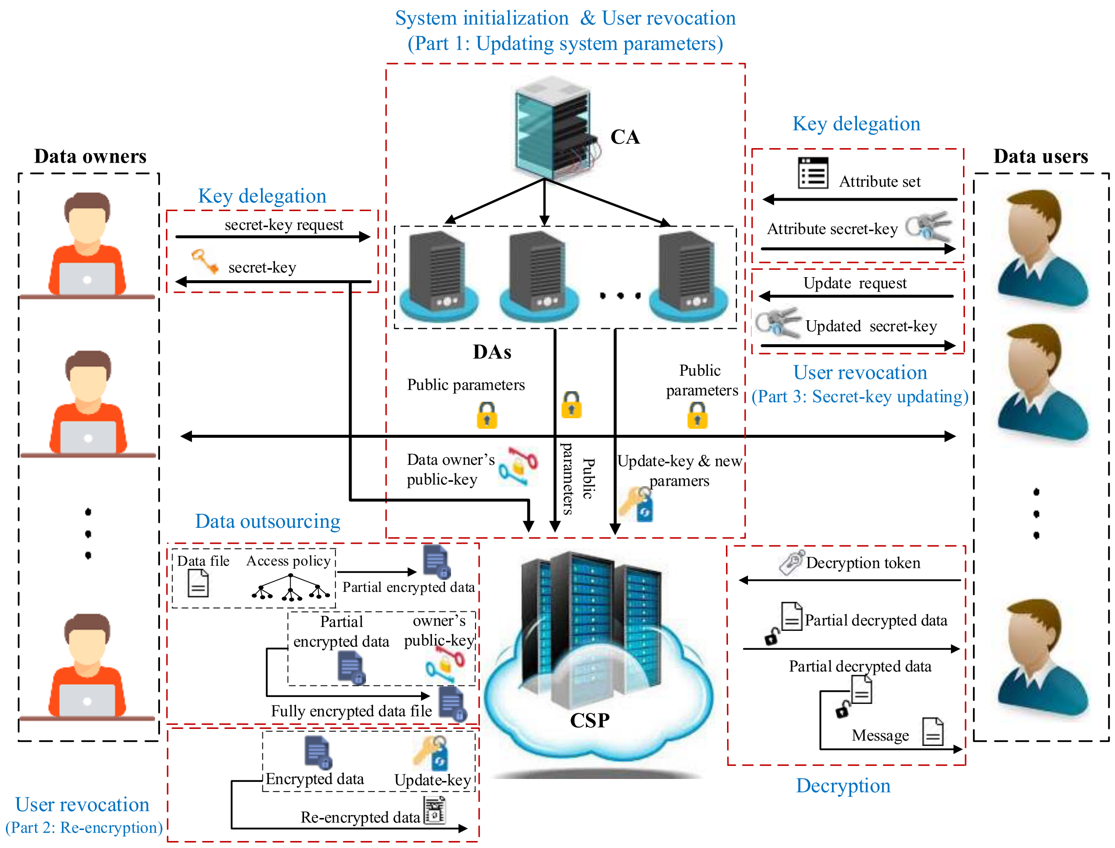

FIGURE 3. Workflow of LW-RHABE.

user possesses an attribute, it selects an arbitrary DA supporting the attribute and asks it to generate the corresponding secret-key. The DA checks if the data user has the attribute or not. If so, it provides the data user with the requested secret-key. Moreover, DAs can help data owners in generating their public-keys and their secret-keys.

- CSP: The CSP possesses almost unlimited storage and computational resources. It provides storage and computational services for data owners and data users. When a data owner wants to encrypt its data, without leaking partial information, it can outsource most of heavy computational operations to the CSP. Also, the CSP can provide data users with computational services. Indeed, in the decryption phase, most of the expensive operations can be outsourced to the CSP such that no information about data users' secret-keys and underlying data files is leaked to the CSP.
- Data owners: Data owners model tiny sensors and smart devices in an IoT network. They possess limited processing and storage resources. They collect data from the environment and encrypt them under an access control policy. The encrypted data is outsourced to the CSP.
- Data users: Data users intend to access the outsourced data files for different purposes such as marketing, research, etc. Using their attribute secret-keys, they can request partial decryption from the CSP. Then, performing lightweight operations, they can recover their intended data files.

As shown in Fig. 3, our scheme consists of five phases *System initialization*, *Key delegation*, *Data outsourcing*, *Decryption*, and *User revocation*. Blow, we give an overview of each phase.

- *System initialization*: The CA manages this phase. It first generates the public parameters of the system and its own master secret-key. Then, it initializes a number of DAs to lighten the burden caused by key delegation and user revocation procedures.
- *Key delegation*: DAs operates this phase. In this phase, they generate secret-keys of data users according to their attributes. Also, they provide data owners with their secret-keys and public-keys.
- *Data outsourcing*: Data owners and the CSP manage this phase. When a data owner wants to outsource its collected data, to provide confidentiality and fine-grained access control, it first defines an access control policy, and then by using the computational power of the CSP, it encrypts the data under the access control policy. The encrypted data file is stored by the CSP for long-term storing and online/offline analysis.

- *Decryption*: The CSP and data users execute this phase. In this phase, using the CSP's computational resources, eligible data users can decrypt the outsourced encrypted data files and obtain their desired data.
- *User revocation*: DAs and the CSP manage this phase. When a data user misses an attribute, DAs update the secret and the public parameters of the system associated with the attribute. Then, the CSP re-encrypts the outsourced data files according to the new parameters without learning any information about the secret parameter and the outsourced data files. Finally, DAs update secret-keys corresponding to the attribute for the authorized data users.

B. THREAT MODEL

The CA and DAs are assumed to be trusted. They never grant unauthorized access rights to data users. Also, they never collude with the other parties. The CSP is assumed to be honest but curious. It does not collude with data users, and it always executes the given protocols correctly. However, it is curious to learn unauthorized information about outsourced data files. Data users are assumed to be malicious. If a data user is authorized for a data file, it does not share the data with the other parties. However, data users may collude with each other to obtain unauthorized information about the outsourced data. Data owners are also assumed to be trusted.

We assumed that the communication channels between the CA and DAs, DAs and DUs, and DAs and the CSP are secure. Data files transmitted through the channels may not be eavesdropped or changed by the other parties. Moreover, it is assumed that there exists no direct communication channel between data owners and data users, and they have to communicate with each other through the CSP. Furthermore, the communication channels between DAs and data owners, the CSP and data owners, and the CSP and data users are assumed to be tamper-resistant. Although the data files transmitted through these channels may be eavesdropped by some malicious parties, they are not definitely tampered with.

IV. PRELIMINARIES

For an algorithm $\mathscr{A}$, assume that $O \leftarrow \mathscr{A}(I)$ denotes running $\mathscr{A}$ on input I and outputting O. Also, for an attribute set S, let $x \leftarrow S$ denote the random selection of x form S. In the following, we give some cryptographic background related to our work.

A. CRYPTOGRAPHIC BACKGROUND

Bilinear map: Consider a prime number q and two cyclic groups G_1 and G_2 of order q. We say that a function $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is a bilinear map if the following conditions hold:

- **Bilinearity**: $\hat{e}(g^a, g^b) = \hat{e}(g^b, g^a) = \hat{e}(g, g)^{ab}$, for each $a, b \in \mathbb{Z}_q$ and $g \in G_1$.
- **Non-degeneracy**: There is a $g \in G_1$ such that $\hat{e}(g, g) \neq 1$.
- **Computability**: There is an efficient algorithm computing $\hat{e}(g, h)$, for any $g, h \in G_1$.

Consider a probabilistic polynomial time (PPT) algorithm $\mathscr{G}$ that $(\lambda, q, G_1, G_2, \hat{e}) \leftarrow \mathscr{G}(1^\lambda)$, where λ is the security parameter of the system and $(q, G_1, G_2, \hat{e})$ is the same as above.

In this work, we consider the Decisional Bilinear Diffie Hellman (DBDH) assumption on $\mathscr{G}$:

The DBDH assumption: Consider $g \leftarrow G_1, \alpha, \beta, \gamma \leftarrow \mathbb{Z}_q$, and $(\lambda, q, G_1, G_2, \hat{e}) \leftarrow \mathscr{G}(1^n)$. This assumption states that for all PPT adversaries $\mathscr{A}$, there is a negligible function *negl* such that:

$$\begin{aligned}|&\Pr(\mathscr{A}(n, q, g, g^\alpha, g^\beta, g^\gamma, g^{\alpha\beta\gamma}, G_1, G_2, \hat{e})=1)\\ &-\Pr(\mathscr{A}(n, q, g, g^\alpha, g^\beta, g^\gamma, g^z, G_1, G_2, \hat{e})=1)| \leq negl(\lambda),\end{aligned} \tag{1}$$

where the probabilities are taken over the selection of $g \in G_1$ and $\alpha, \beta, \gamma, z \in \mathbb{Z}_q$, and the randomness used in $\mathscr{G}$ and $\mathscr{A}$.

B. ACCESS TREES

In an access tree, leaf nodes represent an attribute set, and inner nodes represent a threshold value set. Also, threshold value associated with each leaf node is equal to 1. Assume that $\mathscr{T}$ is an access tree, v_a denotes the leaf node corresponding to an attribute a, k_v denotes the threshold value associated with a node v, $R_{\mathscr{T}}$ denotes root node $\mathscr{T}$, $L_{\mathscr{T}}$ denotes the leaf node set of the access tree, and $\mathscr{T}_v$ denotes the subtree of $\mathscr{T}$ rooted at a node v.

Let $\mathbb{U}$ be the universal attribute set, and $\mathscr{T}$ be an access tree. For a node v in $\mathscr{T}$, consider a function $F_{\mathscr{T}_v} : 2^{\mathbb{U}} \rightarrow \{0, 1\}$ performing as follows:

- When v is the leaf node corresponding to an attribute a, $F_{\mathscr{T}_v}(Att) = 1$ if and only if $a \in Att$.
- When v is an inner node with threshold value k_v, $F_{\mathscr{T}_v}(Att) = 1$ if and only if v has at least k_v children $c_1, \ldots, c_{k_v}$ such that $F_{\mathscr{T}_{c_i}}(Att) = 1$, for $i = 1, \ldots, k_v$.

We say that an attribute set Att satisfies $\mathscr{T}$ if $F_{\mathscr{T}_{R_{\mathscr{T}}}}(Att) = 1$.

For a prime number q and an access tree $\mathscr{T}$, consider an algorithm $\{q_v(0)\}_{v \in L_{\mathscr{T}}} \leftarrow \mathbf{Share}_q(\mathscr{T}, r)$ that shares a secret $r \in \mathbb{Z}_q$ with respect to q and $\mathscr{T}$ as follows:

- Assign a $(k_{R_{\mathscr{T}}} - 1)$-degree polynomial $q_{R_{\mathscr{T}}}$ to root node $R_{\mathscr{T}}$ such that $q_{R_{\mathscr{T}}}(0) = r$, and other coefficients are selected uniformly at random from $\mathbb{Z}_q$.
- For each non-leaf node v with a polynomial q_v, if children of v have not got their polynomials yet, assign a $(k_{c_i} - 1)$-degree polynomial q_{c_i} to the i-th child such that $q_{c_i}(0) = q_v(i)$, and other coefficients of q_{c_i} are selected uniformly at random from $\mathbb{Z}_q$.

When this algorithm stops, a value $q_{v_i}(0)$ is assigned to the leaf node v_i.

V. SYSTEM DEFINITION AND SECURITY MODEL

In this section, we present the system definition and security definition for an **LW-RHABE**.

A. SYSTEM DEFINITION

An **LW-RHABE** scheme consists of twelve PPT algorithms defined as follows:

- **Setup**$(1^\lambda, \mathbb{U})$: The CA executes this algorithm. On input the security parameter and the universal attribute set $\mathbb{U}$, it outputs public parameters *params* and the master secret-key *MSK*.
- **KAGen**$(params, MSK, \{id_j\}_{j=1}^n, \{\mathbb{U}_j\}_{j=1}^n)$: The CA operates this algorithm. It takes as input the public parameters *params*, master secret-key *MSK*, identifiers of the new DAs, $\{id_j\}_{j=1}^n$, and subsets of the universal attribute set, $\{\mathbb{U}_j\}_{j=1}^n$. The algorithm outputs a set of public-keys and master secret-keys $\{PK_j, MSK_j\}_{j=1}^n$ of the new DAs.
- **User.KeyGen**$(params, MSK_j, id_u, i)$: DAs run this algorithm. It inputs the public parameters *params*, master secret-key of the DA running the algorithm, MSK_j, an identifier of a data user, id_u, and an attribute i. It outputs a secret-key $sk_{i,u}^{(j)}$ or an error message $\perp$.
- **Owner.KeyGen**$(params)$: DAs or data owners execute this algorithm. On input the public parameters *params*, it generates a pair of secret-key and public-key (sk_O, pk_O).
- **Owner.Enc**$(params, sk_O, M, \mathcal{T})$: Data owners execute this algorithm. It takes as input the public parameters *params*, a data owner's secret-key sk_O, a message M, and an access tree $\mathcal{T}$. It outputs a partial ciphertext $PCT_{\mathcal{T}}$.
- **CSP.Enc**$(params, pk_O, PCT_{\mathcal{T}})$: The CSP runs this algorithm. Given the public parameters *params*, a data owner's public-key pk_O, and a partial ciphertext $PCT_{\mathcal{T}}$, it outputs a ciphertext $CT_{\mathcal{T}}$.
- **Dec.TokenGen**$(params, id_u, \{sk_{i,u}^{(j_i)}\}_{i\in S})$: A data user runs this algorithm. It takes as input the public parameters *params*, the data user's identifier, id_u, and a secret-key set $\{sk_{i,u}^{(j_i)}\}_{i\in S}$. It returns a decryption token tk_u and a secret s.
- **CSP.Dec**$(params, tk_u, CT_{\mathcal{T}})$: The CSP operates this algorithm. Given public parameters *params*, a decryption token tk_u, and a ciphertext $CT_{\mathcal{T}}$, this algorithm outputs a partially decrypted ciphertext M'.
- **User.Dec**$(params, M', CT_{\mathcal{T}}, s)$: A data user runs this algorithm. On input public parameters *params*, a partially decrypted ciphertext M', a ciphertext $CT_{\mathcal{T}}$, and a secret s, this algorithm returns the message corresponding to the ciphertext.
- **UpdateParams**$(params, MSK_j, i_0)$: DAs executes this algorithm. On input the public parameters *params*, master secret-key of the DA running this algorithm, MSK_j, and an attribute i_0, it returns a secret parameter $\tilde{s}_{i_0}$, a public parameter $\tilde{pk}_{i_0}$, and an update-key $UKey_{i_0}$ associated with attribute i_0.
- **ReEnc**$(params, i_0, CT_{\mathcal{T}}, UKey_{i_0})$: The CSP operates this algorithm. It takes as input the public parameters *params*, an attribute i_0, a ciphertext $CT_{\mathcal{T}}$ possessing a leaf node associated with i_0, and an update-key $UKey_{i_0}$. It outputs a re-encrypted ciphertext $\widetilde{CT}_{\mathcal{T}}$.
- **UpdateKey**$(params, id_u, i_0, sk_{i_0,u}^{(j)}, UKey_{i_0})$: DAs execute this algorithm. Given the public parameters *params*, an identifier id_u, an attribute i_0, a secret-key of the data user associated with the attribute, $sk_{i_0,u}^{(j)}$, and an update-key $UKey_{i_0}$, this algorithm outputs an updated secret-key $\tilde{sk}_{i_0,u}^{(j)}$.

B. SECURITY MODEL

Security of **LW-RHABE** requires that for each PPT adversary that models the CSP or a group of unauthorized data users colluding with each other, the advantage of the adversary in obtaining partial information about the outsourced data files is a negligible function of the security parameter. In other words, no PPT adversary can distinguish the encryption of two data files of its choice. In the following, by using the indistinguishability experiment presented below, we formalize the security requirement.

Indistinguishability game LW-RHABE$_{\mathcal{A},\Pi}(\lambda)$:

Let Π be an **LW-RHABE** scheme and $\mathcal{A}$ be a PPT adversary. Consider the following game:

1) **Setup**: A challenger selects a security parameter λ and a universal attribute set $\mathbb{U}$. It runs $(params, MSK) \leftarrow$ **Setup**$(1^\lambda, \mathbb{U})$ and for a natural number n, it selects n identifiers $id_1, \ldots, id_n$ and n subsets $\mathbb{U}_1, \ldots, \mathbb{U}_n$. Then, it executes $\{PK_i, MSK_i\}_{i=1}^n \leftarrow$ **KAGen**$(params, MSK, \{id_i\}_{i=1}^n, \{\mathbb{U}_i\}_{i=1}^n)$ and gives *params*, $\{id_i\}_{i=1}^n$, and $\{PK_i\}_{i=1}^n$ to the adversary.
2) **Phase 1**: $\mathcal{A}$ makes polynomially many queries to the following oracle, and for each data user with identifier id_u, the challenger keeps a list L_{id_u} which is initially empty.
   $\mathcal{O}_{User.KeyGen}(i, id_j, id_u)$: The challenger runs $sk_{i,u}^{(j)} \leftarrow$ **UKeyGen**$(params, MSK_j, id_u, i)$ and returns $sk_{i,u}^{(j)}$ to the adversary. It also substitutes $L_{id_u} \cup \{i\}$ with L_{id_u}.
3) **Challenge**: The adversary $\mathcal{A}$ declares two equal length messages M_0 and M_1 and an access tree $\mathcal{T}^*$. The challenger checks whether there exists an identifier id_u that L_{id_u} satisfies $\mathcal{T}^*$ or not. If so, the challenger aborts and returns 0. Otherwise, it chooses $b \leftarrow \{0,1\}$ and executes $(sk_O, pk_O) \leftarrow$ **Owner**.**KeyGen**$(params)$ and $PCT_{\mathcal{T}^*}^b \leftarrow$ **Owner**.**Enc**$(params, sk_O, M_b, \mathcal{T}^*)$. The challenger returns PK_O and $PCT_{\mathcal{T}^*}^b$ to $\mathcal{A}$.
4) **Phase 2**: $\mathcal{A}$ makes more queries to the oracle $\mathcal{O}_{User.KeyGen}(i, id_j, id_u)$ and the challenger answers them if and only if $L_{id_u} \cup \{i\}$ does not satisfy $\mathcal{T}^*$.
5) **Guess**: $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.

The output of the game is defined to be 1 if $b = b'$, and 0 otherwise. We say that the adversary $\mathcal{A}$ wins the game, and we write **LW-RHABE**$_{\mathcal{A},\Pi}(\lambda) = 1$ if the output of the game is equal to 1.

Definition 1: An **LW-RHABE** scheme Π is said to be secure if for any PPT adversary $\mathcal{A}$ there exists a negligible function *negl* such that:

$$\Pr(\textbf{LW-RHABE}_{\mathcal{A},\Pi}(\lambda) = 1) \leq \frac{1}{2} + negl(\lambda). \qquad (2)$$

TABLE 1. Notation description.

Symbol	Description
λ	Security parameter of the system
$\mathbb{U}$	Universal attribute set of the system
$params$	Global public parameters of the system
MSK	Master secret-key of the CA
id_j	Identifier of the j-th DA
$\mathbb{U}_j$	The attribute set supported by the j-th DA
PK_j	Public-key of the j-th DA
MSK_j	Master secret-key of the j-th DA
id_u	Identifier of a data user
(sk_O, pk_O)	Secret-key and public-key of a data owner
$\mathscr{T}$	An access tree
M	A message
$PCT_{\mathscr{T}}$	Partial ciphertext associated with an access tree $\mathscr{T}$
$sk_{i,u}^{(j)}$	Secret-key of a data user associated with an attribute i and generated by the j-th DA
$CT_{\mathscr{T}}$	A ciphertext associated with an access tree $\mathscr{T}$
tk_u	A decryption token
s	A private-key generated in the decryption phase
M'	Partially decrypted ciphertext
$UKey_{i_0}$	Update-key associated with an attribute i_0
$\tilde{CT}_{\mathscr{T}}$	Re-encrypted ciphertext
s_i	Secret parameter of the system associated with attribute i
pk_i	Public parameter of the system associated with attribute i
$\tilde{s}_{i_0}$	Updated secret parameter associated with attribute i_0
$\tilde{pk}_{i_0}$	Updated public parameter associated with attribute i_0

VI. OUR CONSTRUCTION

In this section, we present the concrete construction of **LW-RHABE**. The notations employed in this section are described in Table 1. As mentioned in Subsection III-A, **LW-RHABE** consists of five phases *System initialization*, *Key delegation*, *Data outsourcing*, *Decryption*, and *User revocation*. In the following, each of the phases is described in detail.

A. SYSTEM INITIALIZATION

The CA first determines a universal attribute set $\mathbb{U}$ and a security parameter λ. It runs $(params, MSK) \leftarrow$ **Setup**$(1^\lambda, \mathbb{U})$ and generates the public parameters $params$ and master secret-key MSK. Then, for an integer n, it considers n subsets $\{\mathbb{U}_j\}_{j=1}^n$ of $\mathbb{U}$ and selects n identifiers $\{id_j\}_{j=1}^n$. It runs $\{PK_j, MSK_j\}_{j=1}^n \leftarrow$ **KAGen**$(params, MSK, \{id_j\}_{j=1}^n, \{\mathbb{U}_j\}_{j=1}^n)$ and generates public-keys and the master secret-keys of DAs. $params$, $\{id_j\}_{j=1}^n$, and $\{PK_j\}_{j=1}^n$ are published to the other entities, MSK is kept confidential by the CA, and MSK_j is given to the j-th DA. In the following, the two mentioned algorithms are described in detail.

Setup$(1^\lambda, \mathbb{U})$: On input the security parameter λ and the universal attribute set $\mathbb{U}$, this algorithm first runs $(\lambda, q, G_1, G_2, \hat{e}) \leftarrow \mathscr{G}(1^\lambda)$ and then selects $x, y \leftarrow \mathbb{Z}_q$ and $g_1, g_2, g_3, g_4 \leftarrow G_1$. It calculates $h_1 = g_1^x$, $h_2 = g_1^y$, $msk_1 = g_2^x$, $msk_2 = h_1^y$, $P_1 = \hat{e}(g_2, h_1)$ and $P_2 = \hat{e}(h_2, h_1)$. Then, for each $i \in \mathbb{U}$, it selects $s_i \leftarrow \mathbb{Z}_q$ and sets $pk_i = g_1^{s_i}$. It selects a secure symmetric-key scheme $\Pi_S = (\textbf{Enc}, \textbf{Dec})$ and a secure hash function $H : G_2 \to \{0,1\}^m$, for an $m \in \mathbb{Z}^+$. Finally, it returns:

$$params = (\lambda, q, G_1, G_2, \hat{e}, g_1, g_2, g_3, h_1, h_2, \{pk_i\}_{i\in\mathbb{U}}, P_1, P_2, \Pi_S, H), \quad (3)$$

as the global public parameters, and

$$MSK = (x, y, g_4, msk_1, msk_2, \{s_i\}_{i\in\mathbb{U}}), \quad (4)$$

as the master secret-key of the CA.

KAGen$(params, MSK, \{id_j\}_{j=1}^n, \{\mathbb{U}_j\}_{j=1}^n)$: It first selects $s_j' \leftarrow \mathbb{Z}_q$ and computes

$$SK_j = msk_1 g_4 id_j^{s_j'} \quad (5)$$

and

$$PK_j = msk_2 g_4 id_j^{s_j'}, \quad (6)$$

for $j = 1, \ldots, n$. Then, it returns $\{PK_j, MSK_j\}_{j=1}^n$, where $MSK_j = (SK_j, \{s_j\}_{i\in\mathbb{U}_j})$.

B. KEY DELEGATION

In this phase, DAs generate data users' secret-keys according to their attributes, and they issue public-keys and secret-keys of data owners. When a data user possesses an attribute $i \in \mathbb{U}$, it should ask a DA supporting the attribute to generate its secret-key. When a DA with identifier id_j receives a request (i, id_u) from a data user, it first checks whether the data user has the attribute i or not. If not, it aborts. Otherwise, it executes $sk_{i,u}^{(j)} \leftarrow$ **User**.**KeyGen**$(params, MSK_j, id_u, i)$ and gives $sk_{i,u}^{(j)}$ to the data user. Also, when a data owner asks a DA to generate its public-key and secret-key, the DA executes $(sk_O, pk_O) \leftarrow$ **Owner**.**KeyGen**$(params)$ and gives sk_O to the data owner. It also outsources pk_O to the CSP. Note that public-key and secret-key of a data owner can be generated by itself. However, as they have limited computational resources, we assume that this service is provided by DAs.

User.KeyGen$(params, MSK_j, id_u, i)$: Given $MSK_j = (SK_j, \{s_i\}_{i\in\mathbb{U}_j})$ and an attribute i, this algorithm checks whether $i \in \mathbb{U}_j$ or not. If so, it returns

$$sk_{i,u}^{(j)} = SK_j id_u^{s_i}. \quad (7)$$

Otherwise, it returns an error message $\perp$.

Owner.KeyGen$(params)$: It first selects $sk_O \leftarrow \mathbb{Z}_q$ and sets $pk_O^{(1)} = P_2^{sk_O}$, $pk_O^{(2)} = g_1^{sk_O}$, $pk_O^{(3)} = g_3^{sk_O}$, and $pk_{i,O} = pk_i^{sk_O}$. It returns (sk_O, pk_O), where $pk_O = (pk_O^{(1)}, pk_O^{(2)}, pk_O^{(3)}, \{pk_{i,O}\}_{i\in\mathbb{U}})$.

C. DATA OUTSOURCING

In this phase, a data owner with a data file M and a secret-key sk_O encrypts its data and stores the generated ciphertext to the CSP. At first, by defining an access tree $\mathscr{T}$, it determines authorized data users to access its data. Then, it runs $PCT_{\mathscr{T}} \leftarrow$ **Owner**.**Enc**$(params, sk_O, M, \mathscr{T})$ and returns a partial ciphertext $PCT_{\mathscr{T}}$ to the CSP. The CSP considers the data owner's public-key pk_O and generates a ciphertext $CT_{\mathscr{T}} \leftarrow$ **CSP**.**Enc**$(params, pk_O, PCT_{\mathscr{T}})$. The mentioned two algorithms are described as follows:

Owner.Enc$(params, sk_O, M, \mathscr{T})$: It selects $r \leftarrow \mathbb{Z}_q$ and computes $r' = r - sk_O$ and

$$k = H(P_1^r). \quad (8)$$

Then, it runs $\{q_{v_i}(0)\}_{v_i \in L_{\mathscr{T}}} \leftarrow$ **Share**$_q(\mathscr{T}, r')$ and $C \leftarrow$ **Enc**(M, k). Finally, it outputs the partial ciphertext $PCT_{\mathscr{T}} = (C, r', \mathscr{T}, \{q_{v_i}(0)\}_{v_i \in L_{\mathscr{T}}})$.

CSP.Enc$(params, pk_O, PCT_{\mathscr{T}})$: Given $pk_O = (pk_O^{(1)}, pk_O^{(2)}, pk_O^{(3)}, \{pk_{i,O}\}_{i \in \mathbb{U}})$ and $PCT_{\mathscr{T}} = (C, r', \mathscr{T}, \{q_{v_i}\}_{v_i \in L_{\mathscr{T}}})$, this algorithm calculates

$$C' = g_3^{r'} pk_O^{(3)} = g_3^r, \tag{9}$$

$$C'' = P_2^{r'} pk_O^{(1)} = P_2^r, \tag{10}$$

and for each $v_i \in L_{\mathscr{T}}$, it sets

$$C_i = g_1^{q_{v_i}(0)} pk_O^{(2)} = g_1^{q_{v_i}(0)+sk_O},$$

$$C'_i = (pk_i g_3^{-1})^{q_{v_i}(0)} pk_{i,O} (pk_O^{(3)})^{-1} = (pk_i g_3^{-1})^{q_{v_i}(0)+sk_O}. \tag{11}$$

It returns

$$CT_{\mathscr{T}} = (\mathscr{T}, C, C', C'', \{C_i\}_{v_i \in L_{\mathscr{T}}}, \{C'_i\}_{v_i \in L_{\mathscr{T}}}). \tag{12}$$

D. DECRYPTION

When a data user with attribute set Att_u wants to access an outsourced data file corresponding to a ciphertext $CT_{\mathscr{T}}$, it first checks if there is a subset $S \subseteq Att_u$ satisfying $\mathscr{T}$ or not. If not, it aborts. Otherwise, it considers the secret-key set $\{sk_{i,u}^{(j_i)}\}_{i \in S}$ and runs $(s, tk_u) \leftarrow$ **Dec.TokenGen**$(params, id_u, \{sk_{i,u}^{(j_i)}\}_{i \in S})$. tk_u is given to the CSP, and s is kept confidential by the data user. The CSP runs $M' \leftarrow$ **CSP.Dec**$(params, tk_u, CT_{\mathscr{T}})$ and returns a partially decrypted ciphertext M' to the data user. Finally, the data user recovers the corresponding data by running $M \leftarrow$ **User.Dec**$(params, M', CT_{\mathscr{T}}, s)$. Below, we describe the three mentioned algorithms.

Dec.TokenGen$(params, id_u, \{sk_{i,u}^{(j_i)}\}_{i \in S})$: This algorithm selects $s \leftarrow \mathbb{Z}_q$ and returns the secret s and a decryption token $tk_u = \{t_u, tk_{i,u}, tk'_{i,u}\}_{i \in S}$, where $t_u = id_u^s$, $tk_{i,u} = (sk_{i,u}^{(j_i)})^s$, and $tk'_{i,u} = C_i^s$.

CSP.Dec$(params, tk_u, CT_{\mathscr{T}})$: Given $tk_u = \{t_u, tk_{i,u}, tk'_{i,u}\}_{i \in S}$ and $CT_{\mathscr{T}} = (\mathscr{T}, C, C', C'', \{C_i\}_{v_i \in L_{\mathscr{T}}}, \{C'_i\}_{v_i \in L_{\mathscr{T}}})$, this algorithm first calculates

$$\frac{\hat{e}(tk_{i,u}, C_i)}{\hat{e}(PK_{j_i}, tk'_{i,u})\hat{e}(t_u, C'_i)} = P_1^{s(q_{v_i}(0)+sk_O)} P_2^{-s(q_{v_i}(0)+sk_O)} \times \hat{e}(t_u, g_3)^{s(q_{v_i}(0)+sk_O)}, \tag{13}$$

for each $i \in S$. Then, by using polynomial interpolation technique, it computes:

$$C_{r,s} = P_1^{sr} P_2^{-sr} \hat{e}(t_u, C'). \tag{14}$$

Finally, it returns partially decrypted ciphertext

$$M' = C_{r,s}\hat{e}(t_u, C')^{-1} = P_1^{sr} P_2^{-sr}. \tag{15}$$

User.Dec$(params, M', CT_{\mathscr{T}}, s)$: It first computes

$$k = H(M'^{s^{-1}} C'') \tag{16}$$

and then returns

$$M = \mathbf{Dec}(k, C). \tag{17}$$

E. USER REVOCATION

When a data user misses an attribute i_0, it should be made sure that the data user no longer has a valid secret-key associated with i_0. To accomplish this, the following steps are performed:

- DAs update the secret parameter s_{i_0} and public parameter pk_{i_0}.
- According to the new parameters, the CSP re-encrypts ciphertexts with a leaf node associated with i_0 in their access trees.
- According to the new parameters, DAs update the authorized data users' secret-keys associated with i_0.

In practice, a DA first executes $(\tilde{s}_{i_0}, \tilde{pk}_{i_0}, Ukey_{i_0}) \leftarrow$ **UpdateParams**$(params, MSK_j, i_0)$ and returns a new secret parameter $\tilde{s}_{i_0}$, a new public parameter $\tilde{pk}_{i_0}$, and an update-key $Ukey_{i_0}$. $\tilde{s}_{i_0}$ is given to the other DAs, $\tilde{pk}_{i_0}$ is published to the system, and $Ukey_{i_0}$ is given to the CSP and DAs. Then, for each ciphertext $CT_{\mathscr{T}}$ with a leaf node associated with i_0, by running $\widetilde{CT}_{\mathscr{T}} \leftarrow$ **ReEnc**$(params, i_0, CT_{\mathscr{T}}, UKey_{i_0})$, the CSP re-encrypts $CT_{\mathscr{T}}$ to $\widetilde{CT}_{\mathscr{T}}$. Also, when an authorized data user with identifier id_u asks a DA to update its secret-key $sk_{i_0,u}^{(j)}$, the DA runs $\tilde{sk}_{i_0,u}^{(j)} \leftarrow$ **UpdateKey**$(params, id_u, i_0, sk_{i_0,u}^{(j)}, UKey_{i_0})$ and returns $\tilde{sk}_{i_0,u}^{(j)}$ to the data user. The mentioned algorithms are performed as follows:

UpdateParams$(params, MSK_j, i_0)$: This algorithm checks whether $i_0 \in \mathbb{U}_j$ or not. If not, it returns $\perp$. Otherwise, it selects $\tilde{s}_{i_0} \leftarrow \mathbb{Z}_q$ and computes $\tilde{pk}_{i_0} = g_1^{\tilde{s}_{i_0}}$ and $Ukey_{i_0} = \tilde{s}_{i_0} - s_{i_0}$. It returns $\tilde{s}_{i_0}, \tilde{pk}_{i_0}$, and $Ukey_{i_0}$.

ReEnc$(params, i_0, CT_{\mathscr{T}}, UKey_{i_0})$: Given an attribute i_0 a ciphertext $CT_{\mathscr{T}} = (\mathscr{T}, C, C', C'', \{C_i\}_{v_i \in L_{\mathscr{T}}}, \{C'_i\}_{v_i \in L_{\mathscr{T}}})$ such that $\mathscr{T}$ has a leaf node associated with i_0, and an update-key $UKey_{i_0}$, this algorithm outputs a re-encrypted ciphertext $\widetilde{CT}_{\mathscr{T}} = (\mathscr{T}, C, C', C'', \{C_i\}_{v_i \in L_{\mathscr{T}}}, \{C'_i\}_{v_i \in L_{\mathscr{T}} \setminus \{v_{i_0}\}} \cup \{\tilde{C}'_{i_0}\})$, where

$$\tilde{C}'_{i_0} = C'_{i_0} C_{i_0}^{UKey_{i_0}}. \tag{18}$$

UpdateKey$(params, id_u, i_0, sk_{i_0,u}^{(j)}, UKey_{i_0})$: This algorithm updates a secret-key $sk_{i_0,u}^{(j)}$ associated with an attribute i_0 as follows:

$$\tilde{sk}_{i_0,u}^{(j)} = sk_{i_0,u}^{(j)} id_u^{UKey_{i_0}}. \tag{19}$$

VII. SECURITY ANALYSIS

In this section, we prove that **LW-RHABE** is secure in the standard model.

Theorem 2: If the **DBDH** problem is hard relative to $\mathscr{G}$, then our construction is secure in the standard model.

Proof: Let Π be our **LW-RHABE** scheme and $\mathscr{A}$ be a PPT adversary in the game **LW-RHABE**$_{\mathscr{A},\Pi}(\lambda)$, where λ is the security parameter of the system. In the following, we prove that

$$\Pr(\mathbf{LW\text{-}RHABE}_{\mathscr{A},\Pi}(\lambda) = 1) \leq \frac{1}{2} + negl(\lambda). \tag{20}$$

for a negligible function $negl$.

TABLE 2. Feature comparison.

Schemes	Flexible key delegation and user revocation	Lightweight encryption	Lightweight decryption	Security model	Standard model
Yao *et al.* [24]	No	No	No	Selective	Yes
Rasori *et al.* [27]	No	No	No	\	\
Liu *et al.* [36]	No	No	No	\	\
Deng *et al.* [37]	No	No	No	Adaptive	Yes
Huang *et al.* [38]	No	No	Yes	\	\
Yu *et al.* [51]	No	No	No	Selective	Yes
Waters *et al.* [15]	No	No	No	Adaptive	Yes
Ruj *et al.* [52]	No	No	No	Selective	No
Hur *et al.* [53]	No	No	No	Selective	No
Yang *et al.* [54]	No	No	No	Selective	No
Xu *et al.* [55]	No	No	Yes	Selective	Yes
Jung *et al.* [56]	No	No	No	Selective	No
Lai *et al.* [57]	No	No	Yes	Adaptive	No
Li *et al.* [58]	No	No	No	Adaptive	Yes
Lin *et al.* [59]	No	No	Yes	Selective	Yes
LW-RHABE	Yes	Yes	Yes	Adaptive	Yes

Consider another PPT adversary $\mathscr{B}$ aiming to solve the **DBDH** problem. $\mathscr{B}$ receives a tuple $(q, G_1, G_2, \hat{e}, g, g^{\alpha}, g^{\beta}, g^{\gamma}, h = \hat{e}(P, P)^{z})$ from a challenger, where $(q, G_1, G_2, \hat{e}) \leftarrow \mathscr{G}(1^{\lambda})$, $\alpha, \beta, \gamma \leftarrow \mathbb{Z}_q$, $g \leftarrow G_1$, and z is either equal to $\alpha\beta\gamma$ or is chosen uniformly at random from $\mathbb{Z}_q$. $\mathscr{B}$ attempts to determine the case of z. It executes $\mathscr{A}$ as a subroutine as follow:

1) **Setup**: $\mathscr{B}$ chooses $t_1, t_2 \leftarrow \mathbb{Z}_q$ and calculates:

$$g_1 = g \quad (21)$$
$$g_2 = g^{\gamma} \quad (22)$$
$$g_3 = g^{t_2} \quad (23)$$
$$h_1 = g^{\alpha} \quad (24)$$
$$h_2 = g^{\gamma - t_1} \quad (25)$$
$$P_1 = \hat{e}(h_1, g_2) \quad (26)$$
$$P_2 = \hat{e}(h_2, h_1). \quad (27)$$

Then, it selects a universal attribute set $\mathbb{U}$ and selects $s_i \leftarrow \mathbb{Z}_q$, for each $i \in \mathbb{U}$. It returns the system public parameters $params = (\lambda, q, G_1, G_2, \hat{e}, g_1, g_2, g_3, h_1, h_2, \{PK_i\}_{i\in\mathbb{U}}, P_1, P_2, \Pi_S, H)$ to $\mathscr{A}$, where H and Π_S are a secure hash function and a secure symmetric key scheme, respectively. Then, $\mathscr{B}$ selects $g' \leftarrow G_1$ and assumes that, for an unknown value $g_4 \in G_1$, the following equation holds

$$g' = g^{\alpha\gamma} g_4. \quad (28)$$

Also, it assumes that for unknown values $\alpha, \gamma - t_1, g^{\alpha\beta}$, and msk_2, $MSK = (x = \alpha, y = \gamma - t_1, g_4, msk_1 = g^{\alpha\gamma}, msk_2, \{sk_i\}_{i\in\mathbb{U}})$ is the master secret-key associated with $params$. Then, for a natural number n, $\mathscr{B}$ selects n subsets $\mathbb{U}_1, \ldots, \mathbb{U}_n$, n identifiers $\{id_j\}_{j=1}^{n}$, and n random elements $\{s'_j\}_{i=j}^{n} \subset \mathbb{Z}_q$, and for each $j = 1, \ldots, n$, computes:

$$SK_j = g' id_j^{s'_j} \quad (29)$$

and

$$PK_j = g' g^{\alpha - t_1} id_j^{sk_j}. \quad (30)$$

$\mathscr{B}$ gives $params$, $\{id_j\}_{j=1}^{n}$, and $\{PK_j\}_{j=1}^{n}$ to the adversary $\mathscr{A}$.

Note that, in the scheme presented in Section VI, we had $h_2 = g_1^{y}$, $h_1 = g_1^{x}$, $msk_1 = g_2^{x}$, and $msk_2 = h_1^{y}$, where $x, y \leftarrow \mathbb{Z}_q$. Therefore, assuming $h_2 = g_1^{\gamma - t_1}$, $h_1 = g_1^{\alpha}$ and $g_2 = g_1^{\gamma}$, we see that $y = \gamma - t_1$. Thus, $msk_1 = g_2^{x} = g^{\gamma\alpha}$, $msk_2 = h_1^{y} = g^{\alpha\gamma - t_1\alpha}$, and

$$SK_j = g' id_j^{s'_j} = g^{\alpha\gamma} g_4 id_j^{s'_j} = msk_1 g_4 id_j^{s'_j}. \quad (31)$$

comparing Equation (31) with (5), one concludes that SK_i is selected correctly. Moreover,

$$\begin{aligned} PK_j &= g' g^{\alpha - t_1} id_j^{sk_j} \\ &= g^{\alpha\gamma} g_4 g^{\alpha - t_1} id_j^{sk_j} \\ &= g^{\alpha(\gamma - t_1)} g_4 id_j^{sk_j} \\ &= msk_2 g_4 id_j^{sk_j}. \end{aligned} \quad (32)$$

By Equation (6), we conclude that PK_j is also chosen correctly.

2) **Phase 1**: When $\mathscr{A}$ makes a query on $\mathscr{O}_{User.KeyGen}(i, id_j, id_u)$, $\mathscr{B}$ returns

$$sk_{i,u}^{(j)} = SK_j id_u^{s_i}, \quad (33)$$

to $\mathscr{A}$ and adds i into L_{id_u}.

3) **Challenge**: $\mathscr{A}$ declares two equal length plaintexts M_0 and M_1, and an access tree $\mathscr{T}^*$ such that there is no list L_{id_u} satisfying $\mathscr{T}$. When $\mathscr{B}$ receives the plaintexts, it selects $r' \leftarrow \mathbb{Z}_q$ and assumes that for an unknown value $sk_O \in \mathbb{Z}_q$, $r' = \beta - sk_O$. It calculates

$$pk_O^{(1)} = h\hat{e}(g^{\alpha}, g^{\beta})^{-t_1} P_2^{-r'}, \quad (34)$$
$$pk_O^{(2)} = g^{-r'} g^{\gamma} = g^{sk_O - \gamma} g^{\gamma} = g^{sk_O} = g_1^{sk_O}, \quad (35)$$
$$pk_O^{(3)} = (pk_O^{(2)})^{t_2} = (g^{sk_O})^{t_2} = (g^{t_2})^{sk_O} = g_3^{sk_O}, \quad (36)$$

and then for each $i \in \mathbb{U}$, it sets

$$pk_{i,O} = (pk_O^{(2)})^{s_i} = (g_1^{s_i})^{sk_O} = pk_i^{sk_O}. \tag{37}$$

Afterwards, it computes

$$k = H(h) \tag{38}$$

and runs $C \leftarrow$ **Enc**(M_b, k), where $b \leftarrow \{0, 1\}$. Finally, it runs $\{q_{v_i}\}_{v_i \in L_{\mathcal{T}^*}} \leftarrow$ **Share**$(r', \mathcal{T}^*)$ and returns $PCT_{\mathcal{T}^*} = (C, r', \mathcal{T}^*, \{q_{v_i}\}_{i \in L_{\mathcal{T}^*}})$ and $pk_O = (pk_O^{(1)}, pk_O^{(2)}, pk_O^{(3)}, \{pk_{i,O}\}_{i\in\mathbb{U}})$ to $\mathscr{A}$.
Note that, if $h = \hat{e}(g, g)^{\alpha\beta\gamma}$, then

$$\begin{aligned} pk_O^{(1)} &= h\hat{e}(g^\alpha, g^\beta)^{-t_1} P_2^{-r'} \\ &= \hat{e}(g, g)^{\alpha\beta\gamma}\hat{e}(g^\alpha, g^\beta)^{-t_1}\hat{e}(h_2, h_1)^{sk_O-\beta} \\ &= \hat{e}(g, g)^{\alpha\beta\gamma}\hat{e}(g^\alpha, g^\beta)^{-t_1}\hat{e}(g, g)^{\alpha(\gamma-t_1)(sk_O-\beta)} \\ &= \hat{e}(g, g)^{\alpha(\gamma-t_1)sk_O} \\ &= \hat{e}(g^\alpha, g^{\gamma-t_1})^{sk_O} \\ &= \hat{e}(h_1, h_2)^{sk_O} \\ &= P_2^{sk_O} \end{aligned} \tag{39}$$

and

$$\begin{aligned} k &= H(h) = H(\hat{e}(g, g)^{\alpha\beta\gamma}) = H(\hat{e}(g^\alpha, g^\gamma)^\beta) \\ &= H(\hat{e}(h_1, g_2)^\beta). \end{aligned} \tag{40}$$

Therefore, assuming $h = \hat{e}(g, g)^{\alpha\beta\gamma}$ and the randomness r in the **Owner.Enc** algorithm is equal to the unknown value β, one concludes that pk_O is a valid public-key associated with sk_O, and $PCT_{\mathcal{T}^*}$ is a valid partial ciphertext associated with $\mathcal{T}^*$.

4) **Phase 2:** $\mathscr{A}$ makes more queries, and $\mathscr{B}$ answers them as before.
5) **Guess**: $\mathscr{A}$ returns a bit $b' \in \{0, 1\}$, and $\mathscr{B}$ outputs 1 if and only if $b' = b$.

As we have seen, if $h = \hat{e}(g, g)^{\alpha\beta\gamma}$, then the ciphertext and the public-key given to $\mathscr{A}$ are valid. Therefore, if $h = \hat{e}(g, g)^{\alpha\beta\gamma}$, the advantage of $\mathscr{B}$ in solving the DBDH problem is equal to the advantage of $\mathscr{A}$ in winning the game **LW-RHABE**$_{\mathscr{A},\Pi}(\lambda)$. In other words,

$$\begin{aligned} &Pr(\mathscr{B}(g, g^\alpha, g^\beta, g^\gamma, \hat{e}(g, g)^{\alpha\beta\gamma}) = 1) \\ &\qquad = Pr(\mathbf{LW\text{-}RHABE}_{\mathscr{A},\Pi}(\lambda)). \end{aligned} \tag{41}$$

Also, when z is a uniform element of $\mathbb{Z}_q$, $h = \hat{e}(g, g)^z$ is also a uniform element of G_2. Therefore, assuming that H and Π_S are secure, one can see that $PCT_{\mathcal{T}}$ does not leak any partial information about M_b and therefore,

$$Pr(\mathscr{B}(g, g^\alpha, g^\beta, g^\gamma, \hat{e}(g, g)^z)) = 1) = \frac{1}{2}. \tag{42}$$

Combining the hardness assumption of the DBDH problem and two Equations (41) and (42), we conclude that

$$Pr(\mathbf{LW\text{-}RHABE}_{\mathscr{A},\Pi}(\lambda)) \leq \frac{1}{2} + negl(\lambda), \tag{43}$$

for a negligible function $negl$. This proves the theorem. ■

TABLE 3. Notations used in our asymptotic analysis.

Symbol	Description
T_{e_1}	Time of an exponential operation in G_1
T_{e_2}	Time of an exponential operation in G_2
T_p	Pairing operation time
L_{G_1}	Size of an element in G_1
L_{G_2}	Size of an element in G_2
$\lvert l_C\rvert$	Size of a ciphertext in Π_S
$L_{\mathbb{Z}_q}$	Size of an element in $\mathbb{Z}_q$
$\lvert Att_u\rvert$	Data user's attribute set carnality
$\lvert L_{\mathcal{T}}\rvert$	Number of leaf nodes in an access tree $\mathcal{T}$
$\lvert S\rvert$	Carnality of the attribute set satisfying a given access tree
$\lvert\mathbb{U}\rvert$	Carnality of the universal attribute set

VIII. PERFORMANCE ANALYSIS

In this section, we first analyze the performance of **LW-RHABE** by comparing its features with those provided by some similar work. Then, we compare the running time, storage cost, and communication overhead of **LW-RHABE** with the schemes presented in [57], [59]. We present both the asymptotic analysis and actual execution time. The notations employed in this section are given in Table 3.

In our asymptotic analysis, the execution time is calculated in terms of three types of expensive computational operations: the paring operation, the exponential operation in G_1, and the exponential operation in G_2. We ignore the other operations as they are significantly more efficient than the three operations.

We also measure the actual execution time and storage cost by using an Ubuntu 18.04 laptop with an Intel Core i5-2410M Processor 2.3 GHz, 6 GB RAM using python Pairing-Based Cryptography (pyPBC) library [60], and Type A pairings. Moreover, we consider the XOR scheme [61] as the symmetric-key encryption scheme, the SHA-1 algorithm as the hash function, and And-gate access structures as access control policies. For a prime number p that $p = 3 \bmod 4$, Type A pairings are constructed on the curve $E(\mathbb{F}_p) : y^2 = x^3 + x$ over the field $\mathbb{F}_p$ [62]. Assuming that $(q, G_1, G_2, \hat{e})$ is the same as before, in this case, q is a factor of $p + 1$, and G_1 and G_2 are subgroups of $E(\mathbb{F}_p)$ and $E(\mathbb{F}_{p^2})$, respectively. In this section, it is assumed that p and q are a 512-bit and a 160-bit prime numbers, respectively. Thus, according to notations given in Table 3, we have $l_{\mathbb{Z}_q} = 160$, $l_{G_1} = 512$, and $l_{G_2} = 1024$.

A. CHARACTERISTICS COMPARISON

As shown in Table 2, **LW-RHABE** is the only scheme providing flexible key delegation and user revocation mechanisms. Indeed, in other multi-authority schemes, either for each attribute in the universe, there is only one key generator authority supporting the attribute, or for each data user in the system, there is only one key authority supporting the data user. However, as shown in the previous section, in **LW-RHABE**, there is no restriction for data users and DAs. Also, only **LW-RHABE** can provide a lightweight encryption mechanism. Moreover, we see that [38], [55], [57], [59], and **LW-RHABE** offer lightweight decryption mechanisms.

TABLE 4. Communication overhead from data owners to the CSP.

Schemes	Communication overhead
Lai et al. [57]	$(4\|L_{\mathscr{T}}\|+3)l_{G_1}+2l_{G_2}$
Lin et al. [59]	$(\|L_{\mathscr{T}}\|+1)l_{G_1}$
LW-RHABE	$l_C+(\|L_{\mathscr{T}}\|+1)l_{\mathbb{Z}_q}$

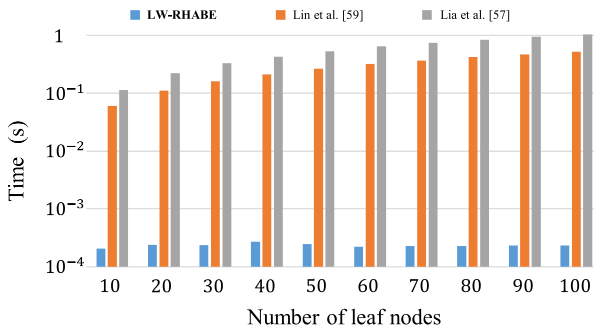

FIGURE 4. Encryption time on the data owner side.

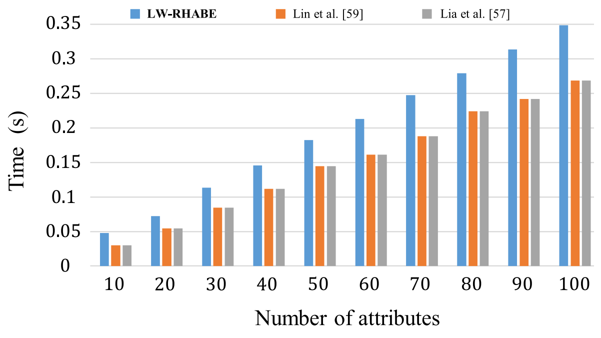

FIGURE 5. Decryption time on the data user side.

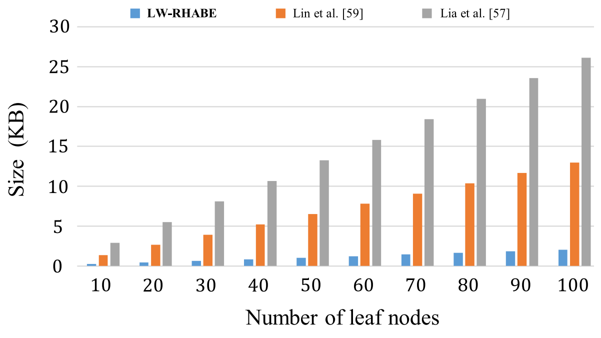

FIGURE 6. Communication overhead from data owners to the CSP.

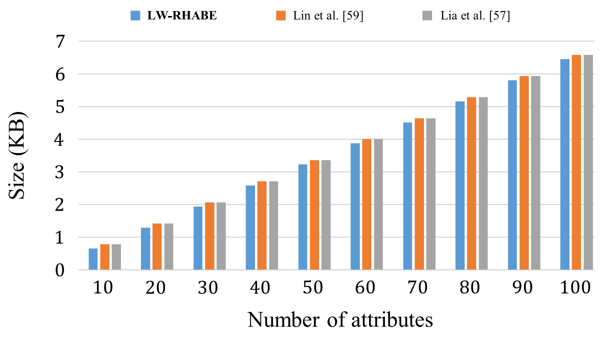

FIGURE 7. Size of a data user's secret-key.

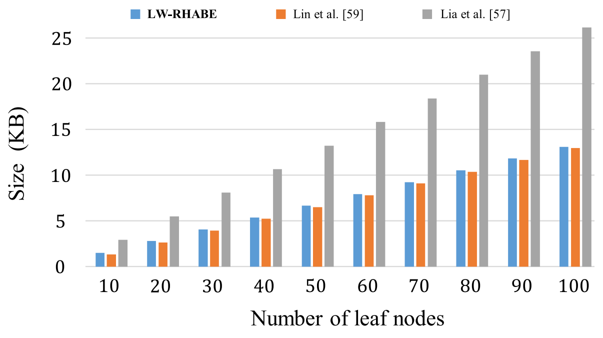

FIGURE 8. Size of a ciphertext.

TABLE 5. Execution overhead incurred by data users and data owners.

Schemes	Encryption time	Decryption time
Lai *et al.* [57]	$(6\|L_{\mathscr{T}}\|+4)T_{e_1}+2T_{e_2}$	$(2\|S\|+1)T_p+\|S\|T_{e_2}$
Lin *et al.* [59]	$(2\|L_{\mathscr{T}}\|+1)T_{e_1}$	$(2\|S\|+1)T_p+\|S\|T_{e_2}$
LW-RHABE	T_{e_2}	$(2\|S\|+1)T_{e_1}$

TABLE 6. Storage overhead.

Schemes	Secret-key length	Ciphertext length
Lai *et al.* [57]	$(\|Att_u\|+2)l_{G_1}$	$(4\|L_{\mathscr{T}}\|+3)l_{G_1}+2l_{G_2}$
Lin *et al.* [59]	$(\|Att_u\|+2)l_{G_1}$	$(\|L_{\mathscr{T}}\|+1)l_{G_1}$
LW-RHABE	$\|Att_u\|l_{G_1}$	$(2\|L_{\mathscr{T}}\|+1)l_{G_1}+l_C+l_{G_2}$

In the other schemes, the decryption phase is costly for data users. Also, we see that only schemes [15], [37], [58], and **LW-RHABE** have adaptive security in the standard model.

B. ASYMPTOTIC AND EXPERIMENTAL RESULTS

As shown in Fig. 4, our proposed encryption mechanism is significantly more efficient than the schemes presented in [57] and [59]. The mentioned fact is confirmed by the data given in Table 5. Moreover, in Fig. 5, we see that the decryption time in **LW-RHABE** is almost the same as the schemes [57] and [59] that both of them provide lightweight decryption mechanisms.

Also, as we see in Fig. 6, our scheme significantly decreases the communication overhead from data owners to the CSP. The reason is that, in **LW-RHABE**, a data owner transfers only lightweight partial ciphertexts to the CSP instead of full ciphertexts. Furthermore, observing the given data in Table 6 and Figures 7 and 8, one concludes that the storage overhead in **LW-RHABE** is acceptable.

IX. CONCLUSION

We designed a revocable lightweight hierarchical attribute-based encryption scheme for IoT networks. In our scheme by performing very efficient computational operations, data owners modeling tiny sensors and smart devices in an IoT

$$
\begin{aligned}
\frac{\hat{e}(tk_{i,u}, C_i)}{\hat{e}(PK_{j_i}, tk'_{i,u})\hat{e}(t_u, C'_i)} &= \frac{\hat{e}((sk_{i,u}^{(j_i)})^s, C_i)}{\hat{e}(PK_{j_i}, tk'_{i,u})\hat{e}(t_u, C'_i)} \\
&= \frac{\hat{e}(SK_{j_i}^s id_u^{ss_i}, C_i)}{\hat{e}(PK_{j_i}, tk'_{i,u})\hat{e}(t_u, C'_i)} \\
&= \frac{\hat{e}(msk_1^s g_4^s id_{j_i}^{ss'_{j_i}} id_u^{ss_i}, g_1^{q_{v_i}(0)+sk_O})}{\hat{e}(PK_{j_i}, tk'_{i,u})\hat{e}(t_u, C'_i)} \\
&= \frac{\hat{e}(g_2^{sx} g_4^s id_{j_i}^{ss'_{j_i}} id_u^{ss_i}, g_1^{q_{v_i}(0)+sk_O})}{\hat{e}(PK_{j_i}, tk'_{i,u})\hat{e}(t_u, C'_i)} \\
&= \frac{\hat{e}(g_2^x, g_1^{q_{v_i}(0)+sk_O})^s \hat{e}(h_1^{-y} h_1^y g_4 id_{j_i}^{s'_{j_i}}, g_1^{q_{v_i}(0)+sk_O})^s \hat{e}(id_u^{s_i}, g_1^{q_{v_i}(0)+sk_O})^s}{\hat{e}(PK_{j_i}, tk'_{i,u})\hat{e}(t_u, C'_i)} \\
&= \frac{\hat{e}(g_2^x, g_1^{q_{v_i}(0)+sk_O})^s \hat{e}(PK_{j_i}, tk'_{i,u}) e(h_1^{-y}, g_1^{q_{v_i}(0)+sk_O})^s \hat{e}(id_u, C'_i)^s \hat{e}(id_u^s, g_3^{q_{v_i}(0)+sk_O})}{\hat{e}(PK_{j_i}, tk'_{i,u})\hat{e}(t_u, C'_i)} \\
&= \hat{e}(g_2^x, g_1^{q_{v_i}(0)+sk_O})^s \hat{e}(h_1^{-y}, g_1^{q_{v_i}(0)+sk_O})^s \hat{e}(id_u, g_3^{q_{v_i}(0)+sk_O})^s \\
&= \hat{e}(g_2, h_1)^{s(q_{v_i}(0)+sk_O)} \hat{e}(h_1, h_2)^{-s(q_{v_i}(0)+sk_O)} \hat{e}(id_u, g_3)^{s(q_{v_i}(0)+sk_O)} \\
&= P_1^{s(q_{v_i}(0)+sk_O)} P_2^{-s(q_{v_i}(0)+sk_O)} \hat{e}(t_u, g_3)^{s(q_{v_i}(0)+sk_O)}.
\end{aligned}
$$

network can encrypt their collected data. Also, users can outsource most of the computational operations in the decryption phase to the cloud server. Moreover, our scheme offers a flexible and scalable key delegation and user revocation mechanisms. Indeed, in our scheme, there are several key authorities managing key delegation and user revocation in a distributed way. We provided the security definition for the new primitive and proved its security in the standard model and under the hardness assumption of the decisional bilinear Diffie-Hellman (DBDH) problem. Our performance and security analysis demonstrated that our proposed scheme is efficient, secure, and suitable for IoT applications.

APPENDIX

CORRECTNESS ANALYSIS

Theorem 3: The decryption phase is correct.

Proof: Consider an access tree $\mathscr{T}$, an attribute set S satisfying $\mathscr{T}$, a data user's identifier id_u, a ciphertext $CT_{\mathscr{T}} = (\mathscr{T}, C, C', C'', \{C_i\}_{v_i \in L_{\mathscr{T}}}, \{C'_i\}_{v_i \in L_{\mathscr{T}}})$ associated with the access tree, a secret-key set $\{sk_{i,u}^{(j_i)}\}_{i \in S}$ associated with S and id_u such that $sk_{i,u}^{(j_i)}$ is generated by a DA with public-key PK_{j_i}, and a token set $tk_u = \{t_u, tk_{i,u}, tk'_{i,u}\}_{i \in S}$. In the following, we prove the correctness of the decryption phase. We have the equation can be derived, as shown at the top of this page:

This proves Equation (13). Also, the correctness of Equations (14), (15) is clear. Combining Equations (10) and (15), we have:

$$M'^{s^{-1}} C'' = (P_1^{sr} P_2^{-sr})^{s^{-1}} P_2^r = P_1^r. \tag{44}$$

Therefore,

$$k = H(P_1^r) = H(M'^{s^{-1}} C''). \tag{45}$$

Comparing Equations (8) and (45), one concludes that the decryption phase is correct. ■

Theorem 4: The revocation phase is correct.

Proof: We first show that for the new parameters and update key $(\tilde{s}_{i_0}, \tilde{pk}_{i_0}, UKey_{i_0}) \leftarrow$ **UpdateParams**$(params, MSK_j, i_0)$, a re-encrypted ciphertext $\widetilde{CT}_{\mathscr{T}} \leftarrow$ **ReEnc**$(params, i_0, CT_{\mathscr{T}}, UKey_{i_0})$ is a valid ciphertext associated with the updated public parameter $\tilde{pk}_{i_0}$. Then, we show that an updated secret-key $\tilde{sk}_{i_0,u}^{(j)} \leftarrow$ **UpdateKey**$(params, id_u, i_0, sk_{i_0,u}^{(j)}, UKey_{i_0})$ is also a valid secret-key associated with the new secret parameter $\tilde{s}_{i_0}$.

We have: $\widetilde{CT}_{\mathscr{T}} = (\mathscr{T}, C, C', C'', \{C_i\}_{v_i \in L_{\mathscr{T}}}, \{C'_i\}_{v_i \in L_{\mathscr{T}} \setminus \{v_{i_0}\}} \cup \{\tilde{C}'_{i_0}\})$, where

$$
\begin{aligned}
\tilde{C}'_{i_0} &= C'_{i_0} C_{i_0}^{UKey_{i_0}} \\
&= (pk_i g_3^{-1})^{q_{v_i}(0)+sk_O} (g_1^{q_{v_i}(0)+sk_O})^{\tilde{s}_{i_0} - s_{i_0}} \\
&= (g_1^{s_{i_0}} g_3^{-1})^{q_{v_i}(0)+sk_O} (g_1^{q_{v_i}(0)+sk_O})^{\tilde{s}_{i_0} - s_{i_0}} \\
&= (g_1^{\tilde{s}_{i_0}} g_3^{-1})^{q_{v_i}(0)+sk_O} \\
&= (\tilde{pk}_i g_3^{-1})^{q_{v_i}(0)+sk_O}.
\end{aligned} \tag{46}
$$

Comparing (46) with (11), one concludes that $\widetilde{CT}_{\mathscr{T}}$ is a valid ciphertext associated with $\tilde{pk}_{i_0}$.

Also, we have:

$$\tilde{sk}_{i_0,u}^{(j)} = sk_{i_0,u}^{(j)} id_u^{UKey_{i_0}} = SK_j id_u^{s_i} id_u^{\tilde{s}_{i_0} - s_{i_0}} = SK_j id_u^{\tilde{s}_{i_0}}. \tag{47}$$

By Equation (7), we see that $\tilde{sk}_{i_0,u}^{(j)}$ is a valid secret-key associated with $\tilde{s}_{i_0}$. ■

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[2] R. Kandaswamy and D. Furlonger. *Blockchain-Based Transformation*. Accessed: May 2018. [Online]. Available: https://www.gartner.com/doc/3869696?srcId=1- 3132930191#a-1126710717

[3] *Gsma. Safety, Privacy and Security*. Accessed: Jan. 29, 2019. [Online]. Available: https://www.gsma.com/publicpolicy/resources/safetyprivacysecurity-across-mobile-ecosystem/

[4] M. Hassanalieragh, A. Page, T. Soyata, G. Sharma, M. Aktas, G. Mateos, B. Kantarci, and S. Andreescu, "Health monitoring and management using Internet-of-Things (IoT) sensing with cloud-based processing: Opportunities and challenges," in *Proc. IEEE Int. Conf. Services Comput.*, Jun. 2015, pp. 285–292.

[5] Z. Ji, I. Ganchev, M. O'droma, L. Zhao, and X. Zhang, "A cloud-based car parking middleware for IoT-based smart cities: Design and implementation," *Sensors*, vol. 14, no. 12, pp. 22372–22393, Nov. 2014.

[6] Y. Miao, R. Deng, X. Liu, K.-K. R. Choo, H. Wu, and H. Li, "Multi-authority attribute-based keyword search over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, to be published.

[7] S. Xu, G. Yang, Y. Mu, and X. Liu, "Efficient attribute-based encryption with blackbox traceability," in *Proc. Int. Conf. Provable Secur.* Cham, Switzerland: Springer, 2018, pp. 182–200.

[8] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, and H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3008–3018, Aug. 2018.

[9] X. Liu, H. Zhu, J. Ma, J. Ma, and S. Ma, "Key-policy weighted attribute based encryption for fine-grained access control," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC)*, Jun. 2014, pp. 694–699.

[10] K. Zhang, H. Li, J. Ma, and X. Liu, "Efficient large-universe multi-authority ciphertext-policy attribute-based encryption with white-box traceability," *Sci. China Inf. Sci.*, vol. 61, no. 3, 2018, Art. no. 032102.

[11] N. Attrapadung and H. Imai, "Dual-policy attribute based encryption," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Berlin, Germany: Springer, 2009, pp. 168–185.

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

[13] 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.*, 2006, pp. 89–98.

[14] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 321–334.

[15] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2011, pp. 568–588.

[16] J. Li, Y. Wang, Y. Zhang, and J. Han, "Full verifiability for outsourced decryption in attribute based encryption," *IEEE Trans. Serv. Comput.*, to be published.

[17] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 195–203.

[18] 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. Cryptograph. Techn.* Berlin, Germany: Springer, 2010, pp. 62–91.

[19] J. Li, Q. Yu, Y. Zhang, and J. Shen, "Key-policy attribute-based encryption against continual auxiliary input leakage," *Inf. Sci.*, vol. 470, pp. 175–188, Jan. 2019.

[20] S. Belguith, N. Kaaniche, M. Laurent, A. Jemai, and R. Attia, "PHOABE: Securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted IoT," *Comput. Netw.*, vol. 133, pp. 141–156, Mar. 2018.

[21] 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.

[22] H. Qian, J. Li, Y. Zhang, and J. Han, "Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation," *Int. J. Inf. Secur.*, vol. 14, no. 6, pp. 487–497, Nov. 2015.

[23] J. Li, N. Chen, and Y. Zhang, "Extended file hierarchy access control scheme with attribute based encryption in cloud computing," *IEEE Trans. Emerg. Topics Comput.*, to be published.

[24] X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the Internet of Things," *Future Gener. Comput. Syst.*, vol. 49, pp. 104–112, Aug. 2015.

[25] R. Li, C. Shen, H. He, X. Gu, Z. Xu, and C.-Z. Xu, "A lightweight secure data sharing scheme for mobile cloud computing," *IEEE Trans. Cloud Comput.*, vol. 6, no. 2, pp. 344–357, Apr. 2018.

[26] F. Guo, Y. Mu, W. Susilo, D. S. Wong, and V. Varadharajan, "CP-ABE with constant-size keys for lightweight devices," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 5, pp. 763–771, May 2014.

[27] M. Rasori, P. Perazzo, and G. Dini, "A lightweight and scalable attribute-based encryption system for smart cities," *Comput. Commun.*, vol. 149, pp. 78–89, Jan. 2020.

[28] H. Li and T. Jing, "A Lightweight fine-grained searchable encryption scheme in fog-based healthcare IoT networks," *Wireless Commun. Mobile Comput.*, vol. 2019, pp. 1–15, May 2019.

[29] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, and H. Li, "Lightweight fine-grained search over encrypted data in fog computing," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 772–785, Sep. 2019.

[30] R. Chaudhary, G. S. Aujla, S. Garg, N. Kumar, and J. J. Rodrigues, "SDN-enabled multi-attribute-based secure communication for smart grid in IIoT environment," *IEEE Trans. Ind. Informat.*, vol. 14, no. 6, pp. 2629–2640, Jun. 2018.

[31] Q. He, N. Zhang, Y. Wei, and Y. Zhang, "Lightweight attribute based encryption scheme for mobile cloud assisted cyber-physical systems," *Comput. Netw.*, vol. 140, pp. 163–173, Jul. 2018.

[32] J. Hao, C. Huang, J. Ni, H. Rong, M. Xian, and X. S. Shen, "Fine-grained data access control with attribute-hiding policy for cloud-based IoT," *Comput. Netw.*, vol. 153, pp. 1–10, Apr. 2019.

[33] Y. Yang, X. Liu, R. H. Deng, and Y. Li, "Lightweight sharable and traceable secure mobile health system," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 1, pp. 78–91, Jan. 2020.

[34] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proc. 17th ACM Conf. Comput. Commun. Secur.*, 2010, pp. 735–737.

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

[36] Q. Liu, G. Wang, and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Inf. Sci.*, vol. 258, pp. 355–370, Feb. 2014.

[37] H. Deng, Q. Wu, B. Qin, J. Domingo-Ferrer, L. Zhang, J. Liu, and W. Shi, "Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts," *Inf. Sci.*, vol. 275, pp. 370–384, Aug. 2014.

[38] Q. Huang, Y. Yang, and M. Shen, "Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts," *Future Gener. Comput. Syst.*, vol. 72, pp. 239–249, Aug. 2017.

[39] J. Wei, X. Chen, X. Huang, X. Hu, and W. Susilo, "RS-HABE: Revocable-storage and hierarchical attribute-based access scheme for the secure sharing of e-health records in public cloud," *IEEE Trans. Dependable Secure Comput.*, to be published.

[40] Y. Wang and J. Gao, "A regulation scheme based on the ciphertext-policy hierarchical attribute-based encryption in bitcoin system," *IEEE Access*, vol. 6, pp. 16267–16278, 2018.

[41] Y. Guo, J. Li, Y. Zhang, and J. Shen, "Hierarchical attribute-based encryption with continuous auxiliary inputs leakage," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 4852–4862, Dec. 2016.

[42] J. Li, Q. Yu, and Y. Zhang, "Hierarchical attribute based encryption with continuous leakage-resilience," *Inf. Sci.*, vol. 484, pp. 113–134, May 2019.

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

[44] W. Guo, J. Li, G. Chen, Y. Niu, and C. Chen, "A PSO-optimized real-time fault-tolerant task allocation algorithm in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3236–3249, Dec. 2015.

[45] H. Cheng, N. Xiong, L. T. Yang, and Y.-S. Jeong, "Distributed scheduling algorithms for channel access in TDMA wireless mesh networks," *J. Supercomput.*, vol. 45, no. 1, pp. 105–128, Jul. 2008.

[46] L.-H. Yang, Y.-M. Wang, Q. Su, Y.-G. Fu, and K.-S. Chin, "Multi-attribute search framework for optimizing extended belief rule-based systems," *Inf. Sci.*, vols. 370–371, pp. 159–183, Nov. 2016.

[47] X. Chen, A. Li, X. Zeng, W. Guo, and G. Huang, "Runtime model based approach to IoT application development," *Frontiers Comput. Sci.*, vol. 9, no. 4, pp. 540–553, Aug. 2015.

[48] H. Cheng, Z. Su, N. Xiong, and Y. Xiao, "Energy-efficient node scheduling algorithms for wireless sensor networks using Markov random field model," *Inf. Sci.*, vol. 329, pp. 461–477, Feb. 2016.

[49] W. Z. Guo, J. Y. Chen, G. L. Chen, and H. F. Zheng, "Trust dynamic task allocation algorithm with Nash equilibrium for heterogeneous wireless sensor network," *Secur. Commun. Netw.*, vol. 8, no. 10, pp. 1865–1877, Jul. 2015.
[50] Z.-Z. Liu, D.-H. Chu, C. Song, X. Xue, and B.-Y. Lu, "Social learning optimization (SLO) algorithm paradigm and its application in QoS-aware cloud service composition," *Inf. Sci.*, vol. 326, pp. 315–333, Jan. 2016.
[51] 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.*, 2010, pp. 261–270.
[52] S. Ruj, A. Nayak, and I. Stojmenovic, "DACC: Distributed access control in clouds," in *Proc. IEEE 10th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Nov. 2011, pp. 91–98.
[53] J. Hur, "Improving security and efficiency in attribute-based data sharing," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2271–2282, Oct. 2013.
[54] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "DAC-MACS: Effective data access control for multiauthority cloud storage systems," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1790–1801, Aug. 2013.
[55] S. Xu, Y. Li, R. Deng, Y. Zhang, X. Luo, and X. Liu, "Lightweight and expressive fine-grained access control for healthcare Internet-of-Things," *IEEE Trans. Cloud Comput.*, to be published.
[56] T. Jung, X.-Y. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2625–2633.
[57] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 8, pp. 1343–1354, Aug. 2013.
[58] Q. Li, J. Ma, R. Li, X. Liu, J. Xiong, and D. Chen, "Secure, efficient and revocable multi-authority access control system in cloud storage," *Comput. Secur.*, vol. 59, pp. 45–59, Jun. 2016.
[59] S. Lin, R. Zhang, H. Ma, and M. Wang, "Revisiting attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 10, pp. 2119–2130, Oct. 2015.
[60] *The Python Pairing Based Cryptography Library*. Accessed: Nov. 2017. [Online]. Available: https://github.com/debatem1/pypbc
[61] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. London, U.K.: Chapman & Hall, 2014.
[62] B. Lynn. PBC library manual 0.5. 11. Stanford University. Accessed: 2006. [Online]. Available: http://crypto.stanford.edu/pbc/manual/

MOHAMMAD ALI received the B.Sc. degree in applied mathematics from Shahed University, Tehran, Iran, in 2014, and the M.Sc. degree in applied mathematics from the Amirkabir University of Technology, Tehran, in 2016, where he is currently pursuing the Ph.D. degree with the Department of Mathematics. His fields of interests are cryptography and cloud computing.

MOHAMMAD-REZA SADEGHI received the Ph.D. degree from Carleton University, Ottawa, ON, Canada, in 2003. In 2004, he was a Postdoctoral Fellow with Carleton University. He is currently an Associate Professor with the Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran. His research interests include coding theory, lattice codes, and lattice-based cryptography. He received the Marwah Award for outstanding graduate (Ph.D. degree), in 2003. He received the Scholarship from the Fields Institute of Mathematical Studies, in 2004.

XIMENG LIU (Member, IEEE) received the B.Sc. degree in electronic engineering and the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2010 and 2015, respectively. He is currently a Full Professor with the College of Mathematics and Computer Science, Fuzhou University. He was a Research Fellow with the School of Information System, Singapore Management University, Singapore. He has published more than 100 articles on the topics of cloud security and big data security, including articles in the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON SERVICE COMPUTING, and the IEEE INTERNET OF THINGS JOURNAL. His research interests include cloud security, applied cryptography, and big data security. He is a member of ACM and CCF. He was awarded the Minjiang Scholars Distinguished Professor, Qishan Scholars in Fuzhou University, and ACM SIGSAC China Rising Star Award, in 2018. He served on Program Committee for several conferences, such as the 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, the 2017 IEEE Global Communications Conference, and the 2016 IEEE Global Communications Conference. He served as a Lead Guest Editor for *Wireless Communications and Mobile Computing*.