IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 19, 2021, accepted September 3, 2021, date of publication September 7, 2021, date of current version September 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3110824

Accountable and Revocable Large Universe Decentralized Multi-Authority Attribute-Based Encryption for Cloud-Aided IoT

KAIQING HUANG
Modern Industrial Innovation Practice Center, Dongguan Polytechnic, Dongguan 523808, China
School of Mathematical Sciences, South China Normal University, Guangzhou 510631, China
e-mail: kqhuang@m.scnu.edu.cn

This work was supported in part by the Program for Young Innovative Talents in 2019 of the Education Department of Guangdong Province under Grant 2019GKQNCX004, in part by the Scientific Research Foundation of Dongguan Polytechnic under Grant 2019a21, in part by the Special Project in Key Fields of Colleges and Universities in Guangdong Province in 2021 under Grant 2021ZDZX1146, and in part by Dongguan Science and Technology Commissioner Project under Grant 20201800500362.

ABSTRACT The data collected, stored, shared, and accessed across different platforms in the dynamic IoT is mostly confidential and privacy-sensitive. Data security and access control issues urgently need to be addressed. Multi-authority attribute-based encryption (MA-ABE) is seen as a potential solution for addressing data access control security concerns in the dynamic IoT since it allows for dynamic access control over encrypted data. However, the existing key abuse problem is severely destroying the security access control of MA-ABE. The existing accountable attribute-based encryption schemes only support small attributes (users) universe and single authority. Moreover, they do not support revocation. Some schemes are inefficient since they are constructed in the composite order bilinear group. In this article, the author proposes the first accountable and revocable large universe decentralized multi-authority attribute-based encryption scheme with outsourcing decryption based on prime order bilinear groups. The proposed scheme allows for the dynamic capacity expansion of attributes, users, and authorities. An audit mechanism is given to judge if the suspicious key was leaked by a malicious user or by authorities and to determine the identity of the leaker. The malicious user who divulges key can be punished by user-attribute revocation. The revocation mechanism is resistant to collusion attacks undertaken by revoked users and non-revoked users. Meanwhile, it satisfies the requirements of forward and backward security. The proposed scheme is static security in the random oracle model under the q-DPBDHE2 assumption. To save resources, the outsourced decryption module is optional for users with restricted resources. According to the results of the performance analysis, it is suited for large-scale cross-domain cooperation in the dynamic cloud-aided IoT.

INDEX TERMS Decentralized, multi-authority attribute-based encryption, accountability, user-attribute revocation, outsourcing decryption, collusion attack.

I. INTRODUCTION

The Internet of Things (IoT) is a new paradigm that integrates more and more physical things in the real world across different areas into communication networks by ubiquitous enabling device technologies such as near field communication (NFC) devices, RFID tags and readers, and embedded sensor and actuator nodes. It collects, analytics, stores, shares, and access data across different platforms, then provides intelligent services in the form of smart cities, smart grids, smart homes, smart transportation, smart healthcare with the help of other technologies such as cloud computing and artificial intelligence. Since the data is mostly confidential and privacy-sensitive, IoT security and access control issues urgently need to be addressed.

Encryption is an ideal way to protect data confidentiality. Attribute-based encryption (ABE) is considered an ideal technology to realize fine-grained access control on encrypted data which is introduced by Sahai and Waters [4]. In 2006, Goyal *et al.* distinguished ABE into key-policy attribute-based encryption (KP-ABE) and ciphertext-policy attribute-based encryption (CP-ABE) [5]. In KP-ABE, secret keys are

The associate editor coordinating the review of this manuscript and approving it for publication was Lo'ai A. Tawalbeh.

associated with access policies, and ciphertexts are associated with attributes, while in CP-ABE the ciphertexts are associated with access policies, and secret keys are associated with attributes. A user can decrypt the ciphertext (i.e. access data) if and only if her/his attributes (i.e. access privileges) satisfy the ciphertext access policy. Therefore, ABE is often used to solve the challenging issue in secure data storage [6].

However, in the Internet of things, data are shared and applied across different domains and organizations, which means that the attributes of users are authorized by different authorities in the same system. So multi-authority attribute-based encryption is more practical than single-authority attribute-based encryption for the Internet of things.

Moreover, the dynamic capacity expansion of attributes and users is necessary for the Internet of things. So large universe ABE is more practical than small universe ABE. In small universe ABE, all the attributes are fixed and enumerated when the system is initialized. After that, even adding only one attribute will lead to rebuilding the system and possibly re-encrypting all the data. Conversely, in the large universe ABE, any string can be used as an attribute, and the attributes do not need to be enumerated during system setting, which is more flexible and practical for the dynamic IoT.

Nevertheless, the existing key abuse problem is severely destroying the security access control of multi-authority attribute-based encryption systems. On the one hand, the user's secret key is only associated with attributes. Different users may have the same attributes, so it is impossible to determine the owner of the secret key from the attributes. Malicious users leak their secret keys without fear of being caught. Therefore, they may leak the partial or modified decryption keys to others for profits. On the other hand, any party can simply become an authority by creating the public key, so the authority is semi-trusted, even compromises or colludes with malicious authorities or users. Authority may forge user's secret key by using user's information, and uses or leaks the user's secret key for profits. Authority also may re-distribute the user's secret key to others. If unauthorized users obtain the decryption key through the above ways, they can access the data that they have no right to access, which severely damages data security.

The purpose of this study is to determine who leaked the key and punish them, so as to prevent the misuse of the key in the large universe multi-authority attribute-based encryption system for the Internet of things.

A. RELATED WORK

1) MULTI-AUTHORITY ABE

In 2007, Chase proposed the first multi-authority attribute-based encryption (MA-ABE), which supports multiple attribute authorities [7], but the central authority (CA) has the power to decrypt every ciphertext. Chase and Chow proposed the first decentralized MA-ABE which removes the trusted central authority [8]. Lewko and Waters proposed the first decentralized MA-CP-ABE system by using the dual system encryption methodology based on the composite order bilinear group [9]. In decentralized MA-ABE, a party can simply become an authority by creating the public key and issuing the user's private key according to her/his attributes [9].

2) LARGE-UNIVERSE ABE

In 2012, Lewko proposed the first large universe KP-ABE on prime order bilinear groups [10]. Rouselakis and Waters proposed two large universe attribute-based encryption constructions (CP-ABE and KP-ABE) on prime order bilinear groups [11]. However, these schemes only support single-authority. Rouselakis and Waters constructed a large-universe MA-CP-ABE scheme in prime order bilinear groups, which is more efficient than composite order bilinear groups [12]. Now, large universe MA-ABE is considered a promising technique to achieve fine-grained access control on the encrypted data in large-scale cross-domain applications for the IoT.

3) ACCOUNTABLE ABE

a: TRACEABLE SINGLE-AUTHORITY ABE

There was much work devoted to traceable single-authority ABE. White-box tracing algorithm can identify which user leaked the key. The suspicious key to be tracked is a well-formed decryption key. Li *et al.* firstly proposed two traceable ABE schemes [13]. One of the schemes is to identify the dishonest user who shares the decryption key. Another solution is to detect the misbehavior of authority. But both schemes only support the small attributes universe and AND-gate with wildcard access structure. Liu *et al.* proposed a new traceable CP-ABE system that supports policies expressed in any monotone access structures [20]. But the scheme is constructed based on composite order bilinear group and only supports the small attributes universe. Ning *et al.* adopted Shamir's (t,n) threshold scheme and proposed a practical large universe CP-ABE system supporting white-box traceability by which malicious users who leak their decryption keys could be traced [15]. But the scheme is only selective security, so they subsequently presented a fully secure traceable CP-ABE system from their proposed non-interactive commitments [16]. But the scheme is constructed based on composite order bilinear group. Cui *et al.* presented a large universe CP-ABE scheme with secure provenance by using techniques including embedding identification information in the private attribute key and generating signatures via signature of knowledge to subtly achieve user traceability [17]. The scheme also supports user revocation. In 2020, they provided a generic transformation to convert CP-ABE schemes of a certain type to key regeneration-free CP-ABE schemes [18]. A user can only delegate his/her decryption key in the same form without any modification so that any abused or pirated key can be traced back to its original owner [18]. Yan *et al.* proposed a white-box traceable CP-ABE scheme in a multi-domain environment by linkable ring signature technology which supports two-layer white-box tracing at both domains and intradomain users [19].

Black-box tracing algorithm can identify whose key(s) have been used in constructing a decryption black-box. The suspicious key in the decryption black box to be tracked is could be hidden. Liu *et al.* proposed the first black-box traceable CP-ABE which can identify a user whose key has been used in building a decryption device from multiple users [20]. The scheme is constructed based on the composite order bilinear group and supports the small attributes universe. In 2015, they further proposed a black-box traceable ABE scheme based on the prime order bilinear group. It is full security in the standard model [21]. They gave the first black-box traceable CP-ABE schemes simultaneously supports revocation and large attribute universe Liu and Wong [22]. Ning *et al.* developed a new methodology for constructing traitor tracing functionality, and present the first black-box traceable CP-ABE with short ciphertexts which are independent of the number of users [23].

However, in those schemes, the size of the decryption key and the public parameters depend on the number of system users. Xu *et al.* presented a generic construction of the black-box traceable ABE scheme from their proposed attribute-based set encryption scheme and fingerprint code [24]. The decryption key size is reduced to be irrelevant to the number of system users. Ye *et al.* proposed an unbounded KP-ABE scheme with black-box traceability. The scheme has a constant size of public parameters. But the scheme is constructed based on composite order bilinear group [25]. Liu *et al.* proposed a way of transforming (non-traceable) ABE schemes satisfying certain requirements to fully collusion-resistant black-box traceable ABE schemes, which adds only $O(\sqrt{K})$ elements to the ciphertext where K is the number of users in the system [26].

b: ACCOUNTBLE SINGLE-AUTHORITY ABE

Traceability technology cannot determine whether the user or the authority leaked the key. The audit is a combination of traceability technology and other technologies to identify the leaker of the suspicious key. Ning *et al.* proposed the first accountable authority CP-ABE with white-box traceability that provides an auditor to judge publicly whether a suspected user is guilty or is framed by the authority [27]. But the scheme is constructed based on composite order bilinear group and supports the small attributes universe. Yu *et al.* proposed an accountable CP-ABE scheme that allows any third party to publicly verify the identity embedded in a leaked decryption key, allows an auditor to publicly check whether a malicious user or the authority should be responsible for an exposed decryption key [28]. Based on a short signature of partial decryption key signed by the user, the malicious user or the authority can't deny the judgment [28]. But the scheme is constructed based on the composite order bilinear group and supports the small attributes universe. In the same year, Zhang *et al.* proposed an Accountable CP-ABE that simultaneously supports large universe and full security [29]. But the scheme is constructed based on the composite order bilinear group. Yu *et al.* improved the scheme [28] by constructing it based on prime order bilinear group [30]. But the scheme only supports the small attributes universe.

Lai *et al.* gave the first black-box traceable ABE with authority accountability which only doubles the ciphertext size of the underlying ABE scheme [31]. However, the size of the decryption key and the public parameters depend on the number of system users.

Li *et al.* constructed an accountable attribute-based encryption scheme with hiding ciphertext-policy, which guarantees the privacy of the users and traces the malicious user [32].

c: ACCOUNTBLE MULTI-AUTHORITY ABE

There was only a little work devoted to traceable multi-authority ABE and accountable multi-authority ABE. Li *et al.* proposed an MA-CP-ABE scheme, which allows tracing the identity of a misbehaving user who leaked the decryption key to others [33]. They improved their scheme to support hiding the attribute information in the ciphertext in 2018 [34]. But both of the schemes only support small attributes universe and AND-gate with wildcard access structure. Zhang *et al.* presented a large universe multi-authority CP-ABE with white-box traceability [35]. But those schemes can't judge whether a suspected decryption key is leaked by the user or is framed by the authority.

Xue *et al.* proposed the auditable MA-ABE with an auditing mechanism to detect which authority has incorrectly or maliciously performed the legitimacy verification procedure [36]. But central authority in the scheme has the right to decrypt all the ciphertext by using users' keys. Moreover, the scheme only supports the small attributes universe.

4) REVOCABLE ABE

After that Boldyreva *et al.* proposed the first revocable KP-ABE scheme [37], there was much work devoted to revocable ABE [38]. Attrapadung and Imai proposed the first directly revocable ABE scheme [39], and the first hybrid revocable ABE scheme that allows senders to select on-the-fly whether to use either direct or indirect revocation mode when encrypting [40]. Hur *et al.* proposed an access control mechanism using CP-ABE to enforce access control policies with efficient user-attribute revocation [41]. However, Li *et al.* pointed out it does not resist collusion attacks launched by revoked users and non-revoked users and presented a user collusion avoidance CP-ABE scheme with attribute revocation [42]. Yang *et al.* constructed new MA CP-ABE schemes with user-attribute revocation, i.e., it supports revoke an attribute from a user [43], [44]. Qian *et al.* proposed a multi-authority attribute-based encryption scheme with revocation [45]. To save space, readers can learn more from [38]. But these revocable ABE schemes do not support accountability.

Last but not the least, outsourcing decryption or sharing decryption can be used to reduce the overhead of decryption for users [38], [46], [47].

B. MOTIVATION AND CONTRIBUTIONS

To prevent the misuse of keys, the author revisited the existing solutions. Most of them only support tracing the users who have leaked their keys. Only some of them support audit. But some issues make them unsuitable to the dynamic Internet of things. Firstly, they only support small attributes (users) universe and single authority, which implies that the dynamic capacity expansion of attributes (users) and authorities is not supported. Moreover, they do not support revocation, which implies that the key leaker can continue to use or leak the key even after being caught. Last but not the least, some schemes are inefficient since they are constructed in the composite order bilinear group, and do not use outsourcing decryption or sharing decryption to reduce the overhead of decryption for users.

As far as the author's knowledge, there is not an accountable ABE solving all the issues simultaneously. In this article, the author designed an accountable and revocable large universe MA-ABE based on prime order bilinear groups by improving the previous scheme [38]. The proposed scheme simultaneously supports the following properties:

(1) As the same as the previous scheme, the proposed scheme supports the dynamic capacity expansion of attributes, users, and authorities. It is suitable for large-scale multi-domain collaboration in the dynamic IoT [38].

(2) The proposed scheme provides an audit mechanism to judge whether a malicious user or authorities leak the suspicious key, and to determine the identity of the leaker.

(3) The malicious user who divulges key can be punished by user-attribute revocation, i.e. the revoked user will lose one or more attributes (corresponding to the leaked key), and she/he can access the data so long as her/his remaining attributes satisfy the access policy. The revocation mechanism is secure against the collusion attack launched by revoked users and non-revoked users. Meanwhile, it meets the requirements of forward and backward security [38].

(4) The limited-resource user can choose to outsource decryption for saving resources. The performance analysis results indicate that the proposed scheme is more efficient and suitable for the IoT [38].

C. PAPER ORGANIZATION

The rest of this paper is organized as follows. Some related preliminaries are reviewed in Section II. The system model and security model are presented in Section III and Section IV respectively. The concrete construction is proposed in Section V. The security analysis of the proposed scheme is given in Section VI. The performance analysis and experimental results are shown in Section VII. Finally, the author concludes the paper in Section VIII.

II. PRELIMINARIES

A. NOTATIONS

In order to facilitate the understanding, the author explain some notations used throughout this article in Table 1.

TABLE 1. Entities.

Symbol	Description
$[n]$	The integer set $\{1, 2, ..., n\}$.
$[i, j]$	The set containing consecutive integers from i to j.
$s \overset{R}{\leftarrow} S$	The variable s is chosen uniformly at random from the set S.
PPT	Probabilistic Polynomial Time.
$\mathbb{Z}_p^*$	The set $\mathbb{Z}_p - \{0\}$.
$\mathbb{Z}_p^{m \times n}$	The set of matrices of size $m \times n$ with elements in $\mathbb{Z}_p$.
aid/uid	Attribute authority global identity/User global identity.
U/U_{AA}	The system attribute/authority universe.
GP	The system global public parameter.
H/F	The hash function maps each uid/attribute to a element of $\mathbb{G}$.
T	The function maps each attribute to the unique attribute authority who controls it.
S_{uid}	The attribute set of user uid.
UL_a	The user list of attribute a.
ASK_{aid}/APK_{aid}	The secret key/public key for authority aid.
SCK_a	The public key of attribute a for key sanity check.
$USK_{uid,a}$	The private key of attribute a for user uid.
ReK_a	The re-encrypt key of attribute a.
$UpK_{a,uid}$	The update key of attribute a for user uid.
$TK_{uid,S_{uid}}$	The transformation key of attribute-set S_{uid} for user uid.
RK_{uid}	The retrieving key for user uid.
CT	The original ciphertext generated by the data owner.
CT_{CSP}	The ciphertext generated after CSP re-encrypts CT by using the latest re-encrypt key.
CT_{uid}	The pre-decrypted ciphertext generated by user uid.
CT_{out}	The partial decrypted ciphertext generated by CSP.

B. BILINEAR PAIRINGS AND COMPLEXITY ASSUMPTION

Definition 1 (Bilinear Pairings): Let $\mathbb{G}$ and $\mathbb{G}_T$ be the cyclic multiplicative groups with prime order p. The identities of $\mathbb{G}$ and $\mathbb{G}_T$ are denoted as $1_{\mathbb{G}}$ and $1_{\mathbb{G}_T}$ respectively. We say a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear pairing if it satisfy the following properties.

(1) Bilinear. $\forall g_1, g_2 \in \mathbb{G}, \forall a, b \in \mathbb{Z}_p^*, e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.

(2) Non-degenerate. $\exists g_1, g_2 \in \mathbb{G}, s.t. e(g_1, g_2) \neq 1_{\mathbb{G}_T}$.

(3) Computable. There is an efficient algorithm to compute $e(g_1, g_2)$ for any g_1, g_2 in $\mathbb{G}$.

Definition 2 (q-Decisional Parallel Bilinear Diffie-Hellman Exponent 2 Assumption [48], q-DPBDHE2): The q-Decisional parallel bilinear Diffie-Hellman exponent 2 (q-DPBDHE2) problem is stated as follows: Let $\mathbb{G}$ and $\mathbb{G}_T$ be the bilinear groups with prime order p, and g be a generator of $\mathbb{G}$. $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map defined on $\mathbb{G}$. Pick $s, a, b_1, b_2, \ldots, b_q \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and $R \overset{R}{\leftarrow} \mathbb{G}_T$. Given $D = \mathbb{G}, p, e, g, g^s, g^{a^i}, g^{a^i b_j}, g^{s/b_j}, i \in [2q], j \in [q], i \neq q+1, g^{sa^i b_j / b_{j'}}, i \in [q+1], j \in [q], j' \in [q], j \neq j'$, and asked to distinguish $(D, e(g, g)^{sa^{q+1}})$ from (D, R).

An algorithm $\mathcal{A}$ solves the q-DPBDHE2 problem in group G with advantage ϵ if

$$|Pr[\mathcal{A}(D, e(g, g)^{sa^{q+1}}) = 0] - Pr[\mathcal{A}(D, R) = 0]| \geqslant \epsilon$$

The q-DPBDHE2 Assumption holds in $\mathbb{G}$ if no probabilistic polynomial time algorithm has a non-negligible advantage ϵ in solving the q-DPBDHE2 problem in $\mathbb{G}$.

Definition 3 (q-Strong Diffie-Hellman Assumption [49], q-SDH): The q-Strong Diffie-Hellman (q-SDH) problem is stated as follows: Let g be a generator of a cyclic group G of order p. Pick $s \overset{R}{\leftarrow} \mathbb{Z}_p^*$. Given a $q+1$-tuple $(g, g^s, g^{s^2}, \ldots, g^{s^q})$ and asked to compute a pair $(r, g^{\frac{1}{s+r}})$, where r is chosen freely from $\mathbb{Z}_p^* \setminus \{-s\}$.

An algorithm $\mathcal{A}$ solves the q-SDH problem in group G with advantage ϵ if

$$Pr[\mathcal{A}(g, g^s, g^{s^2}, \ldots, g^{s^q}) = (r, g^{\frac{1}{s+r}})] \geqslant \epsilon$$

We say that the q-SDH assumption holds in G if no probabilistic polynomial time algorithm has a non-negligible advantage ϵ in solving the q-SDH problem in G.

Definition 4 (Discrete Logarithm Problem Assumption, DLP): The Discrete Logarithm Problem (DLP) is stated as follows: Let g be a generator of a multiplication cyclic group $\mathbb{G}$ with prime order p. Pick $x \overset{R}{\leftarrow} \mathbb{Z}_p^*$. Given g^x and asked to work out x.

An algorithm $\mathcal{A}$ solves the DL problem in group G with advantage ϵ if

$$Pr[\mathcal{A}(g^x) = x] \geqslant \epsilon$$

We say that the DLP assumption holds in G if no probabilistic polynomial time algorithm has a non-negligible advantage ϵ in solving the discrete logarithm problem in G.

C. ACCESS STRUCTURES AND LINEAR SECRET-SHARING SCHEMES

Definition 5 (Access Structure [50]): Let $P = \{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^P$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection)$\mathbb{A}$ of non-empty subsets of P, i.e., $A \in 2^P \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

In attribute-based encryption scheme, the parties are replaced by the attributes. An access structure $\mathbb{A}$ will contain some authorized sets of attributes. Similarly, an access structure we mean a monotone access structure in this study.

Definition 6 (Linear Secret-Sharing Schemes (LSSS) [50]): Let p be a prime and U the attribute universe. A secret-sharing scheme π with domain of secrets $\mathbb{Z}_p$ realizing access structures on U is linear over $\mathbb{Z}_p$ if

1. The shares of a secret $s \in \mathbb{Z}_p$ for each attribute form a vector over $\mathbb{Z}_p$.

2. For each access structure $\mathbb{A}$ on U, there exists a matrix $M \in \mathbb{Z}_p^{l \times n}$, called the share-generating matrix, and a function ρ, that labels the rows of M with attributes from U, i.e., $\rho : [l] \rightarrow U$, which satisfy the following: During the generation of the shares, we consider the column vector $\vec{v} = (s, r_2, \ldots, r_n)^\top$, where $r_2, \ldots, r_n \overset{\$}{\leftarrow} \mathbb{Z}_p$. Then the vector of l shares of the secret s according to π is equal to $M\vec{v} \in \mathbb{Z}_p^{l \times 1}$. The share $(M\vec{v})_j$ where $j \in [l]$ "belongs" to attribute $\rho(j)$.

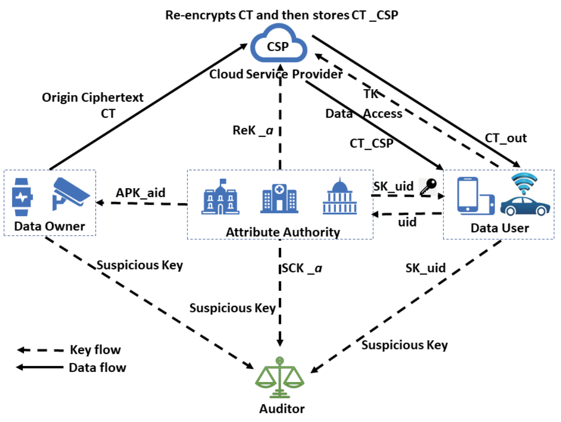


FIGURE 1. System model.

We will be referring to the pair (M, ρ) as the policy of the access structure $\mathbb{A}$.

D. ZERO-KNOWLEDGE PROOF KNOWLEDGE OF DISCRETE LOGARITHM

A zero-knowledge proof of knowledge (ZK-POK) of discrete logarithm protocol enables a prover to prove that it possesses the discrete logarithm x of a given group element y in question to a verifier.

A ZK-POK protocol has two distinct properties: the zero-knowledge property and the proof of knowledge property. The property of zero-knowledge implies that there exists a simulator S which is able to simulate the view of a verifier in the protocol without being given the witness as input. The proof of knowledge property implies there exists a knowledge-extractor which interacts with the prover and extracts the witness using rewinding techniques. The reader can get more details about ZK-POK in [51].

III. SYSTEM MODEL

As shown in Figure 1, the proposed accountable and revocable large universe decentralized multi-authority attribute-based encryption for cloud-aided IoT consists of the following entities. As the same as the other decentralized attribute-based encryption schemes [9], [12], the system is created during a trusted setup and publishes the global public parameters without secret.

Attribute Authority (AA). Any party can simply become an attribute authority (AA) by creating its correct authority public key. Each AA is independent and manages a disjoint attribute set respectively which implies that each attribute is assigned to a single AA. Each AA authenticates users' attributes and distributes the matching private keys. It also can revoke and update users' attributes by using attribute keys. For example, if using the proposed scheme in social networks, a user Bob can simply become an attribute authority by creating his authority public key and issuing some attributes (i.e. labels like "Bob's friend", "Bob's colleague", "Bob's family member", etc.). If he encrypts one photo with the access policy as "Bob's friend" and "Bob's colleague", then only the users who own the decryption key of both of attributes "Bob's friend" and "Bob's colleague" can access this photo. If Bob is a malicious user (authority) who generates a wrong public key, then no one can access the data which is encrypted with the attributes issued by Bob. However, it will not affect the normal operation of other honest users but causes wrong redundancy and waste system resources. Therefore, in practical application, some measures can be taken to reduce malicious authorities and error redundancy. E.g. the authority should be certified and issue the correct authority public key.

Cloud Service Provider (CSP). CSP is semi-trusted (honest-but-curious). It will honestly and correctly execute the tasks but be curious about the data messages it receives. As the same in [38], [44], CSP will not collude with the malicious users. It has both computational power and large storage. It re-encrypts the original ciphertext from users by using the latest re-encrypt keys and saves the re-encrypted ciphertext. It is also responsible for re-encrypting the ciphertext when a revocation happens. Furthermore, it provides outsourcing decryption for resource-limited users if they require it.

Data Owner (DO). To ensure data confidentiality and achieve flexible access to data, data owners encrypt the data file using a symmetric encryption technique which is a lightweight encryption method, then encrypt the symmetric key under the access policy.

Data User (DU). The data user obtains a set of attributes as well as corresponding decryption keys from authorities. She/He can verify whether they are legal and available or not. The data user can freely obtain any encrypted data from CSP, and decrypt the ciphertext if and only if her/his attributes satisfy the access policy. Also, data users with insufficient resources can outsource decryption to CSP.

Auditor. The auditor is trusted. Any party can submit the suspicious key found in the system to the auditor for trial. The auditor will honestly execute the audit procedure and return the audit result.

A. FRAMEWORK

This proposed scheme mainly contains the probabilistic polynomial time algorithms as follows.

GlobalSetup$(1^\lambda) \rightarrow GP$. The system is created during a trusted setup. It takes the system security parameter λ as input and outputs the global public parameters GP.

AASetup$(GP, aid) \rightarrow (APK_{aid}, ASK_{aid})$. The attribute authority setup algorithm is run by each attribute authority. It takes the global public parameters GP and the authority's identity aid as input and outputs the authority's public key APK_{aid} and secret key ASK_{aid}. The authority keeps the secret key and publishes the public key.

Encrypt$(m, (M, \rho), GP, \{APK_{aid}\}) \rightarrow CT$. The encryption algorithm is run by data owners. It takes a plaintext message m, an access structure (M, ρ), the global public parameters GP, and the involved authorities' public keys $\{APK_{aid}\}$ as input and outputs the ciphertext CT.

ReEncrypt$(CT, ReK_a) \rightarrow CT_{CSP}$. This algorithm is run by CSP. On receiving the ciphertext, CSP re-encrypts it by using the latest re-encrypt keys $\{ReK_a\}$ corresponding to the attributes set $\{a\}$ in the ciphertext. And outputs the re-encrypted ciphertext CT_{CSP}.

If there is a new attribute in the ciphertext, CSP will ask the re-encrypt key to its attribute authority $T(a)$. On receiving the require of the re-encrypt key for a new attribute a, the attribute authority $T(a)$ initializes a list of users $UL_a = \emptyset$, then chooses a random number $v_a \in \mathbb{Z}_p^*$ as the attribute key, and sets the public key $SCK_a = g^{v_a}$ and the re-encrypt key $ReK_a = v_a$. At last, it saves the public key SCK_a and sends the re-encrypt key ReK_a to CSP via secure channel.

USKGen$(uid, T(a), a \in U) \rightarrow USK_{uid,a}$. Since the attribute a is managed by the authority $T(a)$, the user key generation algorithm is run by the authority $T(a)$. If a is a new attribute in the system, the attribute authority initializes a list of users $UL_a = \emptyset$, then chooses a random number $v_a \in \mathbb{Z}_p^*$ as the attribute key, and sets the public key $SCK_a = g^{v_a}$ and the re-encrypt key $ReK_a = v_a$. At last, it saves the public key (SCK_a and sends the re-encrypt key ReK_a to CSP via secure channel. The user uid submits a tuple $(H(uid)^{\chi_{uid,a}}, ZK - POK_{uid,a})$ to the authority, where $\chi_{uid,a} \xleftarrow{R} \mathbb{Z}_p^*$, and $ZK - POK_{uid,a}$ is the zero-knowledge proof of knowledge of discrete logarithm $\chi_{uid,a}$. The authority first checks whether the proof of knowledge is valid. It aborts if it is invalid. It takes the data user's identity uid, his attribute a and the authority's secret key as input, and outputs the private key $USK_{uid,a}$ for the user uid.

The authority sends $(SCK_a, USK_{uid,a})$ to the user.

The user sets $K_{uid,a,0} = \chi_{uid,a}$, and saves the private key $USK'_{uid,a} = K_{uid,a,0} \cup USK_{uid,a}$.

Trace$(GP, USK'_{*,a}, APK_{T(a)}, SCK_a) \rightarrow ((uid, K_{*,a,2})$, **or** $\perp)$. This algorithm can be run by the auditor or the data users. It take the suspicious key $USK'_{*,a}$ of the attribute a, GP, $APK_{T(a)}$, SCK_a as input and outputs $(uid, K_{*,a,2})$, or $\perp$.

Audit$(GP, USK'_{*,a}, APK_{T(a)}, SCK_a) \rightarrow (uid$, **or** $T(a)$, **or** $\perp)$. This algorithm is run by the auditor. It take the suspicious key $USK'_{*,a}$ of the attribute a, GP, $APK_{T(a)}$, SCK_a as input and outputs $\perp$ or uid or $T(a)$.

Decrypt$(CT_{CSP}, GP, USK_{uid,S_{uid}}) \rightarrow m$. The decryption algorithm is run by the data user. It takes the ciphertext CT_{CSP}, the global public parameters GP, and the private key $USK_{uid,S_{uid}}$ as input. If the set of user's attributes $S_{uid} \models (M, \rho)$, it outputs the message m. Otherwise, it outputs $\perp$.

The data user can choose to outsource decryption if he owns limited resources or for saving resources. This feature is implemented in four algorithms.

TKGen$(USK_{uid,S_{uid}}) \rightarrow (RK_{uid}, TK_{uid,S_{uid}})$. This algorithm is run by data users uid. It chooses a random nonzero number as the retrieving key RK_{uid}, and takes the private keys $USK_{uid,S_{uid}}$ as input, then outputs the transformation key $TK_{uid,S_{uid}}$.

PreDec$(USK_{uid,S_{uid}}, CT_{CSP}, GP) \rightarrow CT_{uid}$. This algorithm is run by data users. It takes the ciphertext CT_{CSP}, the global public parameters GP, and the private keys

$USK_{uid,S_{uid}}$ as input and outputs the pre-decrypted ciphertext CT_{uid}.

PartDec$(\{TK_{uid,a}\}, CT_{uid}, GP) \rightarrow CT_{out}$. This algorithm is run by CSP. It takes the transformation key $\{TK_{uid,a}\}$ and the pre-decrypted ciphertext CT_{uid} as input, then outputs the partial decrypted ciphertext CT_{out}.

FinalDec$(CT_{out}, RK_{uid}) \rightarrow m$. This algorithm is run by data users uid. It works out the massage m by using the partial decrypted ciphertext CT_{out} and the retrieving key RK_{uid}.

Revoke$(uid, a, UL_a) \rightarrow (SCK'_a, ReK'_a, \{UpK_{a,uid'}\}_{uid' \in UL_a - \{uid\}})$. This algorithm is run by the attribute authority who manages the attribute a. It takes the attribute a, the user uid as input, then publishes the new public key SCK'_a of the attribute a, the new re-encrypt key ReK'_a for CSP to update the involved data. Meanwhile, it sends the update key $UpK_{a,uid'}$ for the non-revoked users uid' update the involved private keys.

If an attribute authority wants to revoke the attribute a from the system, it revokes the attribute a from all the users who own a from UL_a, by running the above algorithm.

If the system wants to revoke a user uid from the system, it asks all the involved attribute authorities to revoke all the attributes from the user uid, by running the above algorithm.

IV. SECURITY MODEL

A. STATIC SECURITY MODEL

In this section, the security model is similar to [12] which is named statically security. In the security game, the adversary should claim the corrupt authorities. The challenge message can be encrypted by some attributes from some of these corrupt authorities, but should at least one attribute from the honest authority, which means that the ciphertext still cannot be attacked successfully if only part of the encrypted attributes are from corrupted authorities.

The security game played between adversary $\mathcal{A}$ and challenger $\mathcal{C}$ as follows:

Global setup. The challenger $\mathcal{C}$ runs the **GlobalSetup**$(1^{\lambda}) \rightarrow GP$ algorithm to get the global public parameters GP. It sends GP to $\mathcal{A}$.

Adversary's queries. The adversary $\mathcal{A}$ issues a polynomially bounded number of queries statically:

- **Authority's public key queries**. $\mathcal{A}$ submits a set of the non-corrupt authorities $N_{AA} \subseteq U_{AA}$, and a set of the corrupt authorities $C_{AA} \subseteq U_{AA}$, where $N_{AA} \cap C_{AA} = \emptyset$.
- **User's secret key queries**. $\mathcal{A}$ submits a sequence $\{(uid_i, \{H(uid_i)^{\chi_{uid_i,a}}, ZK - POK_{uid_i,a}\}_{a \in S_i}, S_i)\}_{i \in I}$, where $S_i \subseteq U$, $T(S_i) \cap C_{AA} = \emptyset$, and $ZK - POK_{uid_i,a}$ denotes the zero-knowledge proof of knowledge of discrete logarithm $\chi_{uid_i,a}$. $(uid_i, \{H(uid_i)^{\chi_{uid_i,a}}, ZK - POK_{uid_i,a}\}_{a \in S_i}, S_i)$ denotes that the adversary requests the secret keys for the attributes set S_i of the user uid_i.
- **Transformation key queries**. $\mathcal{A}$ submits a sequence $\{(uid_j, \{H(uid_j)^{\chi_{uid_j,a}}, ZK - POK_{uid_j,a}\}_{a \in S_j}, S_j)\}_{j \in J}$, where $J \cap I = \emptyset$, $S_j \subseteq U$, $T(S_j) \cap C_{AA} = \emptyset$, and $ZK - POK_{uid_j,a}$ denotes the zero-knowledge proof of knowledge of discrete logarithm $\chi_{uid_j,a}$. $(uid_j, \{H(uid_j)^{\chi_{uid_j,a}}, ZK - POK_{uid_j,a}\}_{a \in S_j}, S_j)$ denotes that the adversary requests the transformation keys for the attributes set S_j of the user uid_j.
- **Revocation queries**. $\mathcal{A}$ submits a sequence $\{(uid_k, S_k)\}_{k \in K}$, where $S_k \subseteq U$ and $T(S_k) \cap C_{AA} = \emptyset$. A pair (uid_k, S_k) denotes that the adversary asks to revoke the attributes set S_k from the user uid_k.
- **Encryption queries**. $\mathcal{A}$ submits a challenge access structure (M, ρ), and two equal-length messages m_0, m_1. It requires that the attributes set $\bigcup_{i \in I \cup J} S_i \cup \{a \in aid\}_{aid \in C_{AA}}$ does not satisfy (M, ρ).

Challenger's replies. The $\mathcal{C}$ randomly chooses a bit $b \in 0, 1$ and replies to the queries as follows:

- **Authority's public key replies**. For each non-corrupt authority $aid \in N_{AA}$, $\mathcal{C}$ runs the algorithm **AASetup**$(GP, aid) \rightarrow (APK_{aid}, ASK_{aid})$ to get the authority's key pair, then replies to $\mathcal{A}$ with the corresponding public keys. $\mathcal{A}$ can create the public keys of the corrupt authorities by themselves.
- **User's secret key replies**. $\mathcal{C}$ runs the algorithm **USKGen**$(uid_i, T(a), a \in U) \rightarrow USK_{uid_i,a}$ to get the users' secret keys, then sends them to $\mathcal{A}$.
- **Transformation key replies**. Firstly, $\mathcal{C}$ runs the algorithm **USKGen**$(uid_j, T(a), a \in U) \rightarrow USK_{uid_j,a}$ to get the users' secret keys, then runs the algorithm **TKGen**$(\{USK_{uid_j,a}\}) \rightarrow (\{TK_{uid_j,a}\})$ to get the transformation keys and sends them to $\mathcal{A}$.
- **Revocation replies**. For each pair (uid_k, S_k), each attribute $a \in S_k$, $\mathcal{C}$ runs the algorithm **Revoke**$(uid_k, a, UL_a) \rightarrow (SCK'_a, ReK'_a, \{UpK_{a,uid'}\}_{uid' \in UL_a - \{uid_k\}})$ to get the update keys and sends them to $\mathcal{A}$, then updates all the involved data with the last re-encrypt keys $\{ReK_a\}$.
- **Encryption replies**. $\mathcal{C}$ runs the algorithm

Encrypt$(m, (M, \rho), GP, \{APK_{aid}\}) \rightarrow CT$ to encrypt m_b with (M, ρ), then runs the algorithm

ReEncrypt$(CT_{CSP}, ReK_a) \rightarrow CT_{CSP}$ to re-encrypt CT with the last re-encrypt keys $\{ReK_a\}$. At last, it sends CT_{CSP} to $\mathcal{A}$.

Guess: $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$ and wins the game if $b' = b$.

Definition 7: The proposed scheme is static security in the random oracle model if no probabilistic polynomial time adversary can break the above security game with non-negligible advantage.

B. ACCOUNTABILITY MODEL

In this section, the accountability model is defined from three aspects based on the scheme in [27]. Firstly, the identity of the user (uid) in the available decryption key (including the forged decryption key) can be traced. Secondly, it is difficult for a data user to modify or forge his legal key. Thirdly, it is difficult for an attribute authority to forge the legal decryption keys of a legitimate user.

1) TRACEABILITY

The intuition behind this game is that a dishonest data user attempts to change s/he identity (uid) embedded in the legal decryption key or forged available decryption with a new identity (uid) to avoid being caught. i.e. the identity of the user (uid) in the available decryption key (including the forged decryption key) can be traced. The traceability of the proposed scheme is described by a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows.

Setup. $\mathcal{C}$ runs the algorithm **GlobalSetup**$(1^\lambda) \rightarrow GP$ to get the global public parameters GP, then runs the algorithm **AASetup**$(GP, aid) \rightarrow (APK_{aid}, ASK_{aid})$ to get the key pairs $\{(APK_{aid}, ASK_{aid})\}_{aid \in U_{AA}}$. At last, $\mathcal{C}$ sends the global parameters GP and the public keys $\{APK_{aid}\}_{aid \in U_{AA}}$ to the adversary $\mathcal{A}$.

User's secret key queries. $\mathcal{A}$ submits a sequence $\{(uid_i, \{H(uid_i)^{\chi_{uid_i,a}}, ZK - POK_{uid_i,a}\}_{a \in S_i}, S_i)\}_{i \in I}$, where $S_i \subseteq U$, $T(S_i) \cap C_{AA} = \emptyset$, and $ZK - POK_{uid_i,a}$ denotes the zero-knowledge proof of knowledge of discrete logarithm $\chi_{uid_i,a}$. $(uid_i, \{H(uid_i)^{\chi_{uid_i,a}}, ZK - POK_{uid_i,a}\}_{a \in S_i}, S_i)$ denotes that the adversary requests the secrect keys for the attributes set S_i of the user uid_i.

$\mathcal{C}$ runs the algorithm **USKGen**$(uid_i, T(a), a \in U) \rightarrow USK_{uid_i,a}$ to get the users' secret keys, then sends $\{USK_{uid_i,S_i}\}_{i \in I}$ to $\mathcal{A}$.

The secrect keys for the attributes set S_i of the user uid_i is set as

$$USK_{uid_i,S_i} = \{K_{uid_i,1}, \langle K_{uid_i,a,2}, K_{uid_i,a,3}, K_{uid_i,a,4}, K_{uid_i,a,5}\rangle_{a \in S_i}\},$$

where $K_{uid_i,1} = uid_i$.

Key forgery. $\mathcal{A}$ outputs a decryption key USK'_* in the form of

$$\{K_{*,1}, \langle K_{*,a,0}, K_{*,a,2}, K_{*,a,3}, K_{*,a,4}, K_{*,a,5}\rangle_{a \in S_*}\}$$

to $\mathcal{C}$ and wins the game if there exist $a \in S_*$, such that the algorithm **Trace**$(GP, USK'_{*,a}, APK_{T(a)}, SCK_a) \rightarrow ((uid, K_{*,a,2}),$ **or** $\perp)$ outputs $(K_{*,1}, K_{*,a,2})$, but

$$(K_{*,1}, K_{*,a,2}) \neq (uid_i, K_{uid_i,a,2}), \quad \forall i \in I.$$

where

$$USK'_{*,a} = \langle K_{*,a,0}, K_{*,1}, K_{*,a,2}, K_{*,a,3}, K_{*,a,4}, K_{*,a,5}\rangle.$$

Definition 8: The proposed scheme is fully traceable if no probabilistic polynomial time adversary can win the above traceability game with non-negligible advantage.

2) DISHONEST USER GAME

The intuition behind this game is that a dishonest data user attempts to forge the legal decryption keys with a new secret. The new secret is different from the secret submitted to the attribute authority by the data user and embedded in the legal key. The dishonest user game is played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows.

Setup. $\mathcal{C}$ runs the algorithm **GlobalSetup**$(1^\lambda) \rightarrow GP$ to get the global public parameters GP, then runs the algorithm **AASetup**$(GP, aid) \rightarrow (APK_{aid}, ASK_{aid})$ to get the key pairs $\{(APK_{aid}, ASK_{aid})\}_{aid \in U_{AA}}$. At last, $\mathcal{C}$ sends the global parameters GP and the public keys $\{APK_{aid}\}_{aid \in U_{AA}}$ to the adversary $\mathcal{A}$.

User's secret key queries. $\mathcal{A}$ submits a sequence $\{(uid_i, \{H(uid_i)^{\chi_{uid_i,a}}, ZK - POK_{uid_i,a}\}_{a \in S_i}, S_i)\}_{i \in I}$, where $S_i \subseteq U$, $T(S_i) \cap C_{AA} = \emptyset$, and $ZK - POK_{uid_i,a}$ denotes the zero-knowledge proof of knowledge of discrete logarithm $\chi_{uid_i,a}$. $(uid_i, \{H(uid_i)^{\chi_{uid_i,a}}, ZK - POK_{uid_i,a}\}_{a \in S_i}, S_i)$ denotes that the adversary requests the secrect keys for the attributes set S_i of the user uid_i.

$\mathcal{C}$ runs the algorithm **USKGen**$(uid_i, T(a), a \in U) \rightarrow USK_{uid_i,a}$ to get the users' secret keys, then sends $\{USK_{uid_i,S_i}\}_{i \in I}$ to $\mathcal{A}$.

The secrect keys for the attributes set S_i of the user uid_i is set as

$$USK_{uid_i,S_i} = \{K_{uid_i,1}, \langle K_{uid_i,a,2}, K_{uid_i,a,3}, K_{uid_i,a,4}, K_{uid_i,a,5}\rangle_{a \in S_i}\},$$

where $K_{uid_i,1} = uid_i$.

Key forgery. $\mathcal{A}$ outputs a decryption key USK'_* in the form of

$$\{K_{*,1}, \langle K_{*,a,0}, K_{*,a,2}, K_{*,a,3}, K_{*,a,4}, K_{*,a,5}\rangle_{a \in S_*}\}$$

to $\mathcal{C}$ and wins the game if there exist $a \in S_*, i \in I$, such that the algorithm **Trace**$(GP, USK'_{*,a}, APK_{T(a)}, SCK_a) \rightarrow ((uid, K_{*,a,2}),$ **or** $\perp)$ outputs

$$(K_{*,1}, K_{*,a,2}) = (uid_i, K_{uid_i,a,2}),$$

where

$$USK'_{*,a} = \langle K_{*,a,0}, K_{*,1}, K_{*,a,2}, K_{*,a,3}, K_{*,a,4}, K_{*,a,5}\rangle.$$

But $g^{K_{*,a,0}} \neq g^{K_{uid_i,a,0}}$.

Definition 9: The proposed scheme is tamper-resistant if no probabilistic polynomial time adversary can win the above security game with non-negligible advantage.

3) DISHONEST AUTHORITY GAME

The intuition behind this game is that a dishonest attribute authority attempts to forge the legal decryption keys with the same secret which is embedded in the key by the data user. The dishonest authority game is played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows.

Setup. $\mathcal{A}$ with an identity $aid \in U_{AA}$ runs the algorithm **GlobalSetup**$(1^\lambda) \rightarrow GP$ to get the global public parameters GP, then runs the algorithm **AASetup**$(GP, aid) \rightarrow (APK_{aid}, ASK_{aid})$ to get the key pairs (APK_{aid}, ASK_{aid}). At last, $\mathcal{A}$ sends the global parameters GP and the public keys APK_{aid} to the challenger $\mathcal{C}$.

User's secret key queries. $\mathcal{C}$ with an identity uid submits $(uid, H(uid)^{\chi_{uid,a}}, ZK - POK_{uid,a})$, where $T(a) = aid$, and $ZK - POK_{uid,a}$ denotes the zero-knowledge proof of knowledge of discrete logarithm $\chi_{uid,a}$.

$\mathcal{A}$ runs the algorithm **USKGen**$(uid, T(a), a \in U) \rightarrow USK_{uid,a}$ to get the user's secret key, then sends $USK_{uid,a}$ to $\mathcal{C}$.

The secrect key for the attribute a of the user uid is in the form of

$$\langle K_{uid,a,1}, K_{uid,a,2}, K_{uid,a,3}, K_{uid,a,4}, K_{uid,a,5}\rangle.$$

Key forgery. The adversary $\mathcal{A}$ outputs a decryption key USK'_{*} in the form of

$$\langle K_{*,0}, K_{*,1}, K_{*,2}, K_{*,3}, K_{*,4}, K_{*,5}\rangle.$$

to $\mathcal{C}$ and wins the game if $K_{*,0} = K_{uid,a,0}$.

Definition 10: The proposed scheme is key-stolen-resistant if no probabilistic polynomial time adversary can win the above dishonest authority game with a non-negligible advantage.

Definition 11: The proposed scheme is fully accountable if no probabilistic polynomial time adversary can win the above three security games with non-negligible advantage.

V. CONCRETE SCHEME

In this section, the author presents the concrete construction of accountable and revocable decentralized MA-ABE for the IoT based on prime-order bilinear groups as follows.

GlobalSetup$(1^\lambda) \rightarrow GP$. The system is created during a trusted setup. It takes the system security parameter λ as input. It chooses two suitable multiplicative cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ with large prime order $p \in \Theta\{2^\lambda\}$. Let g be a generator of $\mathbb{G}$, and defines a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ on $\mathbb{G}$. The attribute universe is $U = \mathbb{Z}_p$. U_{AA} denotes the set of all attribute authorities. Additionally, it chooses two hash functions H and F maps user identities and attributes to elements of $\mathbb{G}$ respectively. The function T maps each attribute to the unique attribute authority who controls it. Finally, the algorithm outputs the global public parameters $GP = \langle p, g, \mathbb{G}, \mathbb{G}_T, e, U, U_{AA}, H, F, T\rangle$.

AASetup$(GP, aid) \rightarrow (APK_{aid}, ASK_{aid})$. This algorithm is run by attribute authorities. For each authority $aid \in U_{AA}$, it chooses $\alpha_{aid}, \beta_{aid}, \gamma_{aid}, \eta_{aid} \in \mathbb{Z}_p^*$ as its secret key ASK_{aid}, and publishes the public key

$$APK_{aid} = \langle e(g,g)^{\alpha_{aid}}, g^{\beta_{aid}}, g^{\gamma_{aid}}, g^{\eta_{aid}}\rangle.$$

Encrypt$(m, (M,\rho), GP, \{APK_{aid}\}) \rightarrow CT$. This algorithm is run by data owners. It takes a plaintext message m, an access structure (M,ρ), and a set of authority public keys $\{APK_{aid}\}$ as input, where $M \in \mathbb{Z}_p^{l\times n}$ and ρ is a map from each row $\vec{M}_i$ of M to an attribute $\rho(i) \in U$. Let δ be a function maps each row $\vec{M}_i$ to the authority who manages attribute $\rho(i)$. i.e., $\delta(i) = T(\rho(i))$. For encryption, the algorithm randomly picks numbers $s, v_2, \ldots, v_n, w_2, \ldots, w_n \in Z_p^*$. Let $\vec{v} = (s, v_2, \cdots, v_n)^\top$ and $\vec{w} = (0, w_2, \cdots, w_n)^\top$. For $i = 1, \cdots, l$, it computes $\lambda_i = \vec{M}_i\vec{v}$, and $w_i = \vec{M}_i\vec{w}$. The algorithm picks randomly numbers $r_i \in Z_p^*$, and computes:

$$C_0 = me(g,g)^s, \quad C_{1,i} = e(g,g)^{\lambda_i} e(g,g)^{\alpha_{\delta(i)} r_i},$$
$$C_{2,i} = g^{-r_i}, \quad C_{3,i} = g^{\beta_{\delta(i)} r_i} g^{w_i}, \quad C_{4,i} = F(\rho(i))^{r_i},$$
$$C_{5,i} = g^{-\gamma_{\delta(i)} r_i}, \quad C_{6,i} = g^{-\eta_{\delta(i)} r_i}.$$

At last, the original ciphertext is

$$CT = \{(M,\rho), C_0, \{C_{1,i}, C_{2,i}, C_{3,i}, C_{4,i}, C_{5,i}, C_{6,i}\}_{i=1}^{l}\}.$$

Finally, the ciphertext is sent to CSP.

ReEncrypt$(CT, ReK_a) \rightarrow CT_{CSP}$. This algorithm is run by CSP. On receiving the ciphertext, CSP re-encrypts it by using the latest re-encrypt keys $\{ReK_a\}$ corresponding to the attributes set $\{a\}$ in the ciphertext. If there is a new attribute in the ciphertext, CSP will ask the re-encrypt key to its attribute authority $T(a)$. On receiving the require of the re-encrypt key for a new attribute a, the attribute authority $T(a)$ initializes a list of users $UL_a = \emptyset$, then chooses a random number $v_a \in \mathbb{Z}_p^*$ as the attribute key, then sets the public key $SCK_a = g^{v_a}$ and the re-encrypt key $ReK_a = v_a$. At last, it keeps SCK_a sends the re-encrypt key ReK_a to CSP by secure way.

The algorithm takes the original ciphertext CT and the lastest re-encrypt keys $\{ReK_a\}$ as input, and computes:

$$C'_{3,i} = C_{3,i} C_{2,i}^{-ReK_{\rho(i)}} = g^{\beta_{\delta(i)} r_i} g^{v_{\rho(i)} r_i} g^{w_i}.$$

At last, the re-encrypted ciphertext is

$$CT_{CSP} = \{(M,\rho), C_0, \{C_{1,i}, C_{2,i}, C'_{3,i}, C_{4,i}, C_{5,i}, C_{6,i}\}_{i=1}^{l}\}.$$

USKGen$(uid, T(a), a \in U) \rightarrow USK_{uid,a}$. Since the attribute a is managed by the authority $T(a)$, then this algorithm is run by the authority $T(a)$. If a is a new attribute in the system, the attribute authority $T(a)$ initializes a list of users $UL_a = \emptyset$, then chooses a random number $v_a \in \mathbb{Z}_p^*$ as the attribute key, and sets the public key $SCK_a = g^{v_a}$ and the re-encrypt key $ReK_a = v_a$. At last, it saves the public key (SCK_a and sends the re-encrypt key ReK_a to CSP via secure channel. To generate the user's private key, the user uid randomly chooses $\chi_{uid,a} \in \mathbb{Z}_p^*$ as the secret key, and computes the commitment value $H(uid)^{\chi_{uid,a}}$. She/He sends $H(uid)^{\chi_{uid,a}}$ and the zero-knowledge proof of knowledge of discrete logarithm $\chi_{uid_i,a}$ to the authority $T(a)$. The authority first checks whether the proof of knowledge is valid. It aborts if it is invalid. Otherwise, it chooses two random numbers $t_a \in \mathbb{Z}_p^*$ and $y_a \in \mathbb{Z}_p \setminus \{-\frac{\gamma_{T(a)}+uid}{\eta_{T(a)}}\}$, i.e.

$$\gamma_{T(a)} + uid + \eta_{T(a)} y_a \neq 0 (mod p).$$

Then it saves $(uid, y_a, H(uid)^{\chi_{uid,a}})$ into the list of users UL_a, and sets

$$K_{uid,a,1} = uid, \quad K_{uid,a,2} = y_a, \quad K_{uid,a,3} = g^{t_a},$$
$$K_{uid,a,4} = g^{(\gamma_{T(a)}+\eta_{T(a)} y_a) t_a},$$

and

$$K_{uid,a,5} = g^{\frac{\alpha_{T(a)}}{\gamma_{T(a)}+uid+\eta_{T(a)}y_a}} H(uid)^{\frac{\beta_{T(a)}+v_a}{\gamma_{T(a)}+uid+\eta_{T(a)}y_a}} \cdot F(a)^{t_a} (H(uid)^{\chi_{uid,a}})^{\frac{1}{\gamma_{T(a)}+uid+\eta_{T(a)}y_a}},$$
$$= g^{\frac{\alpha_{T(a)}}{\gamma_{T(a)}+uid+\eta_{T(a)}y_a}} H(uid)^{\frac{\beta_{T(a)}+v_a+\chi_{uid,a}}{\gamma_{T(a)}+uid+\eta_{T(a)}y_a}} F(a)^{t_a}.$$

Finally, the private key of the attribute a for the data user uid is

$$USK_{uid,a} = \langle K_{uid,a,1}, K_{uid,a,2}, K_{uid,a,3}, K_{uid,a,4}, K_{uid,a,5}\rangle.$$

The authority sends $(SCK_a, USK_{uid,a})$ to the user.

The user sets $K_{uid,a,0} = \chi_{uid,a}$, and saves the private key

$$USK'_{uid,a} = \langle K_{uid,a,0}, K_{uid,a,1}, K_{uid,a,2}, K_{uid,a,3}, K_{uid,a,4}, K_{uid,a,5}\rangle.$$

Remarks: on receiving each user the private key $USK_{uid,a}$, the user uid can verify whether it is legal and available or not as follows.

Firstly, s/he checks whether $K_{uid,a,1} = uid$ or not. S/He rejects the key if $K_{uid,a,1} \neq uid$. Otherwise, s/he runs the algorithm **Trace**$(GP, USK'_{uid,a}, APK_{T(a)}, SCK_a) \rightarrow ((uid, K_{uid,a,2})$, **or** $\perp)$. S/He rejects the key if it outputs $\perp$. Otherwise, s/he accepts the key.

The user also can save only one uid for all the $USK'_{uid,a}$. i.e. the user saves the private keys as $USK_{uid,S_{uid}} = \{K_{uid,1}, \langle K_{uid,a,0}, K_{uid,a,2}, K_{uid,a,3}, K_{uid,a,4}, K_{uid,a,5}\rangle_{a\in S_{uid}}\}$, where $K_{uid,1} = uid$ and S_{uid} is the user's attribute set.

Trace$(GP, USK'_{*,a}, APK_{T(a)}, SCK_a) \rightarrow ((uid, K_{*,a,2})$, **or** $\perp)$. This algorithm can be run by the auditor or the data users. It takes the suspicious key $USK'_{*,a}$ of the attribute a, GP, $APK_{T(a)}$, and SCK_a as input. If $USK'_{*,a}$ is not in the form of

$$\langle K_{*,a,0}, K_{*,a,1}, K_{*,a,2}, K_{*,a,3}, K_{*,a,4}, K_{*,a,5}\rangle,$$

it outputs $\perp$. Otherwise, it runs a key sanity check on $USK'_{*,a}$ as follows.

$$K_{*,a,0}, K_{*,a,1}, K_{*,a,2} \in \mathbb{Z}_p, \quad K_{*,a,3}, K_{*,a,4}, K_{*,a,5} \in G, \tag{5.1}$$

$$e(g, K_{*,a,4}) = e(K_{*,a,3}, g^{\gamma_{T(a)}} g^{\eta_{T(a)} K_{*,a,2}}), \tag{5.2}$$

$$\begin{aligned} e(K_{*,a,5}, g^{\gamma_{T(a)}} g^{K_{*,a,1}} g^{\eta_{T(a)} K_{*,a,2}}) &= e(g,g)^{\alpha_{T(a)}} \\ &\cdot e(H(K_{*,a,1}), g^{\beta_{T(a)}} SCK_a g^{K_{*,a,0}}) \\ &\cdot e(F(a), K_{*,a,3}^{K_{*,a,1}} K_{*,a,4}). \end{aligned} \tag{5.3}$$

We say $USK'_{*,a}$ passes the key sanity check if the Eqs. (1)-(3) hold for the attribute a. If $USK'_{*,a}$ passes the key sanity check, the algorithm outputs $uid = K_{*,a,1}$ and $K_{*,a,2}$. Otherwise, it outputs $\perp$.

Audit$(GP, USK'_{*,a}, APK_{T(a)}, SCK_a) \rightarrow (uid$, **or** $T(a)$, **or** $\perp)$. This algorithm can be run by the auditor. It takes the suspicious key $USK'_{*,a}$ of the attribute a, GP, $APK_{T(a)}$, and SCK_a as input. Firstly, it run the algorithm **Trace**$(GP, USK'_{*,a}, APK_{T(a)}, SCK_a) \rightarrow ((uid, K_{*,a,2})$, **or** $\perp)$. It aborts and outputs $\perp$ if the **Trace**-algorithm outputs $\perp$.

Otherwise, the **Trace**-algorithm outputs $uid = K_{*,a,1}$, then the auditor makes the following judgment.

If uid exist, the data user is asked to submit his/her private key $USK_{uid,a} = \langle g^{K_{uid,a,0}}, K_{uid,a,1}, K_{uid,a,2}, K_{uid,a,3}, K_{uid,a,4}, K_{uid,a,5}\rangle_{a\in S_{uid}}$, where $g^{K_{uid,a,0}}$ instead of $K_{uid,a,0}$ to keep $K_{uid,a,0}$ in a secret state. If $USK_{uid,a}$ does not pass the key sanity check, it outputs "uid" which indicates that the user with uid is dishonest. If $USK_{uid,a}$ passes the key sanity check, and $g^{K_{*,a,0}} = g^{K_{uid,a,0}}$, it outputs "$uid$" which indicates that the users with uid is dishonest. Otherwise, it outputs "$T(a)$" which indicates that the attribute authority $T(a)$ is dishonest.

Decrypt$(CT_{CSP}, GP, USK_{uid,S_{uid}}) \rightarrow m$. This algorithm is run by data users. Suppose a user uid with a set of unrevoked attributes S_{uid} wants to decrypt the ciphertext CT_{CSP}. If $S_{uid} \not\models (M, \rho)$, this algorithm outputs $\perp$. Otherwise, it exist a subset $\{\rho(i) : i \in I \subset [l]\}$ of S_{uid} satisfy the access policy (M, ρ). Then it calculates constants $\{c_i : i \in I\}$ such that $\sum_{i\in I} c_i \vec{M}_i = (1, 0, \cdots, 0)$. Then it computes: for all $i \in I$,

$$\begin{aligned} D_i &= C_{1,i} e(K_{uid,\rho(i),5}, C_{2,i}^{K_{uid,1}} C_{5,i} C_{6,i}^{K_{uid,\rho(i),2}}) \\ &\quad \cdot e(H(K_{uid,1}), C'_{3,i} C_{2,i}^{-K_{uid,\rho(i),0}}) \\ &\quad \cdot e(K_{uid,\rho(i),3}^{K_{uid,1}} K_{uid,\rho(i),4}, C_{4,i}), \\ \prod_{i\in I} D_i^{c_i} &= e(g,g)^s, \\ \frac{C_0}{e(g,g)^s} &= \frac{me(g,g)^s}{e(g,g)^s} = m. \end{aligned}$$

Outsourcing Decryption. The data user can choose to outsource decryption if he owns limited resources or for saving resources. This feature is implemented in four algorithms.

TKGen$(USK_{uid,S_{uid}}) \rightarrow (RK_{uid}, TK_{uid,S_{uid}})$. This algorithm is run by data users. Assuming that a user uid with a set of unrevoked attributes S_{uid} wants to decrypt the ciphertext CT_{CSP}. Firstly, it chooses a random numbers $z \in Z_p^*$ as the retrieving key, i.e., $RK_{uid} = z$, then computes:

$$K^*_{uid,a,4} = K_{uid,a,3}^{K_{uid,1}} K_{uid,a,4}, \quad K^*_{uid,a,5} = K_{uid,a,5}^{z},$$

Then it sets the transformation key $TK_{uid,S_{uid}} =$

$$\{K_{uid,1}, \langle K^*_{uid,a,4}, K^*_{uid,a,5}\rangle_{a\in S_{uid}}\}.$$

PreDec$(USK_{uid,S_{uid}}, CT_{CSP}, GP) \rightarrow CT_{uid}$. This algorithm is run by data users. Assuming that a user uid with a set of unrevoked attributes S_{uid} wants to decrypt the ciphertext CT_{CSP}. Firstly, it pre-decryptes the ciphertext by computing:

$$C^*_{3,i} = C'_{3,i} C_{2,i}^{-K_{uid,\rho(i),0}}, \quad C^*_{6,i} = C_{6,i}^{K_{uid,\rho(i),2}}.$$

Then it set the pre-decrypted ciphertext as $CT_{uid} = \{(M, \rho), C_0, \{C_{1,i}, C_{2,i}, C^*_{3,i}, C_{4,i}, C_{5,i}, C^*_{6,i}\}_{i=1}^{l}\}$.

At last, it sends $TK_{uid,S_{uid}}$ and CT_{uid} to CSP.

PartDec$(TK_{uid,S_{uid}}, CT_{uid}, GP) \rightarrow CT_{out}$. This algorithm is run by CSP. Assuming that the subset $\{\rho(i) : i \in I \subset [l]\}$ of S_{uid} satisfies the access policy (M, ρ). It calculates constants $\{c_i : i \in I\}$ such that $\sum_{i\in I} c_i \vec{M}_i = (1, 0, \cdots, 0)$, then it computes:

$$CT_1 = \prod_{i\in I} (C_{1,i} e(H(K_{uid,1}), C^*_{3,i}) e(K^*_{uid,\rho(i),4}, C_{4,i}))^{c_i},$$

and

$$CT_2 = \prod_{i\in I} (e(K^*_{uid,\rho(i),5}, C_{2,i}^{K_{uid,1}} C_{5,i} C^*_{6,i}))^{c_i}.$$

Finally, it sets $CT_{out} = (CT_1, CT_2)$ and sends it to the data user.

FinalDec$(CT_{out}, RK_{uid}) \rightarrow m$. This algorithm is run by data users. It computes:

$$\frac{C_0}{CT_1 CT_2^{\frac{1}{z}}} = \frac{me(g,g)^s}{e(g,g)^s} = m.$$

Revoke$(uid, a, UL_a) \rightarrow (SCK'_a, ReK'_a, \{UpK_{a,uid'}\}_{uid' \in UL_a - \{uid\}})$. If an attribute a is revoked from the data user uid, the authority $T(a)$ deletes $(uid, y_a, H(uid)^{\chi_{uid,a}})$ from the list UL_a, and chooses a new attribute key $v'_a \xleftarrow{R} Z_p^*$. Then the authority calculates the new $SCK'_a = g^{v'_a}$, the new re-encrypt key $ReK'_a = v'_a$, and the update key $UpK_{a,uid'} = H(uid')^{\frac{v'_a - v_a}{\gamma_{T(a)}+uid'+\eta_{T(a)}y_a}}$ for the non-revoked user uid' in UL_a.

At last, the authority sends SCK'_a to auditor, ReK'_a to CSP, and $(SCK'_a, UpK_{a,uid'})$ to the non-revoked user uid' respectively.

After receiving the update key, the non-revoked user uid' updates the private keys by calculating:

$$\begin{aligned} K'_{uid',a,5} &= K_{uid',a,5} UpK_{a,uid'} \\ &= g^{\frac{\alpha_{T(a)}}{\gamma_{T(a)}+uid'+\eta_{T(a)}y_a}} H(uid')^{\frac{\beta_{T(a)}+v'_a+\chi_{uid,a}}{\gamma_{T(a)}+uid'+\eta_{T(a)}y_a}} F(a)^{t_a}. \end{aligned}$$

After receiving the new re-encrypt key, CSP updates the involved data by running the algorithm **ReEncrypt**$(CT_{CSP}, ReK'_a - ReK_a) \rightarrow CT_{CSP}$.

$$\begin{aligned} C''_{3,i} &= C'_{3,i}(C_{2,i})^{-(ReK'_a - ReK_a)} = C'_{3,i}(C_{2,i})^{v_a - v'_a} \\ &= g^{\beta_{\delta(i)}r_i} g^{v_{\rho(i)}r_i} g^{w_i} g^{(v'_a - v_a)r_i} = g^{\beta_{\delta(i)}r_i} g^{v'_{\rho(i)}r_i} g^{w_i}, \end{aligned}$$

where $a = \rho(i)$.

If an attribute authority wants to revoke the attribute a from the system, it revokes the attribute a from all the users who own a in UL_a, by running the above algorithm.

If the system wants to revoke a user uid from the system, it asks all the involved attribute authorities to revoke all the attributes from the user uid, by running the above algorithm.

Correctness.

$$\begin{aligned} D_i &= C_{1,i} e(K_{uid,\rho(i),5}, C_{2,i}^{K_{uid,1}} C_{5,i} C_{6,i}^{K_{uid,\rho(i),2}}) \\ &\quad \cdot e(H(K_{uid,1}), C'_{3,i} C_{2,i}^{-K_{uid,\rho(i),0}}) \\ &\quad \cdot e(K_{uid,\rho(i),3}^{K_{uid,1}} K_{uid,\rho(i),4}, C_{4,i}) \\ &= e(g,g)^{\lambda_i} e(g,g)^{\alpha_{\delta(i)}r_i} \\ &\quad \cdot e(g^{\frac{\alpha_{\delta(i)}}{\gamma_{\delta(i)}+uid+\eta_{\delta(i)}y_{\rho(i)}}} H(uid)^{\frac{\beta_{\delta(i)}+v_{\rho(i)}+\chi_{uid,\rho(i)}}{\gamma_{\delta(i)}+uid+\eta_{\delta(i)}y_{\rho(i)}}} F(\rho(i))^{t_{\rho(i)}}, \\ &\quad g^{-r_i uid} g^{-\gamma_{\delta(i)}r_i} g^{-\eta_{\delta(i)}r_i y_{\rho(i)}}) \\ &\quad \cdot e(H(uid), g^{\beta_{\delta(i)}r_i} g^{v_{\rho(i)}r_i} g^{w_i} g^{\chi_{uid,\rho(i)}r_i}) \\ &\quad \cdot e(g^{t_{\rho(i)}uid} g^{(\gamma_{\delta(i)}+\eta_{\delta(i)}y_{\rho(i)})t_{\rho(i)}}, F(\rho(i))^{r_i}) \\ &= e(g,g)^{\lambda_i} e(g,g)^{\alpha_{\delta(i)}r_i} \\ &\quad \cdot e(g^{\frac{\alpha_{\delta(i)}}{\gamma_{\delta(i)}+uid+\eta_{\delta(i)}y_{\rho(i)}}} H(uid)^{\frac{\beta_{\delta(i)}+v_{\rho(i)}+\chi_{uid,\rho(i)}}{\gamma_{\delta(i)}+uid+\eta_{\delta(i)}y_{\rho(i)}}} F(\rho(i))^{t_{\rho(i)}}, \\ &\quad g^{-r_i(\gamma_{\delta(i)}+uid+\eta_{\delta(i)}y_{\rho(i)})}) \\ &\quad \cdot e(H(uid), g^{(\beta_{\delta(i)}+v_{\rho(i)}+\chi_{uid,\rho(i)})r_i} g^{w_i}) \\ &\quad \cdot e(g^{(\gamma_{\delta(i)}+uid+\eta_{\delta(i)}y_{\rho(i)})t_{\rho(i)}}, F(\rho(i))^{r_i}) \\ &= e(g,g)^{\lambda_i} e(g,g)^{\alpha_{\delta(i)}r_i} e(g,g)^{-\alpha_{\delta(i)}r_i} \\ &\quad \cdot e(H(uid)^{\beta_{\delta(i)}+v_{\rho(i)}+\chi_{uid,\rho(i)}}, g^{-r_i}) \\ &\quad \cdot e(F(\rho(i))^{t_{\rho(i)}}, g^{-r_i(\gamma_{\delta(i)}+uid+\eta_{\delta(i)}y_{\rho(i)})}) \\ &\quad \cdot e(H(uid), g^{(\beta_{\delta(i)}+v_{\rho(i)}+\chi_{uid,\rho(i)})r_i} g^{w_i}) \\ &\quad \cdot e(g^{(\gamma_{\delta(i)}+uid+\eta_{\delta(i)}y_{\rho(i)})t_{\rho(i)}}, F(\rho(i))^{r_i}) \\ &= e(g,g)^{\lambda_i} e(H(uid), g)^{w_i}, \end{aligned}$$

If $S_{uid} \models (M, \rho)$, it exist a subset $\{\rho(i) : i \in I \subset [l]\}$ of S_{uid} satisfy the access policy (M, ρ). Then it can calculate constants $\{c_i : i \in I\}$ such that $\sum_{i \in I} \vec{c}_i M_i = (1, 0, \cdots, 0)$. Then, it has

$$\sum_{i \in I} \lambda_i c_i = \sum_{i \in I} \vec{v} \vec{M}_i c_i = \vec{v}(1, 0, \cdots, 0) = s,$$

and

$$\sum_{i \in I} w_i c_i = \sum_{i \in I} \vec{w} \vec{M}_i c_i = \vec{w}(1, 0, \cdots, 0) = 0.$$

Therefore,

$$\begin{aligned} \prod_{i \in I} D_i^{c_i} &= \prod_{i \in I} (e(g,g)^{\lambda_i} e(H(uid), g)^{w_i})^{c_i} \\ &= \prod_{i \in I} e(g,g)^{\lambda_i c_i} e(H(uid), g)^{w_i c_i} \\ &= e(g,g)^{\sum_{i \in I} \lambda_i c_i} e(H(uid), g)^{\sum_{i \in I} w_i c_i} \\ &= e(g,g)^s, \end{aligned}$$

Hence, it has $\frac{C_0}{e(g,g)^s} = \frac{me(g,g)^s}{e(g,g)^s} = m$.

In the outsourcing decryption mode,

$$\begin{aligned} & e(H(K_{uid,1}), C^*_{3,i}) e(K^*_{uid,\rho(i),4}, C_{4,i}) \\ &= e(H(uid), g^{\beta_{\delta(i)}r_i} g^{v_{\rho(i)}r_i} g^{w_i} g^{\chi_{uid,\rho(i)}r_i}) \\ &\quad \cdot e(g^{t_{\rho(i)}uid} g^{(\gamma_{\delta(i)}+\eta_{\delta(i)}y_{\rho(i)})t_{\rho(i)}}, F(\rho(i))^{r_i}) \\ &= e(H(uid), g^{(\beta_{\delta(i)}+v_{\rho(i)}+\chi_{uid,\rho(i)})r_i} g^{w_i}) \\ &\quad \cdot e(g^{(uid+\gamma_{\delta(i)}+\eta_{\delta(i)}y_{\rho(i)})t_{\rho(i)}}, F(\rho(i))^{r_i}) \\ &= e(H(uid), g)^{w_i} \cdot e(H(uid), g)^{(\beta_{\delta(i)}+v_{\rho(i)}+\chi_{uid,\rho(i)})r_i} \\ &\quad \cdot e(g, F(\rho(i)))^{(uid+\gamma_{\delta(i)}+\eta_{\delta(i)}y_{\rho(i)})t_{\rho(i)}r_i}, \end{aligned}$$

$$\begin{aligned} CT_1 &= \prod_{i \in I} (C_{1,i} e(H(K_{uid,1}), C^*_{3,i}) e(K^*_{uid,\rho(i),4}, C_{4,i}))^{c_i}, \\ &= \prod_{i \in I} e(g,g)^{\lambda_i c_i} e(H(uid), g)^{w_i c_i} e(g,g)^{\alpha_{\delta(i)}r_i c_i} \\ &\quad \cdot e(H(uid), g)^{(\beta_{\delta(i)}+v_{\rho(i)}+\chi_{uid,\rho(i)})r_i c_i} \\ &\quad \cdot e(g, F(\rho(i)))^{(uid+\gamma_{\delta(i)}+\eta_{\delta(i)}y_{\rho(i)})t_{\rho(i)}r_i c_i} \\ &= e(g,g)^s \\ &\quad \cdot \prod_{i \in I} e(g,g)^{\alpha_{\delta(i)}r_i c_i} e(H(uid), g)^{(\beta_{\delta(i)}+v_{\rho(i)}+\chi_{uid,\rho(i)})r_i c_i} \\ &\quad \cdot e(g, F(\rho(i)))^{(uid+\gamma_{\delta(i)}+\eta_{\delta(i)}y_{\rho(i)})t_{\rho(i)}r_i c_i}, \end{aligned}$$

$$\begin{aligned} & e(K^*_{uid,\rho(i),5}, C_{2,i}^{K_{uid,1}} C_{5,i} C^*_{6,i}) \\ &= e(g^{\frac{z\alpha_{\delta(i)}}{\gamma_{\delta(i)}+uid+\eta_{\delta(i)}y_{\rho(i)}}} H(uid)^{\frac{z(\beta_{\delta(i)}+v_{\rho(i)}+\chi_{uid,\rho(i)})}{\gamma_{\delta(i)}+uid+\eta_{\delta(i)}y_{\rho(i)}}} \\ &\quad \cdot F(\rho(i))^{zt_{\rho(i)}}, g^{-r_i uid} g^{-\gamma_{\delta(i)}r_i} g^{-\eta_{\delta(i)}r_i y_{\rho(i)}}) \\ &= e(g^{\frac{z\alpha_{\delta(i)}}{\gamma_{\delta(i)}+uid+\eta_{\delta(i)}y_{\rho(i)}}} H(uid)^{\frac{z(\beta_{\delta(i)}+v_{\rho(i)}+\chi_{uid,\rho(i)})}{\gamma_{\delta(i)}+uid+\eta_{\delta(i)}y_{\rho(i)}}} \\ &\quad \cdot F(\rho(i))^{zt_{\rho(i)}}, g^{-r_i(\gamma_{\delta(i)}+uid+\eta_{\delta(i)}y_{\rho(i)})}) \\ &= e(g,g)^{-z\alpha_{\delta(i)}r_i} e(H(uid), g)^{-z(\beta_{\delta(i)}+v_{\rho(i)}+\chi_{uid,\rho(i)})r_i} \\ &\quad \cdot e(g, F(\rho(i)))^{-z(uid+\gamma_{\delta(i)}+\eta_{\delta(i)}y_{\rho(i)})t_{\rho(i)}r_i} \end{aligned}$$

$$
\begin{aligned}
CT_2 &= \prod_{i\in I}(e(K^*_{uid,\rho(i),5}, C_{2,i}^{K_{uid,1}}C_{5,i}C^*_{6,i}))^{c_i}\\
&= \prod_{i\in I} e(g,g)^{-z\alpha_{\delta(i)}r_ic_i}e(H(uid),g)^{-z(\beta_{\delta(i)}+v_{\rho(i)}+\chi_{uid,\rho(i)})r_ic_i}\\
&\quad\cdot e(g,F(\rho(i)))^{-z(uid+\gamma_{\delta(i)}+\eta_{\delta(i)}y_{\rho(i)})t_{\rho(i)}r_ic_i}.
\end{aligned}
$$

Hence,

$$
\frac{C_0}{CT_1CT_2^{\frac{1}{z}}} = \frac{me(g,g)^s}{e(g,g)^s} = m.
$$

VI. SECURITY ANALYSIS

A. STATIC SECURITY

Then Theorem 1 proves that the proposed scheme is statically secure as the RW15 scheme [12].

Theorem 1: The proposed concrete scheme is statically secure in the random oracle model under the $q-DPBDHE2$ assumption.

Proof: Suppose that there exists a PPT adversary $\mathcal{A}$ who can break the proposed scheme with non-negligible advantage ε, then the author can build a simulator $\mathcal{B}$ to break the RW15 scheme [12] with the same advantages. Denote the challenger of the RW15 scheme by $\mathcal{C}$.

System Setup: $\mathcal{B}$ gets the global parameters $GP = \langle p, g, \mathbb{G}, \mathbb{G}_T, e, U, U_{AA}, T, F, H\rangle$ from $\mathcal{C}$, then passes them to the adversary $\mathcal{A}$, where the random oracles H and F are programmed by $\mathcal{C}$.

Adversary's queries: The adversary $\mathcal{B}$ issues a polynomially bounded number of queries statically:

- **Authority's public key queries:** $\mathcal{A}$ submits a set of the non-corrupt authorities $N_{AA} \subseteq U_{AA}$, and a set of the corrupt authorities $C_{AA} \subseteq U_{AA}$, where $N_{AA}\cap C_{AA} = \emptyset$, and $\mathcal{A}$ creates the public keys of the corrupt authorities by himself.
- **User's secret key queries:** $\mathcal{A}$ submits a sequence $\{(uid_i, \{H(uid_i)^{\chi_{uid_i,a}}, ZK-POK_{uid_i,a}\}_{a\in S_i}, S_i)\}_{i\in I}$, where $S_i \subseteq U$, $T(S_i)\cap C_{AA} = \emptyset$, and $ZK-POK_{uid_i,a}$ denotes the zero-knowledge proof of knowledge of discrete logarithm $\chi_{uid_i,a}$. $(uid_i, \{H(uid_i)^{\chi_{uid_i,a}}, ZK-POK_{uid_i,a}\}_{a\in S_i}, S_i)$ denotes that the adversary requests the secrect keys for the attributes set S_i of the user uid_i.
- **Transformation key queries:** $\mathcal{A}$ submits a sequence $\{(uid_j, \{H(uid_j)^{\chi_{uid_j,a}}, ZK-POK_{uid_j,a}\}_{a\in S_j}, S_j)\}_{j\in J}$, where $J\cap I = \emptyset$, $S_j \subseteq U$, $T(S_j)\cap C_{AA} = \emptyset$, and $ZK-POK_{uid_j,a}$ denotes the zero-knowledge proof of knowledge of discrete logarithm $\chi_{uid_j,a}$. $(uid_j, \{H(uid_j)^{\chi_{uid_j,a}}, ZK-POK_{uid_j,a}\}_{a\in S_j}, S_j)$ denotes that the adversary requests the transformation keys for the attributes set S_j of the user uid_j.
- **Revocation queries:** $\mathcal{A}$ submits a sequence $\{(uid_k, S_k)\}_{k\in K}$, where $S_k \subseteq U$ and $T(S_k)\cap C_{AA} = \emptyset$. A pair (uid_k, S_k) denotes that the adversary asks to revoke the attributes set S_k from the user uid_k.
- **Encryption queries:** $\mathcal{A}$ submits a challenge access structure (M,ρ), and two equal-length messages m_0, m_1. We require that the attributes set $\bigcup_{i\in I\cup J} S_i \cup \{a\in aid\}_{aid\in C_{AA}}$ does not satisfy (M,ρ).

Challenger's replies: After receiving the adversary's queries, the simulator $\mathcal{B}$ sends C_{AA}, N_{AA}, $\{(uid_i, S_i)\}_{i\in I\cup J}$, (M,ρ), m_0, m_1 to $\mathcal{C}$ for request the corresponding public keys, secret keys, ciphertext. $\mathcal{C}$ returnes the public keys

$$
\{APK_{aid} = \langle e(g,g)^{\alpha_{aid}}, g^{\beta_{aid}}\rangle\}_{aid\in N_{AA}},
$$

the secret keys

$$
\{\langle g^{\alpha_{T(a)}}H(uid_i)^{\beta_{T(a)}}F(a)^{t_a}, g^{t_a}\rangle_{a\in S_i}\}_{i\in I\cup J},
$$

and the challenge ciphertext CT as follows:

$$
\begin{aligned}
&C_0 = m_be(g,g)^s, \quad C_{1,i} = e(g,g)^{\lambda_i}e(g,g)^{\alpha_{\delta(i)}r_i},\\
&C_{2,i} = g^{-r_i}, \quad C_{3,i} = g^{\beta_{\delta(i)}r_i}g^{w_i}, \quad C_{4,i} = F(\rho(i))^{r_i},
\end{aligned}
$$

where $i = 1,\cdots,l$. Then $\mathcal{B}$ replies the queries as follows:

- **Authority's public key replies:** For each authority $aid \in N_{AA}$, $\mathcal{B}$ randomly chooses $\gamma_{aid}, \eta_{aid} \in \mathbb{Z}_p^*$, and sets the public key as

$$
APK'_{aid} = \langle e(g,g)^{\alpha_{aid}}, g^{\beta_{aid}}, g^{\gamma_{aid}}, g^{\eta_{aid}}\rangle.
$$

For each attribute $a \in aid$, $\mathcal{B}$ initializes a list of users $UL_a = \emptyset$, then chooses a random number $v_a \in \mathbb{Z}_p^*$ as the attribute key.

- **User's secret key replies:** For each pair $(uid_i, \{H(uid_i)^{\chi_{uid_i,a}}, ZK-POK_{uid_i,a}\}_{a\in S_i}, S_i)$, and each attribute $a\in S_i$, $\mathcal{B}$ picks randomly $y_a \in \mathbb{Z}_p^* \setminus \{-\frac{\gamma_{T(a)}+uid_i}{\eta_{T(a)}}\}$, i.e.

$$
\gamma_{T(a)} + uid_i + \eta_{T(a)}y_a \neq 0(mod p),
$$

and implicitly sets $t'_a = \frac{t_a}{\gamma_{T(a)}+uid_i+\eta_{T(a)}y_a}$. And $\mathcal{B}$ computes the secret key of each attribute $a\in S_i$.

$$
\begin{aligned}
K_{uid_i,a,1} &= uid_i, K_{uid_i,a,2} = y_a,\\
K_{uid_i,a,3} &= (g^{t_a})^{\frac{1}{\gamma_{T(a)}+uid_i+\eta_{T(a)}y_a}} = g^{t'_a},\\
K_{uid_i,a,4} &= K_{uid_i,a,3}^{\gamma_{T(a)}+\eta_{T(a)}y_a} = g^{(\gamma_{T(a)}+\eta_{T(a)}y_a)t'_a},\\
K_{uid_i,a,5} &= (g^{\alpha_{T(a)}}H(uid_i)^{\beta_{T(a)}}F(a)^{t_a}H(uid_i)^{v_a})^{\frac{1}{\gamma_{T(a)}+uid_i+\eta_{T(a)}y_a}}\\
&\quad\cdot(H(uid_i)^{\chi_{uid_i,a}})^{\frac{1}{\gamma_{T(a)}+uid_i+\eta_{T(a)}y_a}}\\
&= g^{\frac{\alpha_{T(a)}}{\gamma_{T(a)}+uid_i+\eta_{T(a)}y_a}}H(uid_i)^{\frac{\beta_{T(a)}+v_a+\chi_{uid_i,a}}{\gamma_{T(a)}+uid_i+\eta_{T(a)}y_a}}F(a)^{t'_a}.
\end{aligned}
$$

At last, $\mathcal{B}$ sets the secret key as

$$
\begin{aligned}
&USK_{uid_i,S_i}\\
&\quad= \langle K_{uid_i,a,1}, K_{uid_i,a,2}, K_{uid_i,a,3}, K_{uid_i,a,4}, K_{uid_i,a,5}\rangle_{a\in S_i}.
\end{aligned}
$$

- **Transformation key replies:** For each pair $(uid_j, \{H(uid_j)^{\chi_{uid_j,a}}, ZK-POK_{uid_j,a}\}_{a\in S_j}, S_j)$, $\mathcal{B}$ generates the secret keys as the same as it in user's secret key replies:

$$
\begin{aligned}
&USK_{uid_j,S_j}\\
&\quad= \{K_{uid_j,1}, \langle K_{uid_j,a,2}, K_{uid_j,a,3}, K_{uid_j,a,4}, K_{uid_j,a,5}\rangle_{a\in S_j}\},
\end{aligned}
$$

where $K_{uid_j,1} = uid_j$.

Then $\mathcal{B}$ picks randomly $z_j \in \mathbb{Z}_p$, and computes:

$$
K^*_{uid_j,a,4} = K_{uid_j,a,3}^{K_{uid_j,1}}K_{uid_j,a,4}, \quad K^*_{uid_j,a,5} = K_{uid_j,a,5}^{z_j},
$$

Then it sets the transformation key $TK_{uid_j,S_j} =$

$$\{K_{uid_j,1}, \langle K^*_{uid_j,a,4}, K^*_{uid_j,a,5}\rangle_{a\in S_j}\}.$$

• **Revocation replies:** For each pair (uid_k, S_k), each attribute $a \in S_k$, $\mathcal{B}$ deletes $(uid_k, y_a, H(uid_k)^{\chi_{uid_k,a}})$ from the list UL_a, and chooses a new attribute key $v'_a \xleftarrow{R} Z^*_p$. Then it calculates the new public key $SCK'_a = g^{v'_a}$, the re-encrypt key $ReK'_a = v'_a$, and the update key $UpK_{a,uid'} = H(uid')^{\frac{v'_a - v_a}{\gamma_{T(a)}+uid'+\eta_{T(a)}y_a}}$, where uid' is the non-revoked user. At last, it updates the involved data by running the algorithm **ReEncrypt**$(CT_{CSP}, ReK'_a - ReK_a) \to CT_{CSP}$.

• **Encryption replies:** For $i = 1, \cdots, l$, $\mathcal{B}$ computes

$$C'_{3,i} = C_{3,i}C_{2,i}^{-v_{\rho(i)}} = g^{\beta_{\delta(i)}r_i}g^{w_i}g^{v_{\rho(i)}r_i},$$
$$C_{5,i} = C_{2,i}^{\gamma_{\delta(i)}} = g^{-\gamma_{\delta(i)}r_i}, \quad C_{6,i} = C_{2,i}^{\eta_{\delta(i)}} = g^{-\eta_{\delta(i)}r_i},$$

then sets the challenge ciphertext as

$$CT' = (C_0, \{C_{1,i}, C_{2,i}, C'_{3,i}, C_{4,i}, C_{5,i}, C_{6,i}\}_{i=1}^{l}).$$

Guess: $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$, then $\mathcal{B}$ sends b' to $\mathcal{C}$.

If $\mathcal{A}$ has advantage $Adv_{\mathcal{A}}(\lambda) = \epsilon$ in breaking the concrete scheme, then $\mathcal{B}$ can break the RW15 scheme [12] with the same advantage $Adv_{\mathcal{A}}(\lambda) = \epsilon$. As shown in [12], the RW15 scheme is statically secure in the random oracle model under the $q - DPBDHE2$ assumption, so is the proposed scheme.■

B. ACCOUNTABILITY

Theorem 2 proves that the identity of the user (uid) in the available decryption key (including the forged decryption key) can be traced.

Lemma 1 [49]: If the q-SDH assumption holds in G, then the Boneh-Boyen full signature scheme is secure against strong existential forgery under an adaptive chosen message attack.

Lemma 2: If the Boneh-Boyen full signature scheme [49] is secure against strong existential forgery under an adaptive chosen message attack, then the proposed scheme is fully traceable.

Proof: Suppose that there exists a PPT adversary $\mathcal{A}$ wins the traceability game with a non-negligible advantage ε, then the author can build a simulator $\mathcal{B}$ has the same advantages to break the Boneh-Boyen full signature scheme (BB scheme, [49]) under an adaptive chosen message attack. Denote the challenger of the BB scheme by $\mathcal{C}$.

Since the attribute authority does not know $\chi_{uid,a}$ in $H(uid)^{\chi_{uid,a}}$, the proposed scheme is difficult to be proved fully traceable. Such situation occurs in the schemes [52], the simulator used a knowledge extractor to get $\chi_{uid,a}$ in their security proof. In the proof, it assumes that the simulator knows $\chi_{uid,a}$ by using the same technology.

Setup. Let U_{AA} be the authority universe. For each $aid \in U_{AA}$, Sig_{aid} is a BB scheme in the prime order group $\mathbb{G}$ with the public key $\{p, \mathbb{G}, g, g^{\gamma_{aid}}, g^{\eta_{aid}}\}$. The challenger $\mathcal{C}$ sends the public keys $\{p, \mathbb{G}, g, g^{\gamma_{aid}}, g^{\eta_{aid}}\}_{aid\in U_{AA}}$ to the simulator $\mathcal{B}$. $\mathcal{B}$ chooses a suitable multiplicative cyclic groups $\mathbb{G}_T$ with the prime order p, and defines a bilinear map $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$ on $\mathbb{G}$. The attribute universe is $U = \mathbb{Z}_p$. For each $aid \in U_{AA}$, $\mathcal{B}$ randomly picks $\alpha_{aid}, \beta_{aid} \in \mathbb{Z}^*_p$, and sets the public key as $APK_{aid} = \{e(g, g)^{\alpha_{aid}}, g^{\beta_{aid}}, g^{\gamma_{aid}}, g^{\eta_{aid}}\}$. $\mathcal{B}$ sends the global parameters $GP = \langle p, g, \mathbb{G}, \mathbb{G}_T, e, U, U_{AA}, T, F, H\rangle$ and the public keys $\{APK_{aid}\}_{aid\in U_{AA}}$ to the adversary $\mathcal{A}$, where the random oracles F and H are programmed by $\mathcal{B}$.

$\mathcal{B}$ initializes two empty tables T_1, T_2 and answers the random oracles as follows:

(1) Random oracle hash $H(uid)$: If there is an entry $(uid, h_{uid}, g^{h_{uid}})$ in T_1, then $\mathcal{B}$ outputs $g^{h_{uid}}$. Otherwise, $\mathcal{B}$ randomly chooses $h_{uid} \in \mathbb{Z}^*_p$, then outputs $g^{h_{uid}}$ and saves $(uid, h_{uid}, g^{h_{uid}})$ in T_1.

(2) Random oracle hash $F(a)$: If there is an entry (a, f_a, g^{f_a}) in T_2, then $\mathcal{B}$ outputs g^{f_a}. Otherwise, $\mathcal{B}$ randomly chooses $f_a \in \mathbb{Z}^*_p$, then outputs g^{f_a} and saves (a, f_a, g^{f_a}) in T_2.

User's secret key queries. $\mathcal{A}$ submits a sequence $\{(uid_i, \{H(uid_i)^{\chi_{uid_i,a}}, ZK - POK_{uid_i,a}\}_{a\in S_i}, S_i)\}_{i\in I}$, where $S_i \subseteq U$, $T(S_i) \cap C_{AA} = \emptyset$, and $ZK - POK_{uid_i,a}$ denotes the zero-knowledge proof of knowledge of discrete logarithm $\chi_{uid_i,a}$. $(uid_i, \{H(uid_i)^{\chi_{uid_i,a}}, ZK - POK_{uid_i,a}\}_{a\in S_i}, S_i)$ denotes that the adversary requests the secrect keys for the attributes set S_i of the user uid_i.

For each $(uid_i, \{H(uid_i)^{\chi_{uid_i,a}}, ZK - POK_{uid_i,a}\}_{a\in S_i}, S_i)$, $\mathcal{B}$ extracts $\{\chi_{uid_i,a}\}$ as anaylsis at the begining.

For each $a \in S_i$, if there is no a list of users for the attribute a, $\mathcal{B}$ initializes a list of users $UL_a = \emptyset$, then chooses a random number $v_a \in \mathbb{Z}^*_p$ as the attribute key. $\mathcal{B}$ submits $(uid_i, T(a))$ to $\mathcal{C}$ and obtains the corresponding signature $(g^{\frac{1}{\gamma_{T(a)}+uid_i+\eta_{T(a)}y_a}}, y_a)$, where y_a is a random value in $\mathbb{Z}^*_p \setminus \{-\frac{\gamma_{T(a)}+uid_i}{\eta_{T(a)}}\}$, i.e. $\gamma_{T(a)} + uid_i + \eta_{T(a)}y_a \neq 0(mod p)$. $\mathcal{B}$ chooses a random numbers $t_a \in \mathbb{Z}^*_p$ and computes

$$K_{uid_i,a,1} = uid_i, \quad K_{uid_i,a,2} = y_a, \quad K_{uid_i,a,3} = g^{t_a},$$
$$K_{uid_i,a,4} = (g^{\gamma_{T(a)}})^{t_a}(g^{\eta_{T(a)}})^{y_a t_a} = g^{(\gamma_{T(a)}+\eta_{T(a)}y_a)t_a},$$

and

$$\begin{aligned} K_{uid_i,a,5} &= (g^{\frac{1}{\gamma_{T(a)}+uid_i+\eta_{T(a)}y_a}})^{\alpha_{T(a)}+(\beta_{T(a)}+v_a+\chi_{uid_i,a})h_{uid_i}}g^{f_a t_a} \\ &= g^{\frac{\alpha_{T(a)}}{\gamma_{T(a)}+uid_i+\eta_{T(a)}y_a}}g^{h_{uid_i}\frac{\beta_{T(a)}+v_a+\chi_{uid_i,a}}{\gamma_{T(a)}+uid_i+\eta_{T(a)}y_a}}g^{f_a t_a} \\ &= g^{\frac{\alpha_{T(a)}}{\gamma_{T(a)}+uid_i+\eta_{T(a)}y_a}}H(uid_i)^{\frac{\beta_{T(a)}+v_a+\chi_{uid_i,a}}{\gamma_{T(a)}+uid_i+\eta_{T(a)}y_a}}F(a)^{t_a}. \end{aligned}$$

The secrect keys for the attributes set S_i of the user uid_i is set as

$$USK_{uid_i,S_i} = \{uid_i, \langle K_{uid_i,a,2}, K_{uid_i,a,3}, K_{uid_i,a,4}, K_{uid_i,a,5}\rangle_{a\in S_i}\}.$$

Finally, $\mathcal{B}$ sends $\{USK_{uid_i,S_i}\}_{i\in I}$ to $\mathcal{A}$.

Key forgery. $\mathcal{A}$ outputs a decryption key USK_* to $\mathcal{B}$. Since $\mathcal{A}$ wins the traceability game with a non-negligible advantage ε, the decryption key USK_* is in the form of

$USK_* = \{K_{*,1}, \langle K_{*,a,0}, K_{*,a,2}, K_{*,a,3}, K_{*,a,4}, K_{*,a,5}\rangle_{a\in S_*}\}$. Moreover, there exist $a \in S_*$, such that the algorithm **Trace**$(GP, USK'_{*,a}, APK_{T(a)}, SCK_a) \to ((uid, K_{*,a,2})$, **or** $\perp)$ outputs $(K_{*,1}, K_{*,a,2})$, but

$$(K_{*,1}, K_{*,a,2}) \neq (uid_i, K_{uid_i,a,2}), \quad \forall i \in I.$$

Hence,

$$K_{*,a,0}, K_{*,1}, K_{*,a,2} \in \mathbb{Z}_p, \quad K_{*,a,3}, K_{*,a,4}, K_{*,a,5} \in G, \tag{6.4}$$

$$e(g, K_{*,a,4}) = e(K_{*,a,3}, g^{\gamma_{T(a)}} g^{\eta_{T(a)}K_{*,a,2}}), \tag{6.5}$$

$$\begin{aligned} e(K_{*,a,5}, g^{\gamma_{T(a)}} g^{K_{*,1}} g^{\eta_{T(a)}K_{*,a,2}}) &= e(g,g)^{\alpha_{T(a)}} \cdot e(H(K_{*,1}), g^{\beta_{T(a)}} \\ &\quad \cdot SCK_a g^{K_{*,a,0}}) e(F(a), K_{*,a,3}^{K_{*,1}} K_{*,a,4}). \end{aligned} \tag{6.6}$$

Without loss of generality, assume that $\mathcal{A}$ has made the random oracle hash $H(K_{*,1})$ and $F(a)$ before outputting USK_* which passes the key sanity check. Hence, $\mathcal{B}$ obtains that $H(K_{*,1}) = g^{h_{K_{*,1}}}$ and $F(a) = g^{f_a}$ by extract the records from T_1 and T_2. Since $K_{*,a,3} \in G$, $\mathcal{B}$ can assume that $K_{*,a,3} = g^{t_3}$ where $t_3 \in \mathbb{Z}_p^*$ is unknown. So, according to (6.5), we have

$$\begin{aligned} e(g, K_{*,a,4}) &= e(K_{*,a,3}, g^{\gamma_{T(a)}} g^{\eta_{T(a)}K_{*,a,2}}) \\ &= e(g^{t_3}, g^{\gamma_{T(a)}+\eta_{T(a)}K_{*,a,2}}) \\ &= e(g, g^{(\gamma_{T(a)}+\eta_{T(a)}K_{*,a,2})t_3}). \end{aligned}$$

It implies that $K_{*,a,4} = g^{(\gamma_{T(a)}+\eta_{T(a)}K_{*,a,2})t_3}$.

According to (6.6), we have

$$\begin{aligned} & e(K_{*,a,5}, g^{\gamma_{T(a)}} g^{K_{*,1}} g^{\eta_{T(a)}K_{*,a,2}}) \\ = \ & e(g,g)^{\alpha_{T(a)}} \\ & \cdot e(H(K_{*,1}), g^{\beta_{T(a)}} SCK_a g^{K_{*,a,0}}) e(F(a), K_{*,a,3}^{K_{*,1}} K_{*,a,4}) \\ \Rightarrow \ & e(K_{*,a,5}, g^{\gamma_{T(a)}+K_{*,1}+\eta_{T(a)}K_{*,a,2}}) = e(g,g)^{\alpha_{T(a)}} \\ & \cdot e(g^{h_{K_{*,1}}}, g^{\beta_{T(a)}} g^{v_a} g^{K_{*,a,0}}) e(g^{f_a}, g^{t_3 K_{*,1}} g^{(\gamma_{T(a)}+\eta_{T(a)}K_{*,a,2})t_3}) \\ = \ & e(g,g)^{\alpha_{T(a)}+(\beta_{T(a)}+v_a+K_{*,a,0})h_{K_{*,1}}} \\ & \cdot e(g^{f_a t_3}, g^{K_{*,1}+\gamma_{T(a)}+\eta_{T(a)}K_{*,a,2}}) \\ = \ & e(g^{\frac{\alpha_{T(a)}+(\beta_{T(a)}+v_a+K_{*,a,0})h_{K_{*,1}}}{K_{*,1}+\gamma_{T(a)}+\eta_{T(a)}K_{*,a,2}}} g^{f_a t_3}, g^{K_{*,1}+\gamma_{T(a)}+\eta_{T(a)}K_{*,a,2}}). \\ \Rightarrow \ & K_{*,a,5} = g^{\frac{\alpha_{T(a)}+(\beta_{T(a)}+v_a+K_{*,a,0})h_{K_{*,1}}}{K_{*,1}+\gamma_{T(a)}+\eta_{T(a)}K_{*,a,2}}} g^{f_a t_3} \\ = \ & g^{\frac{\alpha_{T(a)}+(\beta_{T(a)}+v_a+K_{*,a,0})h_{K_{*,1}}}{K_{*,1}+\gamma_{T(a)}+\eta_{T(a)}K_{*,a,2}}} K_{*,a,3}^{f_a}, \end{aligned}$$

Then $\mathcal{B}$ computes

$$\begin{aligned} \sigma_{T(a)} &= \left(\frac{K_{*,a,5}}{K_{*,a,3}^{f_a}}\right)^{\frac{1}{\alpha_{T(a)}+(\beta_{T(a)}+v_a+K_{*,a,0})h_{K_{*,1}}}} \\ &= g^{\frac{1}{K_{*,1}+\gamma_{T(a)}+\eta_{T(a)}K_{*,a,2}}}. \end{aligned}$$

Since $K_{*,a,0}, K_{*,1}, K_{*,a,2} \in \mathbb{Z}_p$, $(\sigma_{T(a)}, K_{*,a,2})$ is a valid signature on message $K_{*,1}$ in the BB scheme $Sig_{T(a)}$, but

$$(K_{*,1}, K_{*,a,2}) \neq (uid_i, K_{uid_i,a,2}), \quad \forall i \in I.$$

where $K_{uid_i,a,2} = y_a$.

Therefore, $(\sigma_{T(a)}, K_{*,a,2})$ is different from $(g^{\frac{1}{\gamma_{T(a)}+uid_i+\eta_{T(a)}y_a}}, y_a)$ which implies that $\mathcal{B}$ breaks the Boneh-Boyen full signature scheme with the same advantage ε.■

Theorem 2: If the q-SDH assumption holds in G, then the proposed scheme is fully traceable.

Proof: It follows directly from Lemma 1 and Lemma 2.■

Theorem 3 proves that no dishonest data user can forge the legal decryption keys.

Theorem 3: If the DLP assumption holds in G, then No PPT adversary can win the dishonest user game with non-negligible advantage.

Proof: Suppose that there exists a PPT adversary $\mathcal{A}$ who can win the dishonest user game with a non-negligible advantage ε, then we can prove that $\mathcal{A}$ break the DLP assumption with the same advantages. Denote the challenger of the propose scheme by $\mathcal{C}$.

Setup. $\mathcal{C}$ runs the algorithm **GlobalSetup**$(1^\lambda) \to GP$ to get the global public parameters GP, then runs the algorithm **AASetup**$(GP, aid) \to (APK_{aid}, ASK_{aid})$ to get the key pairs $\{(APK_{aid}, ASK_{aid})\}_{aid\in U_{AA}}$. At last, $\mathcal{C}$ sends the global parameters GP and the public keys $\{APK_{aid}\}_{aid\in U_{AA}}$ to the adversary $\mathcal{A}$.

User's secret key queries. $\mathcal{A}$ submits a sequence $\{(uid_i, \{H(uid_i)^{\chi_{uid_i,a}}, ZK - POK_{uid_i,a}\}_{a\in S_i}, S_i)\}_{i\in I}$, where $S_i \subseteq U$, $T(S_i) \cap C_{AA} = \emptyset$, and $ZK - POK_{uid_i,a}$ denotes the zero-knowledge proof of knowledge of discrete logarithm $\chi_{uid_i,a}$. $(uid_i, \{H(uid_i)^{\chi_{uid_i,a}}, ZK - POK_{uid_i,a}\}_{a\in S_i}, S_i)$ denotes that the adversary requests the secrect keys for the attributes set S_i of the user uid_i.

$\mathcal{C}$ runs the algorithm **USKGen**$(uid_i, T(a), a \in U) \to USK_{uid_i,a}$ to get the users' secret keys, then sends $\{USK_{uid_i,S_i}\}_{i\in I}$ to $\mathcal{A}$.

The secrect keys for the attributes set S_i of the user uid_i is set as

$$\begin{aligned} & USK_{uid_i,S_i} \\ & \quad = \{K_{uid_i,1}, \langle K_{uid_i,a,2}, K_{uid_i,a,3}, K_{uid_i,a,4}, K_{uid_i,a,5}\rangle_{a\in S_i}\}, \end{aligned}$$

where $K_{uid_i,1} = uid_i$.

Key forgery. $\mathcal{A}$ outputs a decryption key USK'_* in the form of

$$\{K_{*,1}, \langle K_{*,a,0}, K_{*,a,2}, K_{*,a,3}, K_{*,a,4}, K_{*,a,5}\rangle_{a\in S_*}\}$$

to $\mathcal{C}$ and wins the game a non-negligible advantage ε. Hence, there exist $a \in S_*, i \in I$, such that the algorithm **Trace**$(GP, USK'_{*,a}, APK_{T(a)}, SCK_a) \to ((uid, K_{*,a,2})$, **or** $\perp)$ outputs

$$(K_{*,1}, K_{*,a,2}) = (uid_i, K_{uid_i,a,2}),$$

where

$$USK'_{*,a} = \langle K_{*,a,0}, K_{*,1}, K_{*,a,2}, K_{*,a,3}, K_{*,a,4}, K_{*,a,5}\rangle.$$

But $g^{K_{*,a,0}} \neq g^{K_{uid_i,a,0}}$ which implies that

$$H(uid_i)^{K_{*,a,0}} \neq H(uid_i)^{K_{uid_i,a,0}}.$$

Therefore,

$$K_{*,a,0} \in \mathbb{Z}_p, \quad K_{*,a,3}, K_{*,a,4}, K_{*,a,5} \in G, \tag{6.7}$$

$$e(g, K_{*,a,4}) = e(K_{*,a,3}, g^{\gamma_{T(a)}} g^{\eta_{T(a)} K_{uid_i,a,2}}), \tag{6.8}$$

$$\begin{aligned} &e(K_{*,a,5}, g^{\gamma_{T(a)}} g^{uid_i} g^{\eta_{T(a)} K_{uid_i,a,2}}) \\ &\quad = e(g,g)^{\alpha_{T(a)}} \cdot e(H(uid_i), g^{\beta_{T(a)}} SCK_a g^{K_{*,a,0}}) \\ &\quad\quad \cdot e(F(a), K_{*,a,3}^{uid_i} K_{*,a,4}). \end{aligned} \tag{6.9}$$

Since $K_{*,a,3} \in G$, $\mathcal{B}$ can assume that $K_{*,a,3} = g^{t_3}$ where $t_3 \in \mathbb{Z}_p^*$ is unknown. So, according to (6.8), we have

$$\begin{aligned} e(g, K_{*,a,4}) &= e(K_{*,a,3}, g^{\gamma_{T(a)}} g^{\eta_{T(a)} K_{uid_i,a,2}}) \\ &= e(g^{t_3}, g^{\gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2}}) \\ &= e(g, g^{(\gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2}) t_3}) \end{aligned}$$

It implies that $K_{*,a,4} = g^{(\gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2}) t_3}$.

According to (6.9), we have

$$\begin{aligned} &e(K_{*,a,5}, g^{\gamma_{T(a)} + uid_i + \eta_{T(a)} K_{uid_i,a,2}}) \\ &= e(g,g)^{\alpha_{T(a)}} \cdot e(H(uid_i), g^{\beta_{T(a)}} g^{v_a} g^{K_{*,a,0}}) \\ &\quad \cdot e(F(a), g^{t_3 uid_i} g^{(\gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2}) t_3}) \\ &\Rightarrow e(K_{*,a,5}, g^{\gamma_{T(a)} + uid_i + \eta_{T(a)} K_{uid_i,a,2}}) \\ &= e(g^{\alpha_{T(a)}}, g) \cdot e(H(uid_i)^{\beta_{T(a)} + v_a + K_{*,a,0}}, g) \\ &\quad \cdot e(F(a)^{t_3}, g^{uid_i + \gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2}}) \\ &\Rightarrow e(K_{*,a,5}, g^{\gamma_{T(a)} + uid_i + \eta_{T(a)} K_{uid_i,a,2}}) \\ &= e(g^{\frac{\alpha_{T(a)}}{uid_i + \gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2}}}, g^{uid_i + \gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2}}) \\ &\quad \cdot e(H(uid_i)^{\frac{\beta_{T(a)} + v_a + K_{*,a,0}}{uid_i + \gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2}}}, g^{uid_i + \gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2}}) \\ &\quad \cdot e(F(a)^{t_3}, g^{uid_i + \gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2}}) \\ &\Rightarrow K_{*,a,5} \\ &= g^{\frac{\alpha_{T(a)}}{uid_i + \gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2}}} H(uid_i)^{\frac{\beta_{T(a)} + v_a + K_{*,a,0}}{uid_i + \gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2}}} F(a)^{t_3}. \end{aligned}$$

Since $\mathcal{A}$ cannot break the hash function $H(uid)$ and $F(a)$, we can assume that $K_{*,a,5} = K_{uid_i,a,5} H(uid_i)^{d_1} F(a)^{d_2}$, where d_1 and d_2 are known by $\mathcal{A}$. It implies that

$$\frac{K_{*,a,0} - \chi_{uid_i,a,0}}{uid_i + \gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2}} = d_1 \neq 0.$$

So $\mathcal{A}$ can compute

$$\gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2} = \frac{K_{*,a,0} - \chi_{uid_i,a,0}}{d_1} - uid_i.$$

Hence, we can conclude that $\mathcal{A}$ has solved the DL problem $(K_{uid_i,a,3}, K_{uid_i,a,4}) = (g^{t_a}, g^{t_a(\gamma_{T(a)} + \eta_{T(a)} K_{uid_i,a,2})})$. However, solving discrete logarithm problem is hard in G. Therefore, No PPT adversary can win the dishonest user game with non-negligible advantage. ■

Theorem 4 proves that no attribute authority can forge the legal decryption keys of legitimate users.

Theorem 4: If the DLP assumption holds in G, then No PPT adversary can win the dishonest authority game with non-negligible advantage.

Proof: Suppose that there exists a PPT adversary $\mathcal{A}$ who can win the dishonest authority game with a non-negligible advantage ε, then we can build a simulator $\mathcal{B}$ can break the DLP assumption with the same advantages. Denote the challenger in the DLP assumption by $\mathcal{C}$.

Setup. $\mathcal{A}$ with an identity $aid \in U_{AA}$ runs the algorithm **GlobalSetup**$(1^\lambda) \rightarrow GP$ to get the global public parameters GP, then runs the algorithm **AASetup**$(GP, aid) \rightarrow (APK_{aid}, ASK_{aid})$ to get the key pairs (APK_{aid}, ASK_{aid}). At last, $\mathcal{A}$ sends the global parameters GP and the public keys APK_{aid} to the challenger $\mathcal{C}$.

User's secret key queries. $\mathcal{B}$ submits $H(uid)$ to $\mathcal{C}$ and obtains $(H(uid), H(uid)^\chi)$. As the same in [52], by the zero-knowledge property of the proof system, $\mathcal{B}$ uses a simulator to simulate the required proof without of knowledge of χ and submits $(uid, H(uid)^\chi)$ to $\mathcal{A}$.

$\mathcal{A}$ runs the algorithm **USKGen**$(uid, T(a), a \in U) \rightarrow USK_{uid,a}$ to get the users' secret keys, then sends $USK_{uid,a}$ to $\mathcal{B}$.

The secrect key for the attribute a of the user uid is in the form of

$$\langle K_{uid,a,1}, K_{uid,a,2}, K_{uid,a,3}, K_{uid,a,4}, K_{uid,a,5} \rangle.$$

Key forgery. The adversary $\mathcal{A}$ outputs a decryption key USK'_* in the form of

$$\langle K_{*,0}, K_{*,1}, K_{*,2}, K_{*,3}, K_{*,4}, K_{*,5} \rangle.$$

to $\mathcal{B}$, $\mathcal{B}$ sends $K_{*,0}$ to $\mathcal{C}$. If $\mathcal{A}$ wins the game with a non-negligible advantage ε, then $\mathcal{B}$ breaks the DLP assumption with the same advantage.■

C. FORWARD AND BACKWARD SECURITY

Theorem 5 proves that the proposed scheme is secure against the collusion attack launched by revoked users and nonrevoked users.

Theorem 5: In the proposed scheme, the revoked user has no chance to update its secret key even if she/he colludes with the non revoked users and corrupts with the attribute authorities which do not manages the revoked attribute.

Proof: As the same in [38], when an attribute a is revoked from the data user uid, each update key $UpK_{a,uid'} = H(uid')^{\frac{v'_a - v_a}{\gamma_{T(a)} + uid' + \eta_{T(a)} y_a}}$ for the non revoked user $uid' \neq uid$ is associated with the hash value of her/his unique identity which prevents the revoked user from updating her/his secret keys with the other users' update keys.

Meanwhile, it is hard for the non revoked user to calculate $\frac{v'_a - v_a}{\gamma_{T(a)} + uid' + \eta_{T(a)} y_a}$ by solving the discrete logarithm problem, which prevents her/him from working out her/his update key even if s/he colludes with the non revoked users and corrupts with the attribute authorities which do not manages the revoked attribute.■

Theorem 6 proves that the proposed scheme meets the requirements of the forward and backward security.

Theorem 6: The proposed scheme meets the requirements of the forward and backward security in the context of attribute revocation.

Proof: As the same in [38], when an attribute a is revoked from the data user uid, all the involved ciphertext whether the previous ciphertext or the newly ciphertext have been re-encrypted by the lastest re-encrypt key $ReK'_a = v'_{\rho(i)}$: $C'_{3,i} = g^{\beta_{\delta(i)} r_i} g^{v'_{\rho(i)} r_i} g^{w_i}$, where $a = \rho(i)$. It is hard for the revoked user to stretch the re-encrypted ciphertext back to the previous version ciphertext she/he can properly decrypt.

Meanwhile, the newly joined user who has the attribute a is able to decrypt any a-corresponding ciphertext.■

VII. PERFORMANCE ANALYSIS

In this section, the author analyzes and compares the function and performance of the proposed scheme and several related MA-CP-ABE schemes in the theoretical method. Then the author compares their efficiency in the experimental method. The notations are summarized in Table 2.

A. THEORETICAL ANALYSIS

Firstly, the author compares the proposed scheme's properties to those of the previous schemes in Table 3. The proposed scheme outperforms other existing relevant schemes. All these schemes are based on the prime order bilinear group and static security. They all support the large attribute universe. But only the proposed scheme supports simultaneously user accountability, authority accountability, user-attribute revocation, and outsourcing decryption.

Although the proposed scheme has more functions than other previous schemes, it just needs a little more operations than them in terms of computational performance as shown in Table 4 and Table 5. In terms of encryption computation efficiency, the proposed scheme needs the same operations as the scheme in [35]. It is just a little more exponentiation operations than the schemes in [12], [38]. Concerning the computation efficiency of the decryption, the proposed scheme just needs a little more exponentiation operations, and multiplication operations than the scheme in [35]. Especially compared to the schemes in the outsource decryption model, the proposed scheme just needs a little more exponentiation operations, and multiplication operations than the scheme in [35] in terms of pre-decryption and CSP-decryption. Moreover, as shown in Table 5, when an attribute is revoked from a user, the proposed scheme needs the same operations as the scheme in [38] in terms of updating the involved users' secret keys and the involved ciphertext. These are shown more intuitive and clearer in Figure 2.

Finally, the author compares the storage overhead of these schemes in Table 6. The proposed scheme has the same storage overhead as the schemes in [12], [35], [38] in terms of the global public parameters. The attribute authority in the proposed scheme need a few more storage overheads than it in the schemes [12], [35], [38], since each attribute authority's major storage overhead is derived from its master secret key, public key, and global public parameters. The storage overhead for the data owner is derived from the authorities' public keys and global public parameters. So, the data owner's

TABLE 2. The notations in performance analysis.

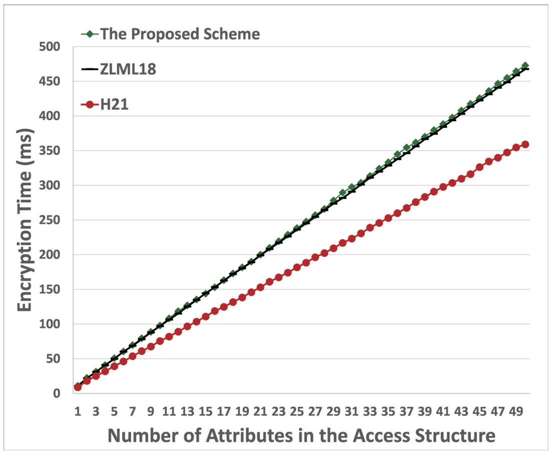
Notation	Description
M/M_T	one multiplication operation in the group $\mathbb{G}/\mathbb{G}_T$
E/E_T	one exponentiation operation on $\mathbb{G}/\mathbb{G}_T$
P	one pairing operation
H	one hash operation
N	the number of the attributes in the system
N_a	the number of attributes managed by the authority AA_a
S_a	the number of the attributes in user's private key
l	the complexity of the access structure
I	the number of attributes required for the decryption
l_Z	the size of an element in $\mathbb{Z}_p$
l_G	the size of an element in $\mathbb{G}$
l_{G_T}	the size of an element in $\mathbb{G}_T$

storage overhead in the proposed scheme is not more than it in the scheme [35]. The data user's storage cost is mostly driven by attribute-related secret keys, authorities' public keys, and global public parameters, implying that the data user's storage cost in the proposed scheme is only slightly higher than in the previous schemes.

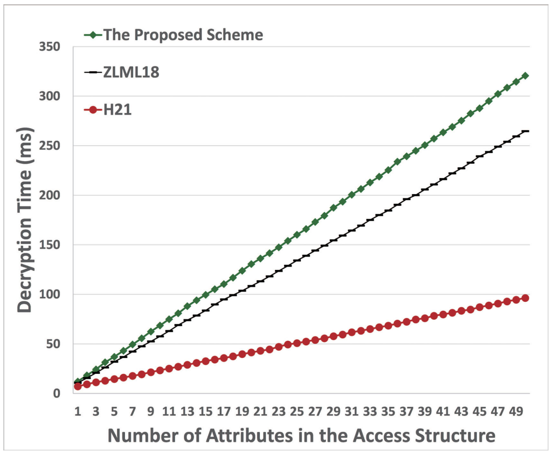
B. EXPERIMENTAL ANALYSIS

The author implemented the ZLML18 scheme [35], the H21 scheme [38], and the proposed scheme in Charm which is an extensible framework for rapidly prototyping cryptographic systems, and supports a variety of C math libraries such as the Stanford Pairing-Based Crypto library [53], [54]. In the programs, the author used a super-singular symmetric elliptic curve group ("SS512") whose order and base field sizes are 160-bit order and 512-bit respectively. The author conducted the programs in 64-bit Linux Ubuntu 18.04.4 installed on virtual machine platform: Vmware@Workstation 15 Pro 15.5.2 build-15785246, running on a laptop with a 2.30Ghz Intel Cored CPU and 2GB RAM. The number of attributes is ranging from 1 to 50. The results of all experiments are the average of 100 trials. Finally, the graphs were drawn to compare the computation cost of these schemes in Figure 2. As shown in Figure 2, the costs of encryption, decryption, updating the user's private keys, and updating the ciphertext increase linearly with the number of attributes. From Figure 2(a), the encryption time of the proposed scheme is a little more than it in the H21 scheme [38] but almost the same as it in the ZLML18 scheme [35]. In Figure 2(b), it is simple to see that the proposed scheme requires a little more decryption time than the ZLML18 scheme [35]. In the outsourcing decryption model, the final decryption just needs a constant amount of computation, so its time is almost negligible. Considering the cost of revocation, Figure 2(d-e) shows that the time of updating the involved users' secret keys and the involved ciphertext in the proposed scheme are almost the same as it in the H21 scheme [38].

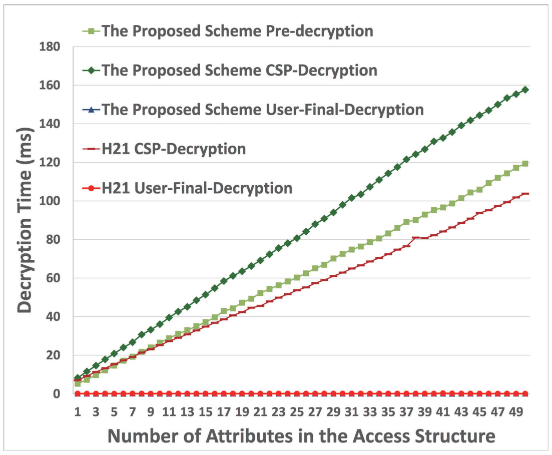
Therefore, the proposed scheme is efficient in terms of the encryption cost, the decryption cost, and the revocation cost, even though it has more functions than the previous schemes.



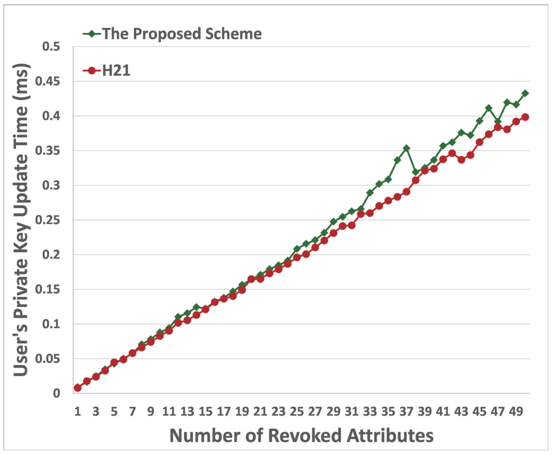
(a) The encryption cost



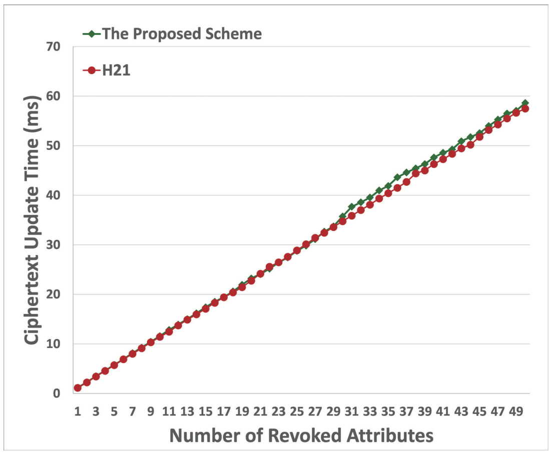
(b) The decryption cost



(c) The decryption cost in outsourcing decryption mode



(d) User's private keys update cost



(e) Ciphertext update cost

FIGURE 2. Experimental performance comparisons.

TABLE 3. Comparison of properties with previous works.

Schemes	Multi-authority	Large attribute-universe	User-attribute revocation	User accountability	Authority accountability	Outsource decryption	Security
RW15[12]	√	√	×	×	×	×	Static-security
ZLML18[35]	√	√	×	Traceable	×	×	Static-security
H21[38]	√	√	√	×	×	√	Static-security
Ours	√	√	√	√	√	√	Static-security

TABLE 4. Comparison of computation cost.

Schemes	Encryption (DO)	Re-encryption (CSP)	Decryption (DU)	Outsourcing decryption model			
				TK-generation	Pre-decryption (DU)	Decryption (CSP)	Final-decryption (DU)
RW15[12]	$lH + (2l+1)E_T + 4lE + (l+1)M_T + lM$	0	$3IP + IE_T + 4IM_T + H$	$\times$	$\times$	$\times$	$\times$
ZLML18[35]	$lH + (2l+1)E_T + 6lE + (l+1)M_T + lM$	0	$3IP + IE_T + 4IM_T + 3IE + 3IM + H$	$\times$	$\times$	$\times$	$\times$
H21[38]	$lH + (2l+1)E_T + 4lE + (l+1)M_T + lM$	$lE + lM$	$3IP + IE_T + 4IM_T + H$	$2IE$	0	$3IP + 2IE_T + (4I-2)M_T + H$	$E_T + 2M_T$
Ours	$lH + (2l+1)E_T + 6lE + (l+1)M_T + lM$	$lE + lM$	$3IP + IE_T + 4IM_T + 4IE + 4IM + H$	$2IE + IM$	$2IE + IM$	$3IP + 2IE_T + IE + (4I-2)M_T + 2IM + H$	$E_T + 2M_T$

TABLE 5. Comparison of computation cost for user-attribute revocation[1].

Schemes	User updates USK[2]	CSP updates ciphertext[3]
RW15[12]	–	–
ZLML18[35]	–	–
H21[38]	M	$E + M$
Ours	M	$E + M$

[1] One attribute is revoked from one user.
[2] One user updates his/her secret keys.
[3] CSP updates one involved ciphertext.

TABLE 6. Comparison of storage overhead.

Schemes	GP	AA's ASK	AA's APK	User's USK	Ciphertext
RW15[12]	$l_G + l_{G_T} + l_Z$	$2l_Z$	$l_{G_T} + l_G$	$2S_a l_G$	$(l+1)l_{G_T} + 3ll_G$
ZLML18[35]	$l_G + l_{G_T} + l_Z$	$4l_Z$	$l_{G_T} + 3l_G$	$3S_a l_G + (1+S_a)l_Z$	$(l+1)l_{G_T} + 5ll_G$
H21[38]	$l_G + l_{G_T} + l_Z$	$(2+N_a)l_Z$	$l_{G_T} + l_G$	$2S_a l_G$	$(l+1)l_{G_T} + 3ll_G$
Ours	$l_G + l_{G_T} + l_Z$	$(4+N_a)l_Z$	$l_{G_T} + 3l_G$	$3S_a l_G + (1+2S_a)l_Z$	$(l+1)l_{G_T} + 5ll_G$

VIII. CONCLUSION

In this article, the author proposes the first accountable and revocable large universe multi-authority attribute-based encryption scheme with outsourcing decryption based on prime order bilinear groups. An audit mechanism is given to judge if the suspicious key was leaked by a malicious user or by authorities and to determine the identity of the leaker. The malicious user who divulges key can be punished by user-attribute revocation. The revocation mechanism is resistant to collusion attacks undertaken by revoked users and non-revoked users. Meanwhile, it satisfies the requirements of forward and backward security. The proposed scheme is static security in the random oracle model under the q-DPBDHE2 assumption. To save resources, the outsourced decryption module is optional for users with restricted resources. According to the results of the performance analysis, it is suited for large-scale cross-domain cooperation in the dynamic cloud-aided IoT. However, the author considers improving the scheme for security without the random oracle model and more efficiency in future work.

ACKNOWLEDGMENT

The author would like to thank the anonymous reviewers for their comments which helped to improve the paper.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] A. Botta, W. Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and Internet of Things: A survey," *Future Generat. Comput. Syst.*, vol. 56, pp. 684–700, Mar. 2016.

[3] S. A. Hamad, Q. Z. Sheng, W. E. Zhang, and S. Nepal, "Realizing an internet of secure things: A survey on issues and enabling technologies," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1372–1391, 2nd Quart., 2020.

[4] 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.

[5] 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.

[6] 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.*, vol. 9, no. 2, pp. 983–993, Apr. 2021.

[7] M. Chase, "Multi-authority attribute based encryption," in *Proc. Theory Cryptogr. Conf.* Berlin, Germany: Springer, 2007, pp. 515–534.

[8] M. Chase and S. S. M. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proc. 16th ACM Conf. Comput. Commun. Secur. (CCS)*, 2009, pp. 121–130.

[9] 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.

[10] A. Lewko, "Tools for simulating features of composite order bilinear groups in the prime order setting," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2012, pp. 318–335.

[11] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 463–474.

[12] Y. Rouselakis and B. Waters, "Efficient statically-secure large-universe multi-authority attribute-based encryption," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2015, pp. 315–332.

[13] J. Li, K. Ren, and K. Kim, "A^{2}BE: Accountable attribute-based encryption for abuse free access control," IACR Cryptol. ePrint Arch., IACR, Tech. Rep. 118, 2009. [Online]. Available: https://eprint.iacr.org/2009/118.pdf

[14] Z. Liu, Z. Cao, and D. S. Wong, "White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 1, pp. 76–88, Jan. 2013.

[15] J. Ning, Z. Cao, X. Dong, L. Wei, and X. Lin, "Large universe ciphertext-policy attribute-based encryption with white-box traceability," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2014, pp. 55–72.

[16] J. Ning, Z. Cao, X. Dong, and L. Wei, "White-box traceable CP-ABE for cloud storage service: How to catch people leaking their access credentials effectively," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 5, pp. 883–897, Sep./Oct. 2018.

[17] H. Cui, R. H. Deng, and Y. Li, "Attribute-based cloud storage with secure provenance over encrypted data," *Future Gener. Comput. Syst.*, vol. 79, no. 2, pp. 461–472, Feb. 2018.

[18] H. Cui, R. H. Deng, B. Qin, and J. Weng, "Key regeneration-free ciphertext-policy attribute-based encryption and its application," *Inf. Sci.*, vol. 517, pp. 217–229, May 2020.

[19] X. Yan, X. He, J. Yu, and Y. Tang, "White-box traceable ciphertext-policy attribute-based encryption in multi-domain environment," *IEEE Access*, vol. 7, pp. 128298–128312, 2019.

[20] Z. Liu, Z. Cao, and D. S. Wong, "Blackbox traceable CP-ABE: How to catch people leaking their keys by selling decryption devices on ebay," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 475–486.

[21] Z. Liu and D. S. Wong, "Traceable CP-ABE on prime order groups: Fully secure and fully collusion-resistant blackbox traceable," in *Proc. Int. Conf. Inf. Commun. Secur.* Cham, Switzerland: Springer, 2015, pp. 109–124.

[22] Z. Liu and D. S. Wong, "Practical attribute-based encryption: Traitor tracing, revocation and large universe," *Comput. J.*, vol. 59, no. 7, pp. 983–1004, Jul. 2016.

[23] J. Ning, Z. Cao, X. Dong, J. Gong, and J. Chen, "Traceable CP-ABE with short ciphertexts: How to catch people selling decryption devices on eBay efficiently," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2016, pp. 551–569.

[24] 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.

[25] Y. Ye, Z. Cao, and J. Shen, "Unbounded key-policy attribute-based encryption with black-box traceability," in *Proc. IEEE 19th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Dec. 2020, pp. 1655–1663.

[26] Z. Liu, Q. Huang, and D. S. Wong, "On enabling attribute-based encryption to be traceable against traitors," *Comput. J.*, vol. 64, no. 4, pp. 575–598, Apr. 2021.

[27] J. Ning, X. Dong, Z. Cao, and L. Wei, "Accountable authority ciphertext-policy attribute-based encryption with white-box traceability and public auditing in the cloud," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2015, pp. 270–289.

[28] G. Yu, Z. Cao, G. Zeng, and W. Han, "Accountable ciphertext-policy attribute-based encryption scheme supporting public verifiability and nonrepudiation," in *Proc. Int. Conf. Provable Secur.* Cham, Switzerland: Springer, 2016, pp. 3–18.

[29] Y. Zhang, J. Li, D. Zheng, X. Chen, and H. Li, "Accountable large-universe attribute-based encryption supporting any monotone access structures," in *Proc. Australas. Conf. Inf. Secur. Privacy*. Cham, Switzerland: Springer, 2016, pp. 509–524.

[30] G. Yu, Y. Wang, Z. Cao, J. Lin, and X. Wang, "Traceable and undeniable ciphertext-policy attribute-based encryption for cloud storage service," *Int. J. Distrib. Sensor Netw.*, vol. 15, no. 4, pp. 1–10, Apr. 2019, doi: 10.1177/1550147719841276.

[31] J. Lai and Q. Tang, "Making any attribute-based encryption accountable, efficiently," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2018, pp. 527–547.

[32] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh, and D. Wang, "Attribute based encryption with privacy protection and accountability for CloudIoT," *IEEE Trans. Cloud Comput.*, early access, Feb. 19, 2020, doi: 10.1109/TCC.2020.2975184.

[33] J. Li, Q. Huang, X. Chen, S. S. M. Chow, D. S. Wong, and D. Xie, "Multi-authority ciphertext-policy attribute-based encryption with accountability," in *Proc. 6th ACM Symp. Inf., Comput. Commun. Secur. (ASIACCS)*, 2011, pp. 386–390.

[34] J. Li, X. Chen, S. S. M. Chow, Q. Huang, D. S. Wong, and Z. Liu, "Multi-authority fine-grained access control with accountability and its application in cloud," *J. Netw. Comput. Appl.*, vol. 112, pp. 89–96, Jun. 2018.

[35] 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, pp. 1–13, Mar. 2018.

[36] K. Xue, Y. Xue, J. Hong, W. Li, H. Yue, D. S. L. Wei, and P. Hong, "RAAC: Robust and auditable access control with multiple attribute authorities for public cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 953–967, Apr. 2017.

[37] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proc. 15th ACM Conf. Comput. Commun. Secur. (CCS)*, 2008, pp. 417–426.

[38] K. Huang, "Secure efficient revocable large universe multi-authority attribute-based encryption for cloud-aided IoT," *IEEE Access*, vol. 9, pp. 53576–53588, 2021.

[39] N. Attrapadung and H. Imai, "Conjunctive broadcast and attribute-based encryption," in *Proc. Int. Conf. Pairing-Based Cryptogr.* Berlin, Germany: Springer, 2009, pp. 248–265.

[40] N. Attrapadung and H. Imai, "Attribute-based encryption supporting direct/indirect revocation modes," in *Proc. IMA Int. Conf. Cryptogr. Coding*. Berlin, Germany: Springer, 2009, pp. 278–300.

[41] 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.

[42] 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.

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

[44] 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.

[45] 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.

[46] J. Li, Y. Wang, Y. Zhang, and J. Han, "Full verifiability for outsourced decryption in attribute based encryption," *IEEE Trans. Services Comput.*, vol. 13, no. 3, pp. 478–487, May 2020.

[47] N. Chen, J. Li, Y. Zhang, and Y. Guo, "Efficient CP-ABE scheme with shared decryption in cloud storage," *IEEE Trans. Comput.*, early access, Dec. 14, 2020, doi: 10.1109/TC.2020.3043950.

[48] 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.

[49] D. Boneh and X. Boyen, "Short signatures without random oracles and the SDH assumption in bilinear groups," *J. Cryptol.*, vol. 21, no. 2, pp. 149–177, 2008.

[50] A. Beimel, "Secure schemes for secret sharing and key distribution," Ph.D. dissertation, Fac. Comput. Sci., Technion-Israel Inst. Technol., Haifa, Israel, 1996.

[51] M. Bellare and O. Goldreich, "On defining proofs of knowledge," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 1992, pp. 390–420.

[52] V. Goyal, "Reducing trust in the PKG in identity based cryptosystems," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2007, pp. 430–447.

[53] 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, 2013.

[54] B. Lynn. *The Pairing-Based Cryptography Library*. Accessed: 2020. [Online]. Available: https://crypto.stanford.edu/pbc/

KAIQING HUANG received the B.E. and M.S. degrees in mathematics from South China Normal University, in 2007 and 2010, respectively, where he is currently pursuing the Ph.D. degree in applied mathematics. Since 2013, he has been with Dongguan Polytechnic, China, as a Lecturer. His research interests include applied cryptography and data security.

• • •