You Are Revoked and Out: Towards Directly Revocable Ciphertext-Policy Attribute-Based Encryption

Feng Yang,1 Limin Liu,2 Weijing You,3 and Jiwu Jing1

1School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China
2State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100089, China
3College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350000, China

Correspondence should be addressed to Weijing You; youweijing@fjnu.edu.cn

Received 26 November 2021; Revised 31 May 2022; Accepted 21 June 2022; Published 23 September 2022

Academic Editor: Helena Rifà-Pous

Copyright © 2022 Feng Yang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Ciphertext-policy attribute-based encryption (CP-ABE) is considered as a promising cryptographic primitive to enable fine-grained access control over encrypted data. Throughout the life circle of the data encrypted with CP-ABE, every single data user might opt out or be identified to be malicious and hence should be revoked to keep continuous access control over sensitive data. In this study, we propose a directly revocable CP-ABE with backward and forward secrecy, which means that a revoked user cannot successfully decrypt ciphertexts after sufficient revocation while valid users will not be disturbed. Combined with an effective proxy mechanism, the proposed scheme delegates complete workloads of re-encryption and ciphertext updating to a semi-trusted third party (e.g., a cloud service provider). In addition, even collusion among users is useless in our construction, which is not well addressed in previous works. The security analysis indicates that the presented CP-ABE is selectively secure against chosen-plaintext attacks in the standard model, and the performance analysis demonstrates that our proposal is practical compared with existing schemes.

1. Introduction

Nowadays, cloud computing has been increasingly used to facilitate data sharing [1]. For security purposes, cloud users often encrypt their data before outsourcing. However, when cloud users selectively share data to different recipients, it would be expensive to generate different encrypted data and transmit them respectively. A fine-grained level of access control over encrypted data in the cloud is desirable. Sahai and Waters [2] firstly address this obstacle by introducing attribute-based encryption (ABE), in which a user’s decryption keys and ciphertexts are labeled with a set of descriptive attributes, and the ciphertext can be decrypted only when there exists a “match” between attributes of this ciphertext and the decryption key. Based on where the access policies are embedded, ABE can be categorized into key-policy attribute-based encryption (KP-ABE) [3] and ciphertext-policy attribute-based encryption (CP-ABE) [4]. In a CP-ABE system, an access policy is specified in a ciphertext, while users obtain different decryption keys based on their own attributes. Since the access policy is embedded in the ciphertext and fully controlled by the data owner in CP-ABE, we focus on CP-ABE in this study. For instance, Alice owns an attribute set {“Student”,”Computer Science”,”Undergraduate”}, Bob owns an attribute set {”Professor”,”Computer Science”,”In-Service”}, and Carol owns an attribute set {”Professor”, ”Astronomy”, ”Retired”}. Suppose that there is a ciphertext generated under the access policy {”Professor” AND ”In-Service”}. Since only Bob has the attributes ”Professor” and ”In-Service” simultaneously, Bob can decrypt successfully.

The access control is a fundamental functionality of CP-ABE, which definitely should be effective throughout the entire life circle of the encrypted data. However, the time for each user to get access to the data is not always identical, especially in the flexible clouds, and the users could be...
An effective revocation mechanism is strongly desirable for the traceable CP-ABE. In a CP-ABE with revocation mechanism, the expired users or valid users whose decryption keys are disclosed should be blocked out of accessing the data to become who are called “revoked users” in general. A data user could retrieve the original data successfully if and only if the attribute sets he/she possesses can satisfy the access policy and, more, his/her identity is absent from the revocation list. In the light of the effect on non-revoked users, the revocation is categorized into indirect revocation [8–11] and direct revocation [12–15]. The difference lies in whether the non-revoked users’ decryption keys should be updated when a revocation event occurs.

In indirect revocation schemes, the authority periodically issues key updates for non-revoked users. In addition, all non-revoked users need to communicate with the authority and update their decryption keys. In particular, the indirect revocation also consists of two stages. Firstly, the original indirect revocation mechanism was adopted from identity-based encryption (IBE) [16]. In the Boneh and Franklin construction [16], each user’s decryption key will expire after a valid period, so the revocation can be achieved by stopping issuing new decryption keys for revoked users. However, such a straightforward revocation brings a relatively large computational and communication overhead. The authority needs to generate a new decryption key for each non-revoked user separately and transmit it over a secure channel. Secondly, to improve the efficiency of key updates, Boldyreva et al. [17] introduced a binary tree data structure [18], in which each user is associated with a unique leaf node. Every user gets decryption keys computed on nodes of the path from the leaf node (corresponding to that user) to the root node. It means that different users would share some nodes in their path from a leaf node to the root node; for example, all users will share the root node. In this way, when a subset of the data users is revoked, we can find a minimum cover set of nodes that contains an ancestor (or the node itself) for each leaf node related to a non-revoked user. Then, the key update can be performed within this subset, thereby reducing the workload from linear to logarithmic in the number of non-revoked users.

After the work of Boldyreva et al. [17], many indirectly revocable CP-ABE schemes [8–11, 19–21] that adopted binary tree data structure have been proposed. However, apart from the natural expiration, a data user may lose its decryption key or disclose the key deliberately for economic purposes. Such a user is supposed to be revoked immediately to guarantee continuous access control. In direct revocation mechanisms, a data owner embeds the latest revocation list into the ciphertext during the encryption phase so that the revoked users cannot decrypt a newly generated ciphertext anymore. There is no need for any key update. Even though in an indirect revocation, a data owner does not need to care about the revocation list. Since the revocation list can be published by the authority, it is not a tough task for a data owner to obtain the latest revocation list. Considering that in a traceable CP-ABE, when a malicious user is caught, it should be revoked immediately. A direct revocation mechanism is more suitable for traceable systems. Some researchers [15, 22] adapted the binary tree data structure into direct revocable constructions and realized traceable and revocable CP-ABE systems.

However, most of the existing revocable CP-ABE schemes either do not support traceability or cannot resist collusion between a revoked user and a non-revoked user. One may think that we do not need to focus on such attacks since a non-revoked user can naturally decrypt the ciphertext and share the plaintext with a revoked user, but we note that this may only be somewhat acceptable in a CP-ABE without a traceable functionality. More concretely, in a CP-ABE with white-box traceability [15, 22], the system can catch a malicious user who discloses his/her decryption key (e.g., selling the key online). The trace mechanism needs a full disclosed decryption key to identify the malicious user. It will not work if there is only a partial decryption key. As a result, the trace mechanism would be meaningless if a non-revoked user’s partial decryption key is enough to restore a revoked user’s decryption privileges. Moreover, since it has been already supposed that a non-revoked user might disclose the full decryption key deliberately in a traceable CP-ABE, we can no longer require the user not to disclose a partial decryption key in a revocable CP-ABE that supports traceability. If a user is willing to disclose a full decryption key, he/she can reveal a partial decryption key since it will be safer than disclosing the full one. Therefore, for such a traceable and revocable CP-ABE scheme, the key generation is supposed to be completely controlled by the trusted authority, and any part of another user’s decryption key should be useless to revalidate a revoked user’s decryption key.

Some researches [10, 19] focus on resisting the collusion between a revoked user and a non-revoked user, but these are indirectly revocable schemes and do not support traceability.

There are another two special security properties that should be considered in revocable access control systems: forward secrecy and backward secrecy [23]. Forward secrecy and backward secrecy mean that a revoked user should be prevented from decrypting the subsequent and previous ciphertexts, respectively. Only systems that support both forward secrecy and backward secrecy can prevent a revoked user from learning any sensitive data [8]. Forward secrecy is implicitly realized during revocation, and a revoked user cannot access the data encrypted after the revocation. As the decryption key of a revoked user is still able to decrypt the ciphertext generated previously, a ciphertext update algorithm is further needed to achieve backward secrecy. An intuitive solution to handle this problem is to decrypt and re-encrypt the stored data whenever some users are revoked. However, this two-way communication makes the system
cumbersome and inefficient. Sahai et al. [8] firstly discussed this situation and dealt with it by introducing a ciphertext delegation that allows a ciphertext to be updated using only public information. In the follow-up schemes, ciphertext update falls into two types: partial update [24, 25] and full update [26–29]. In the partial update, only those components associated with the revocation mechanism are updated. In the full update, almost all components except the underlying plaintext will be updated. The latter provides better protection as any intermediate values computed for the previous ciphertext will be useless for decrypting the refreshed ciphertext. Among recent published schemes that support ciphertext update functionality [24–29], only Li et al. [25] and Sethi et al. [27] realized traceability, but the access policy of [25] can only be expressed as AND gate that sacrifices the flexibility and expressiveness of the CP-ABE. Sethi et al. [27] adopted the expressive linear secret-sharing schemes (LSSS) into access policy, but their proposal is an indirect revocable scheme that allows malicious users to maintain decryption capabilities until the next key update.

Traceable and revocable CP-ABE is a promising solution for data sharing and life cycle access control, but a large number of complex bilinear group operations bring non-negligible computational pressure on resource-constrained devices (e.g., mobile phones and IoT systems) or time-sensitive devices (e.g., ad-hoc vehicular networks).

We revisit the existing CP-ABE schemes and find that most of them only focus on one specific functionality, or do not fully consider the compatibility between different functionalities. This hinders the CP-ABE for real-world applications. Motivated by the aforementioned limitations, we aim to propose a practical solution towards traceable and revocable CP-ABE for cloud computing applications. In this work, we build a directly revocable CP-ABE with traceability and ciphertext update to catch a malicious user and then revoke it immediately and achieve forward and backward secrecy. Besides, we present the ciphertext delegation to outsource to a semi-trusted third party (e.g., cloud service provider) the burden of re-encryption and updating ciphertexts, which is friendly to users with limited computing resources.

1.1. Contribution. The main contributions of the scheme developed in this study are as follows:

(i) **Direct Revocation.** We realize a directly revocable CP-ABE scheme whereby a revoked user cannot successfully decrypt ciphertexts anymore after the sufficient revocation, while legitimate (non-revoked) users will not be disturbed. We adopt the binary tree data structure to improve efficiency. Besides, our direct revocation mechanism saves the communication overhead between the authority and non-revoked users since there is no additional periodic key update.

(ii) **Collusion Resistance.** In our scheme, any component of another user’s decryption key will make no sense in revalidating a revoked user’s decryption key. As a white-box trace mechanism can identify a malicious user who discloses a full decryption key, the collusion resistance capability is necessary for ensuring the effectiveness of white-box traceability in CP-ABE. Combined with our revocation mechanism, we succeed in a continuous process from discovering malicious users to revoking malicious users.

(iii) **Friendly to Resource-Constrained Devices.** We propose a ciphertext delegation mechanism composed of an outsourced re-encryption algorithm and an outsourced ciphertext update algorithm. The algorithms make good use of computing resources provided by a semi-trusted party (e.g., a cloud service provider), reducing data owners’ burden in terms of communication and computation.

(iv) **IND-CPA Secure.** The scheme is proven secure in the security definition of indistinguishability under selective chosen-plaintext attacks (IND-CPA secure) in the standard model based on the decisional q-BDHE assumption.

1.2. Organization. The rest of the study is organized as follows. In Section 2, we review necessary preliminaries used in our scheme. The formal definition and security model are described in Section 3. Section 4 presents the concrete structure of the scheme. The security analysis is given in Section 5. Comparisons of the proposed scheme are made with several related works in Section 6. Section 7 introduces some related works. Section 8 offers a conclusion.

2. Preliminaries

This section describes the necessary background information for the proposed scheme, including bilinear maps, access structures, secret-sharing schemes, binary trees, and hardness assumptions.

2.1. Access Structure

**Definition 1** (Access Structure [30]). Let \( P = \{P_1, P_2, \ldots, P_n\} \) be a set of parties. A collection \( A \subseteq 2^P \) is called monotone if \( \forall B, C: \) if \( B \in A \) and \( B \subseteq C \), then \( C \in A \). The (monotone) access structure is a (monotone) collection \( A \) of non-empty subsets of \( P \), i.e., \( A \subseteq 2^P/\{\emptyset\} \). The sets in \( A \) are called authorized sets, or else the sets are called the unauthorized sets.

In CP-ABE, since the role of the parties is taken by the attributes, the access structure contains the authorized sets of attributes. We restrict our attention to the monotone access structure in this study.

2.2. Linear Secret-Sharing Scheme

**Definition 2** (Linear Secret-Sharing Schemes (LSSS) [31]). A secret-sharing scheme \( \Pi \) over a set of parties \( P \) is called linear (over \( \mathbb{Z}_p \)) if

(i) The shares for each party form a vector over \( \mathbb{Z}_p \).
(ii) There exists a matrix $M$ with $\ell$ rows and $n$ columns called the share-generating matrix for $\Pi$. For $i = 1, 2, \ldots, \ell$, the $i$th row $M_i$ of $M$ is labeled by a party $\rho(i)$, where $\rho$ is a function that maps a row of $M$ into a party in $\mathcal{P}$. Consider the column vector $\mathbf{v} = (s, t_2, t_3, \ldots, t_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $t_2, t_3, \ldots, t_n \in \mathbb{Z}_p$ are randomly selected. Then, $M^T \mathbf{v}$ is the vector of $I$ shares of the secret $s$; here, the share $\lambda_i = M_i \cdot \mathbf{v}$ belongs to the corresponding party $\rho(i)$.

It is shown in [32] that according to the above definition, each secret-sharing scheme also enjoys the linear reconstruction property, defined as follows.

Assuming that $\Pi$ is an LSSS for the access structure $\mathcal{A}$, suppose $S \in \mathcal{A}$ is any authorized set, and let $I \in \{1, 2, \ldots, \ell\}$ be defined as $I = \{i: \rho(i) \in S\}$. There exists a set of constants $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ satisfying that $\sum_{i \in I} w_i M_i = (1, 0, \ldots, 0)$, such that, if $\lambda_i$ is valid shares of the secret $s$ according to $\Pi$, we have $\lambda_i \in \mathbb{Z}_p$. In addition, those constants $\{w_i\}$ can be found in polynomial time in the size of the share-generating matrix $M$ [32].

### 2.3. Binary Tree

Let $\mathcal{U}$ be the set of users and $R$ be the revocation list in system. A binary tree will be used in both key generation and ciphertext generation processes. Now, we briefly introduce the binary tree $BT_5$ [15] as follows. A leaf node in the binary tree is associated with a user in the system. Let $d$ denote the depth of the binary tree, we have $|\mathcal{U}| = 2^d$, and the number of nodes in this tree is $2|\mathcal{U}| - 1$. Those nodes are numbered by the breadth-first search; as such, the root is 0 and the last node is 2$|\mathcal{U}| - 1$. Let $\text{path}(u) = \{\text{root}, \ldots, u_{\overline{d}}\}$ denote the path from the root to a leaf node related to $u$, where root denotes the root node, and $u_{\overline{d}}$ is the leaf node associated with the user $u$. The minimum cover set $\text{cover}(R)$ is a minimum set of nodes that cover all users who are not in the revocation list $R$. Note that for $\forall u \in R$, we have $\text{cover}(R) \cap \text{path}(u) = \emptyset$. As for user $u \notin R$, there exists only one node in the intersection of $\text{cover}(R)$ and $\text{path}(u)$.

We take Figure 1 as an example of the binary tree and by supposing that the revoked user is $u_5$, such that revocation list $R = \{u_5\}$. Then, we get $\text{cover}(R) = \{1, 6, 12\}$. For a non-revoked user, $u_1$, for example, $\text{path}(u_1) = \{0, 1, 3, 7\}$. The interaction of $\text{cover}(R)$ and $\text{path}(u_1)$ is $\{1\}$.  

![Binary tree](image)

**Figure 1:** Binary tree. (a) No user is revoked. $R = \emptyset$ and $\text{cover}(R) = \{0\}$. (b) User $u_5$ is revoked. $R = \{u_5\}$ and $\text{cover}(R) = \{1, 6, 12\}$.  

### 2.4. Bilinear Map and Complexity Assumptions

In this section, we first introduce the bilinear map used in our scheme and then briefly describe the $q$-bilinear Diffie–Hellman exponent ($q$-BDHE) assumption and the $l$-strong Diffie–Hellman ($l$-SDH) assumption, which will be used to prove the security of the proposed scheme.

**Bilinear Map.** Let $G$ and $G_T$ be two multiplicative cyclic groups with prime order $p$, and $g$ be a generator of $G$. A map $e: G \times G \rightarrow G_T$ is a bilinear map [16] if it satisfies the properties:

1. **Bilinearity:** for $\forall u, v \in G, a, b \in \mathbb{Z}_p^*$, $e(u^a, v^b) = e(u, v)^{ab}$
2. **Non-Degeneracy:** $e(u, v) \neq 1$
3. **Computability:** for $\forall u, v \in G$, there exists an efficient algorithm to compute $e(u, v)$

**Assumption 1 (q-BDHE Assumption [31]).** Let $G$ be a bilinear group of prime order $p$, and let $g$ be a generator of $G$. A decisional $q$-BDHE problem can be described as follows: $d, s \in \mathbb{Z}_p^*$ is randomly chosen and given as follows:

$$\mathbf{y} = (g, g^s, g^{d_1}, \ldots, g^{d_1s}, g^{d_2}, g^{d_2s}, \ldots, g^{d_\ell}, g^{d_\ell s}).$$

It is hard for the algorithm $\mathcal{A}$ to distinguish $e(g, g)^{d^{s+1}} \in G_T$ from an element $Z$, which is randomly chosen from $G_T$. Algorithm $\mathcal{A}$ can solve the $q$-BDHE problem with the advantage $\epsilon$ if

$$\Pr[\mathcal{A}(\mathbf{y}, T = e(g, g)^{d^{s+1}}) = 1] - \Pr[\mathcal{A}(\mathbf{y}, T = Z) = 0] \geq \epsilon.$$  

**Definition 3.** The decisional $q$-BDHE assumption holds, if no polynomial-time algorithm $\mathcal{A}$ has a non-negligible advantage in solving the $q$-BDHE problem.

**Assumption 2 (l-SDH Assumption [33]).** Let $G$ be a bilinear group of prime order $p$, and let $g$ be a generator of $G$. An $l$-SDH problem can be described as follows: $x \in \mathbb{Z}_p^*$ is randomly chosen, and given a $(l + 1)$-tuple $(g, g^s, g^{s^2}, \ldots, g^{s^l})$, a pair $(c, g^u) \in \mathbb{Z}_p \times G$ is output. Algorithm $\mathcal{A}$ can solve the $l$-SDH problem with the advantage $\epsilon$ if
\[ \Pr[A(g, g^x, g^{x_2}, \ldots, g^{x_k}) = (c, g^{\ell(x+c)})] \geq \varepsilon. \] (3)

**Definition 4.** The l-SDH assumption holds, if no polynomial-time algorithm \( A \) can solve the l-SDH problem with a non-negligible advantage.

### 3. System and Security Models

In this section, we give the system architecture of the proposed CP-ABE scheme and formalize its definition and security model.

#### 3.1. System Architecture

The four entities in the system architecture are as follows:

(i) **Trusted Authority (TA).** TA publishes the public parameters of the system, generates decryption keys for non-revoked users, and is in charge of realizing traceability and updating the user revocation list \( R \). In our system, the TA is fully trusted.

(ii) **Data Owner (DO).** DO takes charge of specifying an access structure and of completing the local encryption under the access structure before outsourcing to a cloud service provider.

(iii) **Cloud Service Provider (CSP).** CSP is responsible for re-encrypting the original ciphertext that DO uploads and storing the re-encrypted ciphertext. CSP undertakes the task of updating ciphertexts once a revocation event occurs. We suppose that the CSP is honest but curious, which means CSP will faithfully execute every authorization request and not disclose the data, but attempt to obtain as much information as possible from both process and results.

(iv) **Data User (DU).** DU is an entity that intends to access encrypted data. A DU can eventually decrypt the ciphertext if and only if his/her attributes satisfy the access policy and he/she is not in the current revocation list.

#### 3.2. Formal Definition of ABE

The proposed scheme includes the following eight algorithms.

(i) **Setup** \((\lambda, U, BT) \rightarrow (APK, ASK)\): TA runs this setup algorithm and takes as input a security parameter \( \lambda \), an attribute universe \( U \), and the binary tree \( BT \) illustrated in Section 2.3. It outputs a public parameter \( APK \) and TA’s secret key \( ASK \). It initializes a user revocation list \( R = \emptyset \).

(ii) **KeyGen** \((ASK, u, S) \rightarrow SK_u\): the KeyGen algorithm takes as input a user identity \( u \), an attribute set \( S \) owned by \( u \), and the TA’s secret key \( ASK \). It outputs the corresponding decryption key \( SK_u \) for \( u \).

(iii) **LocalEncrypt** \((APK, m, (M, \rho), R) \rightarrow CT\): the data owner runs this algorithm and inputs the public parameter \( APK \), an LSSS access structure \((M, \rho)\), and the current user revocation list \( R \) and a message \( m \) to be encrypted. \( M \) is an \( l \times n \) matrix and \( \rho \) is a function that maps rows of \( M \) to attributes. The algorithm outputs the original ciphertext \( CT \).

(iv) **ReEncrypt** \((CT, APK, R) \rightarrow CT'\): given the original ciphertext \( CT \), the public parameter \( APK \), and the latest user revocation list \( R \), the semi-trusted cloud service provider runs this algorithm to produce and store re-encrypted ciphertext \( CT' \).

(v) **Decrypt** \((SK_u, CT) \rightarrow m \or1\): based on the ciphertext \( CT \) and a decryption key \( SK_u \), a data user can successfully decrypt if and only if he/she is absent from the current user revocation list and his/her attribute set satisfies the access structure of \( CT \); otherwise, it outputs \( \perp \) indicating a failed decryption.

(vi) **Trace** \((SK_u, APK, ASK) \rightarrow u \or1\): the tracing algorithm first determines whether the decryption key needs to be traced by verifying whether the \( SK_u \) is well-formed. A decryption key is well-formed if it passes KeySanityCheck \([34]\). If the decryption key \( SK_u \) is well-formed, the tracing algorithm outputs a user’s identity \( u \) indicating that \( SK_u \) is linked to the user \( u \). Otherwise, the tracing algorithm returns a symbol \( \perp \) implying that \( SK_u \) does not need to be traced.

(vii) **Revoke** \((u, R) \rightarrow R'\): the revocation algorithm takes in the revocation list \( R \) and a user \( u \) who is supposed to be revoked. TA runs this algorithm and publishes the latest revocation list \( R' \).

(viii) **CTUpdate** \((CT, R', APK, k) \rightarrow CT'\): the CTUpdate algorithm is executed by the CSP when a user revocation event occurs. It takes as input the latest user revocation list \( R' \), an update key \( k \), and the ciphertext \( CT \) to be updated and outputs the updated ciphertext \( CT' \).

#### 3.3. IND-CPA Security Model

We introduce the security definition of indistinguishability under selective chosen-plaintext attacks (IND-CPAs) for our scheme by the following game between an adversary and a challenger. Since the original ciphertext and the updated ciphertext are identically distributed, we only give the indistinguishability of the original ciphertext.

(i) **Init:** the adversary chooses a challenged access structure \((M^*, \rho^*)\) and a revocation list \( R^* \), where \( M^* \) is an \( l^* \times n^* \) matrix and \( n^* < q, \rho^* \) maps rows of \( M^* \) into attributes. The adversary sends \((M^*, \rho^*)\) and \( R^* \) to the challenger.

(ii) **Setup:** the challenger runs the setup algorithm and sends the public parameters \( APK \) to the adversary.

(iii) **Phase 1:** the adversary queries the challenger for decryption keys corresponding to \((u_{i_1}, S_{i_1}), \ldots, (u_{i_l}, S_{i_l})\). The challenger generates a decryption key
and returns it to the adversary except in the case that $u_i \notin R^* \land S_i \in (M^*, \rho^*)$.

(iv) **Challenge:** the adversary submits two equal length messages $m_0$ and $m_1$. The challenger flips a random coin $\beta \in \{0, 1\}$ and encrypts $m_0$ under the challenged access structure $(M^*, \rho^*)$ and the revocation list $R^*$. After that, the challenger sends the adversary the final challenge ciphertext $CT^*$.

(v) **Phase 2:** the Phase 2 is the same as Phase 1. The adversary submits a set of tuples $\langle u_{i_1}, S_{i_1} \rangle, \ldots, \langle u_{i_q}, S_q \rangle$ to ask for those related decryption keys.

(vi) **Guess:** the adversary outputs a guess $\beta' \in \{0, 1\}$ of $\beta$.

The advantage of an adversary in the above game is defined to be $|\text{Pr}[\beta' = \beta] - 1/2|$.

**Definition 5.** A directly revocable CP-ABE is selectively secure if all polynomial-time adversaries have at most negligible advantage in this security game.

### 3.4. Traceability Security Model

Here, we introduce the traceability model used in our scheme. We note that the model is similar to that of Liu et al.’s work [15], but since we have modified the structure of the decryption key, we will show that our scheme is adapted to this model in security analysis. The model is described by a security game between a challenger and an adversary as follows.

(i) **Setup:** the challenger runs the setup algorithm and then sends the public parameters $APK$ to the adversary.

(ii) **Key Query:** the adversary submits a series of tuples $\langle u_i, S_i \rangle, \ldots, \langle u_q, S_q \rangle$ to query the related decryption keys, where $u_i \in R$ or $S_i \notin (M^*, \rho^*)$, $i = 1, \ldots, q$.

(iii) **Key Forgery:** the adversary outputs a decryption key $SK^*_u$.

The adversary wins the game if $\text{Trace}(APK, R, SK^*_u) \notin \bot$ and $\text{Trace}(APK, R, SK^*_u) \notin \{u_1, \ldots, u_q\}$. Thus, the advantage of the adversary in this game is defined as $\text{Pr}[$Trace$(APK, R, SK^*_u) \notin \bot] \cup \{u_1, \ldots, u_q\}]$.

**Definition 6.** A directly revocable CP-ABE is traceable if there is no polynomial-time adversary having non-negligible advantage in the above game.

### 4. Our Construction

#### 4.1. Technical Overview

In this subsection, we provide a brief overview of our main ideas and techniques. We start with the prior revocation constructions of [15, 22] and the challenge in making them resistant to user collusion. In these systems, the trusted authority distributes a set of secret values to each user (e.g., $X_1 = \{x_{i_0}, x_{1}, x_{2}\}$ for a user and $X_2 = \{x_{i_0}, x_{1}, x_{2}\}$ for another user). The ciphertext is associated with a minimum cover set obtained according to a revocation list. The intersection of this minimum cover set and the set owned by each legitimate user has one and only one element, and the intersection calculated for a revoked user is empty. In the specific constructions, the minimum cover set is used to compute an intermediate value (e.g., $Y_{uv}$ in [15]) for decryption. That is to say, only users who have the elements in the minimum cover set can correctly calculate the intermediate value and perform subsequent decryption operations.

However, there is a security issue in such constructions. Since these secret values are distributed directly to legitimate users in key generation process, a revoked user can obtain a necessary secret value from a non-revoked user and illegally restore its decryption capability through such complicity without catching. Note that this system is very likely to suffer from user collusion attacks. As the same secret value is usually owned by multiple non-revoked users while the white-box tracing technology only works with a fully exposed key, it is almost impossible to determine who leaked a single secret element. As a result, not only is the revocation mechanism severely challenged, but the white-box tracing technology will also fail.

Our goal is to resist collusion between the revoked user and another user who still has decryption privilege, which is the bottleneck suffered by many schemes [15, 22, 35, 36]. We managed to keep the indispensable information for revocation out of the user’s reach, whereas in [15] such information is sent to legitimate users and used in decryption. If a user $u$ implicitly owns each secret value $x_i$ of the set $X_u$ without knowing the exact values anymore, the user cannot leak such secret. This means that we need to redesign the structure of the decryption key and hence the algorithm of encryption and decryption. Inspired by Li’s scheme [19], we further bind a unique element $r$ to every component that contains secret value $x_i$. To adapt to the change in user key generation, we designed a new direct revocation mechanism and reconstructed the attribute-related ciphertext. In our revocation mechanism, we enable legitimate users to successfully calculate a necessary but “exclusive” intermediate value when decrypting, where “exclusive” means that this intermediate value is useless to other users. As a result, any part of one user’s decryption key will make no sense in revalidating another revoked user’s decryption key. In addition, once a malicious user discloses a full version of decryption key, the malicious user will be captured by white-box tracing technology and then revoked. A well-established revocation mechanism also enables white-box tracing algorithm to work.

We point out that in the above constructions, it is not clear how to realize life circle of data protection; indeed, these constructions tried to prevent the revoked user from decrypting the data encrypted before revocation (backward secrecy, in other words) and the data encrypted in the future (forward secrecy). As we mentioned above, forward security is vulnerable to user collusion attacks. Since ciphertext update algorithm also focuses on revocation mechanism in this construction, backward secrecy is also threatened by complicity. Furthermore, we observe that if a user computes and saves the intermediate value (e.g., $Y_{uv}$ in [15]) or even $e(g^r, g)^{a_z}$ (that is used to hide a plaintext $m$) before being revoked, the user will also acquire illegal decryption.
capabilities. To achieve life cycle data protection, we redesign the ciphertext update algorithm, which not only updates the ciphertext related to revocation but also removes the hidden risks brought by any intermediate value in the decryption process.

Another challenge lies in that the complete ciphertext update algorithm comes with additional computational overhead to the data owner. For devices with limited processing speed or limited battery power, such as mobile devices and Internet of things devices, this might be a burden that should not be ignored. Considering that the cloud allows users to trade capital expenses (such as data centers and physical servers) for variable expenses, and only pay for services as we consume it, the variable expenses are much lower because of the economies of scale. So, we choose to outsource the ciphertext update algorithm to the cloud. To complete the ciphertext update process, the algorithm will choose a new sharing secret $s' \in \mathbb{Z}_p$ and generate shares of the secret $s'$. Finally, the new secret and the corresponding shares will be embedded into the ciphertext. Note that all operations should be performed on the ciphertext. We carefully design the structure of the ciphertext to ensure that the update can be done without revealing the plaintext.

To further ease the burden on data owners, we split the encryption algorithm into local encryption and re-encryption and outsourced the re-encryption algorithm to the cloud. Compared with Li's scheme [19] and Wang's scheme [37] in which the workload of proxy encryption is assigned to the trusted authority, our scheme better regulates participants’ different roles and eliminates communication overhead.

4.2. Concrete Construction. Now, we present the concrete directly revocable CP-ABE scheme with white-box traceability, collusion resistance, and ciphertext delegation based on [6, 15]. We design a new direct revocation mechanism, manifested in the key generation algorithm, encryption algorithm, and decryption algorithm. In the proposed scheme, the encryption algorithm consists of two steps. The data owner generates a local ciphertext; then, a semi-trusted CSP re-encrypts that local ciphertext and stores the re-encrypted data. To prevent the revoked user from accessing previous ciphertexts and implement revocable storage, the scheme implements an effective ciphertext update algorithm. Our proxy mechanism in re-encryption and ciphertext update algorithms is friendly to resource-constrained data owners. The specific algorithms are described as follows.

**Setup** $(\lambda, U, BT) \rightarrow (APK, ASK)$: TA runs the setup algorithm. It firstly generates a binary tree $BT$ and associates each user in $U$ with a different leaf node in $BT$. Let $R$ be an initially empty revocation list. TA randomly chooses $a, a \in \mathbb{Z}_p$, and $h \in \mathbb{G}$ and picks $U_{\text{att}} \in \mathbb{G}$ for each attribute $\text{att} \in U$. For each node in $BT$, it randomly chooses $X = \{x_i\}_{i=0}^{2^{\text{|leaf|}}-2}$ in $\mathbb{Z}_p$ and then computes $Y = \{y_i\}_{i=0}^{2^{\text{|leaf|}}-2} = g^{x_i^{2^{\text{|leaf|}}-2}}$. Note that in our scheme, $X$ is reserved by the TA and will not be distributed to any user. A probabilistic encryption scheme $(\text{Enc}, \text{Dec})$ [38] is chosen, which is a symmetric encryption from $[0,1]^*$ to $\mathbb{Z}_p$ with secret key $K \in \mathbb{Z}_p$ and encrypts the same message to different ciphertexts each time.

The TA’s secret key is set to

$$ASK = (a, a, X = \{x_i\}_{i=0}^{2^{\text{|leaf|}}-2}, K).$$

The public parameters are published as follows:

$$APK = (g, h, c, (g, g), g^r, \{U_{\text{att}}\}_{\text{att} \in S}, Y = \{y_i\}_{i=0}^{2^{\text{|leaf|}}-2}).$$

**KeyGen** $(ASK, u, S) \rightarrow SK_u$: the decryption key generation algorithm is executed by the trusted authority. Let path $(u) = \{\text{root}, \ldots, u_{\text{att}}\}$ denote the path from the root to the leaf node associated with the user $u$ (namely $u_{\text{att}}$) in the binary tree. $c = \text{Enc}_c (u_{\text{att}})$ is computed, where $K$ is the secret key for the probabilistic encryption, and the result $c$ is not distinguished from a random number in $\mathbb{Z}_p$. $r \in \mathbb{Z}_p$ is randomly chosen, and then, the decryption key component with respect to the attribute set of the user can be constructed as follows:

$$K = g^{u(a+c)}\cdot K^r, \quad L = c, \quad L' = g^{ar}, \quad \{K_{\text{att}} = U_{\text{att}}^{|\text{att}|} r \}_{\text{att} \in S}.$$  

In the presented scheme, because a user $u$ implicitly owns each $x_i$ of the path $(u)$ without knowing the exact values, the user cannot leak such secret. However, this is not enough to resist collusion attacks between users. Since paths of different users might have shared nodes, which means secret $x_i$ might not be exclusive and could allow complicity, we further bind a unique element $r$ to every component that contains $x_i$ where $i \in \text{path}(u)$. As a result, any part of one user’s decryption key will make no sense in revalidating another revoked user’s decryption key. Formally, the decryption key component corresponding to the user’s identity $u$ is set to

$$D = \{d_i | d_i = g^{r(a+c)/x_i} \}_{i \in \text{path}(u)}.$$  

Finally, the above two components form the complete decryption key:

$$SK_u = \langle K, K', L, L', \{K_{\text{att}} \}_{\text{att} \in S}, \{D_i \}_{i \in \text{path}(u)} \rangle.$$  

**LocalEncrypt** $(APK, m, (M, \rho), R) \rightarrow CT$: the data owner runs the algorithm and inputs the public parameter APK, an LSSS access structure $(M, \rho)$, and the current user revocation list $R$, as well as a message $m$ to be encrypted. To complete the local encryption, the algorithm first chooses a random vector $\vec{v} = (s, t_2, t_3, \ldots, t_n)$ to share the encryption exponent $s$ and then generates $l$ shares of the secret $s$ by calculating $\lambda_i = M_i \vec{v}$ for $i = 1, 2, \ldots, l$. The original ciphertext $CT$ corresponding to $(M, \rho)$ and $R$ is computed as follows:

$$C = m \cdot c, \quad C_0 = g^{r}, \quad C_{\text{att}} = g^{s_i}, \quad \{C_i = h^{\lambda_i \cdot U_{\text{att}}^{1/\rho}} \}_{i \in \{1, \ldots, \ell\}}.$$  

$$CT = \langle C, C_0, C_{\text{att}} \}_{i \in \{1, \ldots, \ell\}}, \langle M, \rho, R \rangle.$$  

After the encryption operation, the data owner uploads $CT$ to the cloud service provider.
ReEncrypt\((CT, APK, K) \longrightarrow \tilde{CT}\): the re-encryption algorithm is outsourced to the semi-trusted cloud service provider. In particular, the CSP randomly choose \(k \in \mathbb{Z}_p\) and re-encrypts the local ciphertext \(CT\) as follows:

\[
\tilde{C} = C, \tilde{C}_0 = C_0, \tilde{C}_0' = C_0', [\tilde{C}_i = C_i \times g^k = h^i U^{-1}_p(g)^k]_{i \in \{1, \ldots, t\}}.
\]  

(10)

Let cover \((R)\) be the minimum cover set of the revocation list \(R\). The ciphertext component related to the revocation list \(R\) can be described as follows:

\[
\bar{Y} = \{y_j | y_j = y'_j\} \in \text{cover}(R).
\]  

(11)

Finally, the CSP storages the full ciphertext \(\widetilde{CT}\):

\[
\widetilde{CT} = \langle \tilde{C}, \tilde{C}_0, \tilde{C}_0', [\tilde{C}_i]_{i \in \{1, \ldots, t\}}, [\bar{Y}] \rangle \text{ cover}(R), (M, \rho), R) .
\]  

(12)

Decrypt\((SK_u, \widetilde{CT}) \longrightarrow m\): for a data user who is among the current user revocation list, or whose attribute set does not satisfy the access policy of \(CT\), the algorithm outputs \(\bot\) to indicate a failure of decryption; otherwise, it does the following decryption algorithm.

Suppose \(\text{path}(u) = \{\text{root}, \ldots, u_{ad}\}\), where \(u_{ad}\) is the leaf node associated with the user. For \(u \notin R\), there exists a node \(j\) such that \(j \in \text{path}(u) \cap \text{cover}(R)\). For \(S \in \mathbb{Z}_p\), let \(I = \{i: \rho(i) \in S\} \subseteq \{1, 2, \ldots, \ell\}\), and there exist coefficients \([c_i, i \in I]\) such that \(\sum_{i \in I} c_i M_i = (1, 0, 0, \ldots, 0)\). Thus, we have \(\sum_{i \in I} c_i = s\). The original plaintext \(m\) can be retrieved through these steps:

\[
Z_1 = e(K, C_0^K, C_0^s)\]

\[
eq e(g^{u(a+c)s}, h^r, g^{(a+c)s}),
\]

\[
eq e(g, g)^{as} e(g, h)^{(a+c)s},
\]

\[
Z_2 = \prod_{i \in I} \left(e(L^K, L^r, \tilde{C}_i) \cdot e(K_{\rho(i)}, C_0) \cdot e(\tilde{Y}_j, D_j)^{-1}\right)^{c_i}
\]

\[
= \prod_{i \in I} (e(g, h)^{ar+c^r, c_i}),
\]

\[
eq e(g, h)^{ar+c^r, c},
\]

\[
m = \frac{\tilde{C} \cdot Z_1}{Z_1}.
\]  

(13)

Trace \((SK_u, APK, ASK) \longrightarrow u\) or \(\bot\): taking as input a suspected decryption key \(SK_u\), the tracing algorithm determines whether the decryption key needs to be traced by first verifying whether the \(SK_u\) is well-formed. A decryption key is well-formed if it passes Key Sanity Check [34]. Similar to that in previous works [6, 15], our scheme’s Key Sanity Check is a deterministic algorithm containing the following checks:

1. \(K' \in \mathbb{Z}_p, K, L, L', K_{at}, D \in \mathbb{G}\)
2. \(e(g, L') = e(g', L) \neq 1\)
3. \(\exists \mathcal{A} \in S, s.t. (U_{\mathcal{A}}, L', L') = e(g, K_{at}) \neq 1\)

If the decryption key \(SK_u\) does not pass the Key Sanity Check, the tracing algorithm returns the symbol \(\bot\) implying that \(SK_u\) does not need to be traced. Otherwise, the tracing algorithm outputs an identity \(u\) indicating that \(SK_u\) is linked to the user \(u\). In detail, the tracing algorithm computes \(Dec_G(K')\) to retrieve \(u_{ad}\), the leaf node that points to the malicious user \(u\). Then, it further launches the revoke algorithm to revoke \(u\).

Revoke\((u, R) \longrightarrow R':\) if a user \(u\) is supposed to be revoked, the algorithm adds \(u\) into the revocation list and outputs the latest revocation list \(R' \equiv R \cup u\).

CTUpdate\((\widetilde{CT}, R', APK, K) \longrightarrow \tilde{CT}\): taking the original ciphertext \(\tilde{CT}\), the latest revocation list \(R',\) the public key \(APK\), and the previous update key \(K\) as input, the semi-trusted CSP updates the archetype to a refined \(\widetilde{CT}\). To complete the ciphertext update process, the algorithm randomly picks \(k_t, s_t \in \mathbb{Z}_p\), and lets \(\text{cover}(R')\) be the minimum cover set of revocation list \(R'\). After that, the algorithm chooses a random vector \(\bar{u} \in \mathbb{Z}_p^n\) with \(st\) as the first entry and then generates \(l\) shares of the secret \(s'\) by calculating \(\mu_i = (M \bar{u})_i\) for \(i = 1, 2, \ldots, l\). Ciphertext components related to the revocation list \(R_t\) can be computed as follows:

\[
\tilde{Y} = \{y_j | y_j = y'_j\} \in \text{cover}(R').
\]

(14)

\[
\tilde{C}_i = \tilde{C}_i \cdot h^{\mu_i} \cdot g^{s' + k_t} \cdot U^{-1}_p(g) \cdot g^r_{\rho(i)} \cdot g^s_{\rho(i)} \in \mathbb{G}.
\]

The rest of the ciphertext components are transformed to

\[
\tilde{C}_0 = \tilde{C}_0 \cdot g^{s' + k_t}, \tilde{C}_0' = \tilde{C}_0' \cdot g^{s' + k_t} = g^{s' + k_t}.
\]  

(15)

Finally, the updated ciphertext is kept as follows:

\[
\tilde{CT} = \langle \tilde{C}, \tilde{C}_0, \tilde{C}_0', [\tilde{C}_i]_{i \in \{1, \ldots, t\}}, [\tilde{Y}_j] \rangle \text{ cover}(R'), (M, \rho), R'\).
\]  

(16)

Correctness: suppose \(\text{path}(u) = \{\text{root}, \ldots, u_{ad}\}\), where \(u_{ad}\) is the leaf node associated with the user. For \(u \notin R\), there exists a node \(j\) such that \(j \in \text{path}(u) \cap \text{cover}(R)\). For \(\mathcal{S} \in (M, \rho)\), let \(I = \{i: \rho(i) \in \mathcal{S}\} \subseteq \{1, 2, \ldots, \ell\}\), and there exist coefficients \([c_i, i \in I]\) such that \(\sum_{i \in I} c_i M_i = (1, 0, 0, \ldots, 0)\). Thus, we have \(\sum_{i \in I} c_i = s + st\). The original plaintext \(m\) can be retrieved through these steps:
\[ Z_1 = e \left( K, \overline{C}_0 \cdot \overline{C}_1 \right) \\
= e \left( g^{\alpha(a+c)} h^r, g^{(a+c)(s+z)} \right) \\
= e \left( g, g \right)^{(s+z)a} e \left( g, h \right)^{(s+z)(a+c)r}, \\
Z_2 = \prod_{i \in I} \left( e \left( L^K \cdot L^r, \overline{C}_i \right) \cdot e \left( K, \overline{C}_0 \cdot e \left( Y, D_i \right)^{-1} \right)^{c_i} \right), \\
= e \left( g, h \right)^{(a+c)r c_i}, \\
m = \frac{\overline{c}_i \cdot Z_2}{Z_1} \tag{17} \]

Combined with the revocation mechanism, the proposed scheme achieved both forward secrecy and backward secrecy. That is to say, a revoked user cannot successfully decrypt the ciphertexts that are generated later, not even in the past.

5. Security Analysis

In this section, we first prove the IND-CPA security based on \(q\)-BDHE hardness assumption and then prove the traceability based on I-SDH hardness assumption.

5.1. IND-CPA Security. Since the distribution of the updated ciphertext is identical to the original ciphertext, we only need to prove IND-CPA security related to the original.

**Theorem 1.** If the \(q\)-BDHE hardness assumption holds, no polynomial-time adversary can break the proposed CP-ABE scheme with non-negligible advantage under selective access policy and chosen-plaintext attacks (where \(q \geq 2|\mathbb{U}| - 2\) and \(|\mathbb{U}|\) is the number of user set \(\mathbb{U}\)).

**Proof.** Suppose there exists a polynomial-time adversary \(A\) that can break our CP-ABE scheme under selective access policy and chosen-plaintext attacks with non-negligible advantage \(\varepsilon\), then, we can construct a simulator \(\mathbb{B}\) with a non-negligible advantage to break the \(q\)-BDHE assumption. The simulator \(\mathbb{B}\) runs as follows.

Let \(G\) and \(G_T\) be multiplication cycle groups of prime order \(p\). Let \(g\) be the generator of \(G\). Let \(e: G \times G \rightarrow G_T\) be a bilinear map. \(\mathbb{B}\) flips a fair coin \(\eta \in \{0, 1\}\). Given \(\overline{y} = [g, g^\eta, g^{\eta n'}, \ldots, g^{\eta n'}], \) then if \(\eta = 0\), \(\mathbb{B}\) sets \(Z = e (g, g^{\eta n'})\); otherwise, \(\mathbb{B}\) randomly picks \(\gamma \in G_T\) and sets \(Z = \gamma\).

**Init:** the adversary \(A\) chooses an access structure \((M^*, \rho^*)\) and a revocation list \(R^*\), where \(M^*\) is an \(\ell^* \times n^*\) matrix with \(n^* < q\), and \(\rho^*\) maps rows of \(M^*\) into attributes.

**Setup:** \(\mathbb{B}\) randomly chooses \(a, b \in \mathbb{Z}_p\) and sets \(e(g, g)^a = e(g^\beta, g^\beta) e(g, g)^{\rho^*}\), which implicitly sets \(a = \alpha + d^{\rho^*}\). For each \(\text{att} \in U^*_z\), \(\mathbb{B}\) is randomly chosen and then \(U_{\text{att}}\) is generated as follows:

(i) If there exists an \(i \in \{1, 2, \ldots, \rho^*\}\) such that \(\rho^*(i) = \text{att}\), \(U_{\text{att}} = g^{\rho^*} g^{dM_1} g^{dM_2} \ldots g^{dM_{\rho^*}}\) is set;

(ii) Otherwise, \(U_{\text{att}} = g^{\rho^*}\) is set.

\(\mathbb{B}\) randomly selects \(a \in \mathbb{Z}_p\), computes \(g^a\), and lets \(h = g^a\). For \(\forall i = 0, 1, \ldots, 2|U| - 2\), \(v_i \in \mathbb{Z}_p\) is randomly picked, and let \(y_i = (g^a)^{v_i}\), which implicitly sets \(x_i = d \cdot v_i\). Finally, \(\mathbb{B}\) outputs the public parameters:

\[ \text{APK} = (g, h, e(g, g)^a, g^a, U_{\text{att}})^{U_{\text{att}}}, y_i = (g^a)^{v_i} \text{for } i = 0, 1, \ldots, 2|U| - 2. \] \tag{18}

**Phase 1:** \(A\) submits a set of pairs \((u, S)\) to \(\mathbb{B}\) for petitioning the related decryption keys. To respond, \(\mathbb{B}\) does as follows.

**Case 1.** If \(S \in (M^*, \rho^*)\) and \(u \notin R^*\), then it is aborted.

**Case 2.** If \(S \notin (M^*, \rho^*)\) and \(u \in R^*\), \(c \in \mathbb{Z}_p\) is picked randomly, and \(K, K', L', K_{\text{att}}, D_i\) is computed as follows:

\[ K = g^{\alpha(c/a+c)} g^{dM_1} g^{(a+c)M_{1_2}} = g^{\alpha(a+c)} h^r, \]
\[ L = \left( (g^a)^{(1/a+c)} \right)^{-1} \left( g^{dM_1} g^{(a+c)M_{1_2}} \right) = g^r, \]
\[ L' = (L)^a = g^{ar}, \]
\[ K_{\text{att}} = \left( (g^a)^{v_{\text{att}}} \right)^{-1} \left( g^{dM_1} g^{dM_{1_2}} \right)^{v_{\text{att}}} g^{dM_1} g^{dM_{1_2}}, \]
\[ \cdot \prod_{i=2, \ldots, n'} \left( g^{dM_i} \right)^{v_i} \]
\[ \cdot \prod_{i=2, \ldots, n'} \left( g^{dM_{i+1}} \right)^{v_i} \left( g^{dM_{i+1}} g^{dM_{1_2}} \right)^{v_i} = U_{\text{att}}^{(a+c) r}, \]

which implicitly sets \(r = -d|M|/(a+c) + d^{r-1}/(a+c)\cdot M_{1_2}/M_{1_2}^2\).

Suppose path \(u = [\text{root}, \ldots, u_{j}],\) where root denotes the root node and \(u_{\text{id}}\) is a leaf node associated with the user \(u\). \(\mathbb{B}\) picks \(x_i\), where \(i \in \text{path}(u)\) and computes \(D_i\) as follows:

\[ D_i = \left( g^{dM_1} \right)^{-1} \left( g^{dM_{1_2}} \right)^{v_{\text{att}}} = g^{(a+c)/x_i}. \] \tag{20}

**Case 3.** If \(S \notin (M^*, \rho^*)\) and \(u \in R^*\), \(\mathbb{B}\) computes \(K, K', L', K_{\text{att}}, D_i\) as follows. Given \((M^*, \rho^*)\), there exists a
vector $\overrightarrow{w} = (w_1, w_2, \ldots, w_{n'}) \in \mathbb{Z}_p^{n'}$ such that $w_i = -1$ and $M'_1 \cdot \overrightarrow{w} = 0$ for any $i$ satisfying $\rho^t(i) \in S$. $B$ randomly picks $c, t \in \mathbb{Z}_p$ and sets $K' = c$. After that, $B$ computes

$$K = \left[ g^{a_i} \left( g^{d_{t+1}^j} \right)^t \prod_{i=2, \ldots, n'} \left( g^{d_{t+1}^i} \right)_{w_i} \right]^{1/(a+\epsilon)} = g^{a_l + \epsilon},$$

$$L = \left( g^{d_{t+1}^j} \right)^{1/(a+\epsilon)} \prod_{i=2, \ldots, n'} \left( g^{d_{t+1}^i} \right)_{w_i}^{1/(a+\epsilon)} = g^t,$$

$$L' = \left( g^{d_{t+1}^j} \right)^{a/(a+\epsilon)} \prod_{i=2, \ldots, n'} \left( g^{d_{t+1}^i} \right)_{w_i}^{a/(a+\epsilon)} = g^a,$$

(21) which implicitly sets $r = 1/(a + c)(td + w_1d^t + w_2d^{t-1} + \cdots + w_{n'}d^{t-n'+1})$. Since $S \notin (M^*, \rho^t)$, $B$ computes $K_{\text{att}}$ as follows. For each $att \in S$, if there exists $i$ such that $\rho^t(i) = att$, then the following is computed

$$K_{\text{att}} = L^{(a+\epsilon)} \prod_{j=1, \ldots, n'} \left( g^{d_{t+1}^j} \right)_{w_j}^{1/(a+\epsilon)} \prod_{i=2, \ldots, n'} \left( g^{d_{t+1}^i} \right)_{w_i} = U^{(a+\epsilon)}_{\text{att}},$$

(22)

\begin{align*}
\overrightarrow{Y} &= \{ \overrightarrow{y}_j = (g^{d_{t+1}^j}, y_{j,0}^{k}) \}_{j \in \text{cover}(R')}.
\end{align*}

(27) Finally, $B$ gives the final ciphertext as follows:

$$\overrightarrow{CT} = \{ \overrightarrow{C}_i, \overrightarrow{C}_0, \overrightarrow{C}_i^{r \cdot k}, \{ \overrightarrow{C}_i \}_{i \in \{1, \ldots, d\}}, \{ \overrightarrow{Y} \}_{j \in \text{cover}(R')} \}, \{ M^*, \rho^t \}, R' \}.$$  

(28)

Phase 2: it is same as Phase 1.

**Guess:** the adversary $A$ will eventually output a guess $\beta'$ of $\beta$. If $\beta' = \beta$, the simulator $B$ outputs $\eta = 0$ to indicate that $Z = e(g, g)^{d_{t+1}^j}$; otherwise, it outputs $\eta = 1$ to indicate that $Z$ is a random group element in $\mathbb{G}_T$.

In the case where $\eta = 0$, the adversary $A$ obtains $Z = e(g, g)^{d_{t+1}^j}$. Supposing the adversary’s advantage is $\epsilon$, then the probability that $A$ wins the game is $\Pr[\beta' = \beta | \eta = 0] = 1/2 + \epsilon$. Since the simulator outputs $\eta = 0$ when $\beta' = \beta$, there exists $\Pr[\eta = 0 | \eta = 0] = \Pr[\beta' = \beta | \eta = 0] = 1/2 + \epsilon$.

In the case where $\eta = 1$, the adversary $A$ obtains $Z = g$, which is a random element in $\mathbb{G}_T$, such that the adversary has no information about $\beta$ and loses the advantage. We get $\Pr[\eta = 0 | \eta = 1] = \Pr[\beta' = \beta | \eta = 1] = 1/2$.

In the end, the advantage of $B$ to solving $q$-BDHE hardness assumption is as follows: $\Pr[\eta = 1] = \Pr[\eta = \eta | \eta = 0] \cdot \Pr[\eta = 0] + \Pr[\eta = \eta | \eta = 1] \cdot \Pr[\eta = 1] = 1/2 = (1/2 + \epsilon) / 2 = 1/2 + \epsilon$.

5.2. Traceability. Based on Liu’s proof method [15], we reduce the traceability of our CP-ABE scheme to the I-SDH hardness assumption. Note that our work in this subsection is mainly to illustrate that the reconstructed key maintains traceability instead of proposing new proof tools. So, we basically follow the approach in [15]. The specific analysis can be found in appendix.

5.3. Forward and Backward Secrecy. Once a user is revoked, all of the components encrypted with a secret element $s$ in the ciphertext are updated by the semi-trusted CSP with a
new secret \( s \), and the components related to revocation list are also renewed. Even if the revoked user has stored the previous partial decryption result \( e(g, g)^{as} \), it would not help to recover the desired value \( e(g, g)^{a(s^t s_\ell)} \). Thus, the backward secrecy of the updated ciphertext is guaranteed in the proposed scheme.

In the proposed CP-ABE, the data owner will embed the latest revocation list into the ciphertext in the process of local encryption. For a data user who is included in the revocation list, his/her decryption key cannot satisfy the access control policy of subsequent ciphertexts anymore; as a result, the backward secrecy is guaranteed.

### 6. Performance Analysis

In this section, the proposed scheme is compared in terms of functionality and performance with several related works.

As shown in Table 1, the schemes of Ning et al. [6], Liu et al. [15], Sethi et al. [27], Han et al. [22], and this proposal all provide traceability, but Ning et al.'s scheme [6] does not sustain a revocation mechanism, and it cannot revoke a malicious user to ensure security even if that malicious user is traced. Meanwhile, [27] adopts an indirect revocation mechanism and cannot revoke a malicious user timely. Since a malicious user is desired to be revoked immediately, a direct revocation mechanism is more suitable for a CP-ABE with traceability. The schemes of Liu et al. [15], Xu et al. [28], Sethi et al. [27], Han et al. [22], and this proposal try to support the ciphertext update when revocation occurs. The difference is that neither [15] nor [22] can resist user collusion attacks among users, and they cannot provide sufficient forward and backward secrecy. Moreover, Table 1 shows that schemes [6, 15, 22, 28] and ours are proved to be secure under a selective access policy against chosen-plaintext attacks in the standard model. Sethi et al.'s scheme [27] is proved statically [39] secure in the random oracle model. Generally, the standard model is more realistic than the random oracle model.

In Table 2, our scheme's efficiency is compared against several related works. Note that [27] supports multi-authority functionality; here, we make comparisons in a single-authority scenario without reducing its efficiency. Our scheme's complete encryption phase period consists of two parts. One is the LocalEncrypt algorithm performed by a DO, and the other is the ReEncrypt algorithm executed by a CSP, thereby reducing computing overhead for the DO by outsourcing the latter algorithm to the CSP. In the LocalEncrypt algorithm, our scheme is slightly better than previously published schemes [15, 22] and is significantly more efficient than schemes [6, 27, 28].

When we take the ReEncrypt algorithm into consideration, both the exponentiation operations and multiplication operations of our complete encryption process are less than that in [6, 22]. Compared with [15], our complete encryption process has one more exponentiation operation and \( \ell \) additional multiplication operations. Since multiplication is not as costly as exponentiation, it will not bring unacceptable overhead. A more intuitive illustration will be presented in the following experimental analysis. Our scheme is still competitive after accounting for computations of the ReEncrypt algorithm. Compared with these direct revocable schemes [15, 22], the computational effort of our KeyGen algorithm is related to the depth of the binary tree. We note that it is acceptable because we only need to do it once for a user. For those indirect revocable schemes [27, 28], an update key generation (UpdateKeyGen) algorithm is required in addition to the basic KeyGen algorithm, which brings additional overhead. As for the overhead of the Decrypt algorithm, our scheme is similar to that in schemes [15] and is more efficient than [6, 22, 27]. The computation of the Decrypt algorithm in [28] is further affected by the length of the binary representation of a valid period. For the CTUpdate algorithm, our solution performs better than that in [27, 28] but is less efficient than [15, 22]. Our CTUpdate algorithm does bring more computation overhead. With these efforts, our scheme realizes that user collusion attacks resistance and guarantees sufficient backward secrecy, whereas schemes [6, 15, 22] do not hold such functionality. The user collusion resistance is necessary for traceable and revocable CP-ABE. A malicious user in [15, 22] can launch such attacks to illegally revalidate the decryption key without being caught by the trace mechanism. A CP-ABE scheme with user collusion resistance will be more effective and practical. Schemes [27, 28] also achieve backward secrecy by a ciphertext update algorithm. On the premise of meeting the backward secrecy, our ciphertext update algorithm is more efficient among these existing works. Furthermore, as the complete workload of the CTUpdate algorithm is outsourced to the semi-trusted CSP, data owners do not need to worry about such computational pressure.

For the schemes listed in Table 2, Figures 2(a)–2(e) depict the time cost comparisons on KeyGen, Encrypt, Decrypt, Trace, and CTUpdate algorithms. The experiment was carried out on a Linux system with an Intel Core i7-10750H CPU @2.60 GHz and 4.00 GB RAM. To build the code, we used PBC-0.5.14 Library, Charm-Crypto 0.50 framework, and PyCharm Community 2021.1 IDE.

We take Figure 1 as the binary tree in the experiment, and by supposing the previous user revocation list \( R = \{u_3\} \),

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Traceability</th>
<th>Revocation</th>
<th>Ciphertext updating</th>
<th>User collusion resistance</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>[6]</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>Selective</td>
</tr>
<tr>
<td>[15]</td>
<td>√</td>
<td>Direct</td>
<td>√</td>
<td>×</td>
<td>Selective</td>
</tr>
<tr>
<td>[22]</td>
<td>√</td>
<td>Direct</td>
<td>√</td>
<td>×</td>
<td>Selective</td>
</tr>
<tr>
<td>[27]</td>
<td>√</td>
<td>Indirect</td>
<td>√</td>
<td>√</td>
<td>Static</td>
</tr>
<tr>
<td>[28]</td>
<td>×</td>
<td>Indirect</td>
<td>√</td>
<td>√</td>
<td>Selective</td>
</tr>
<tr>
<td>Ours</td>
<td>√</td>
<td>Direct</td>
<td>√</td>
<td>√</td>
<td>Selective</td>
</tr>
</tbody>
</table>

√ denotes that the scheme has this functionality. × denotes that the scheme does not have this function.
Table 2: Efficiency comparisons.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>KeyGen</th>
<th>UpdateKeyGen</th>
<th>LocalEncrypt</th>
<th>ReEncrypt</th>
<th>Decrypt</th>
<th>Trace</th>
<th>CTUpdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>[6]</td>
<td>(5+3\ s)E+(1+2\ s)M</td>
<td>—</td>
<td>(3+5\ ℓ)E+(1+2\ ℓ)M</td>
<td>—</td>
<td>(1+3\ n)P+(2+n)E+(3+3\ n)M</td>
<td>(5+3\ s)P+(2+s)E+(3+2\ s)M</td>
<td>—</td>
</tr>
<tr>
<td>[15]</td>
<td>(7+s)E+2M</td>
<td>—</td>
<td>(3+2\ ℓ+r)E+(1+ℓ)M</td>
<td>—</td>
<td>(3+2\ n)P+(3+n)E+(6+2\ n)M</td>
<td>(2+2\ s)P+E+M</td>
<td>tE</td>
</tr>
<tr>
<td>[22]</td>
<td>(6+s)E+(1+s)M</td>
<td>—</td>
<td>(3+4\ ℓ+r)E+(1+ℓ)M</td>
<td>—</td>
<td>(2+3\ n)P+(3+n)E+(4+3\ n)M</td>
<td>(6+s)P+(2+s)E+(3+s)M</td>
<td>tE</td>
</tr>
<tr>
<td>[27]</td>
<td>(3+s+d)E+sdM</td>
<td>—</td>
<td>(1+v+9\ ℓ+vℓ)M</td>
<td>—</td>
<td>4nP+4nE+8nM</td>
<td>(3+s+sd)P+3E+(4+2sd)M</td>
<td>(1+v+v'+9ℓ+vℓE)+(2+8ℓ+v+2v')M</td>
</tr>
<tr>
<td>[28]</td>
<td>(3+3\ s+d)E+(2\ v+d)M</td>
<td>—</td>
<td>(2+5\ ℓ+5\ v)M</td>
<td>—</td>
<td>P+(n+v)E+(1+3\ n+3+3\ v)M</td>
<td>(2+5\ ℓ+5\ v)E+(2+5\ ℓ+5\ v)M</td>
<td>—</td>
</tr>
<tr>
<td>Ours</td>
<td>(4+s+d)E+M</td>
<td>—</td>
<td>(3+2\ ℓ)E+(1+ℓ)M</td>
<td>(1+r)</td>
<td>E+ℓM</td>
<td>(2+2\ n)P+(2+n)E+(3+3\ n)M</td>
<td>(2+2\ s)P+E+M</td>
</tr>
</tbody>
</table>

s denotes the number of user attributes. ℓ denotes the number of attributes in the access structure. n denotes the number of attributes in the decryption key that satisfy the access structure. r denotes the number of nodes in cover (R). d denotes the depth of the binary tree. v denotes the length of a valid period formatted in binary representation [27, 28]. v' denotes the hamming distance between the binary strings of two different periods in [27]. \( t = \sum_{j\in cover} (depth (j) - 1 - depth(j)) \) denotes the ciphertext affected by revocation. ED denotes an exponentiation operation in \( \text{G} \) and \( \text{GT} \). M denotes a multiplication operation in \( \text{G} \) and \( \text{GT} \). P denotes a bilinear pairing operation.
we have cover(R) = {1, 6, 12}. Then, it is assumed that the newly captured malicious user is u1, which means R′ = {u1, u2} and cover(R′) = {4, 6, 8, 12}. In our experiment, the number of user attributes ranges from 10 to 50 with a step of 10. The final result is the mean of 100 replicates. In addition, since schemes [27, 28] introduce a valid period formatted in binary representation but do not specify the length of the binary string, we restrict it to 1 for simplicity.

Figure 2(a) shows that the KeyGen time of the proposed scheme is highly approximate to that of schemes [15, 22], and all of those two schemes and ours are significantly more efficient than schemes [6, 27, 28]. To make a clear description, we present the overhead of LocalEncrypt and ReEncrypt algorithms simultaneously in Figure 2(b). Our scheme is still competitive with these schemes after accounting for computations of the ReEncrypt algorithm. Figures 2(c) and 2(d) present the data of decrypt time and trace time, respectively, and the results indicate that our scheme is more efficient in both the Decrypt algorithm and Trace algorithm. Figure 2(e) shows that both [15, 22] have higher ciphertext updating efficiency, but the CTUpdate algorithm in these two schemes is limited by their revocation mechanisms and cannot resist a revoked user who preserves the intermediate values (e.g., Yid in [15]) or launches user collusion attacks. The other two schemes [27, 28] and ours do not suffer from such problems. On the premise of meeting the security requirement, our CTUpdate algorithm is more efficient among them. In general, the simulation results agree with the above theoretical analysis.

7. Related Work

Sahai and Waters [2] were the first of many investigators to introduce attribute-based encryption (ABE). In such an attribute-based encryption system, different from the case in traditional public key cryptography, there is no need to prepare exclusive ciphertexts for various users. The advent of ABE brings a unique “one-to-many” public key encryption capability that enables fine-granted access control over encrypted data.

In subsequent studies, to meet various functional requirements, ABE schemes have become increasingly complicated. Revocation is a critical bottleneck limiting its practical application. Given a cloud storage system based on a primitive class of CP-ABE scheme, the case where a malicious user wants to sell his/her decryption key for profit should be considered. Since those attributes associated with the mentioned decryption key might be shared by multiple users with no unique information indicating the holder’s identity, the malicious user can avoid being held accountable. To address this issue, traceable CP-ABE schemes [5, 6]...
in which whoever leaks a decryption key can be identified were proposed.

Improving traceability, though it might identify the leakage, does not guarantee a solution to the problem, of course, since it does not necessarily lead to removing the malicious user. ABE schemes require not only tracing a defected user but also revoking him/her from the system.

Li et al. [7] presented a CP-ABE, which enables directly revoking corrupted users. Liu and Wong [40] gave constructions of both KP-ABE and CP-ABE that sustain revocation. Liu et al. [14] realized direct user revocation mechanisms by embedding a valid period value and a revocation list into the ciphertext. Shi et al. [41] proposed a novel KP-ABE scheme, in which outsourced encrypted data can be updated by an untrusted third party and those revoked users cannot decrypt the prior ciphertexts. Inspired by [5, 41], Liu et al. [15] constructed a CP-ABE scheme with white-box traceability and user revocation that allows, once a malicious user is traced, a user’s identity will be added to a revocation list; when combined with a ciphertext update algorithm, backward secrecy is achieved as the revoked user is denied access to the previously encrypted data. Han et al. [22], focusing on protecting access policy, proposed a CP-ABE scheme that realizes white-box traceability, revocation, and the application of hidden policy. However, both schemes of Liu et al. [15] and Han et al. [22] suffer from collusion attacks between users.

User collusion resistance and backward secrecy are inherent required but easily overlooked properties of a secure CP-ABE system. Li et al.’s scheme [7] supported user collusion resistance, but did not realize backward secrecy. Zhang et al. [42] showed an effective white-box traceable ABE scheme that can resist user collusion attacks, but it lacked a revocation mechanism. Li et al. [19] tried to address collusion attacks executed by the existing users cooperating with those revoked users with a scheme achieving indirect revocation by binding together the user’s decryption key and group secret key. The downside in Li et al.’s scheme [19] was that the trusted authority bore the burden of re-encryption and updating the decryption key whenever revocation was required. For a system with a large number of users, communication protocol and computing overhead would be equally large.

8. Conclusion

In this work, we presented a directly revocable CP-ABE scheme with user collusion resistance, white-box traceability, and both forward and backward secrecy. Once a malicious user is revoked, he/she will lose access to subsequent and previous encrypted data. The scheme also prevents complicity between a revoked user and a legitimate user in the system. Any component of another user’s decryption key will make no sense in revalidating the revoked user’s decryption key, and even if a malicious user chooses to leak the full decryption key, the malicious user could be captured by white-box tracing technology. In addition, using an effective proxy re-encryption mechanism and an outsourced ciphertext update algorithm, the scheme reduces the computational burden on data owners and cuts down the communication overhead between data owners and cloud service providers, respectively. The performance analysis showed that our proposal is better in functionality and competitive in efficiency compared with existing schemes. Further, our scheme was proved selectively secure against chosen-plaintext attacks in the standard model.

Appendix

Traceability Analysis

Theorem A1. If the l-SDH hardness assumption holds, the proposed CP-ABE scheme is traceable with \( q < l \), where \( q \) is the number of key queries.

Proof. Suppose there is a polynomial-time adversary \( A \) that can win the traceability game with non-negligible advantage \( \epsilon \) under \( q \) key queries. Here, we let \( l = q + 1 \); then, we can construct a simulator \( B \) that can have a non-negligible advantage to break the l-SDH assumption. To utilize \( A \) to solve the challenging problem, the simulator \( B \) runs as follows

Let \( G \) and \( G_r \) be multiplication cycle groups of prime order \( p \), and \( g \) be the generator of \( G \). Let \( e: G \times G \rightarrow G_r \) be a bilinear map. \( B \) obtains an l-SDH problem \( (g_1, g_1^a, g_1^b, \ldots, g_1^{q+1}) \) and aims to output a tuple \( (c_r, w_r, g_1^{1/(a+c_r)}) \), where \( g_1 \in G, a \in \mathbb{Z}_p^* \).

Setup: let \( A_i = g_i^a \) for \( \forall i \in \{0, 1, \ldots, l\} \). \( B \) randomly picks \( q \) different elements \( c_1, c_2, \ldots, c_q \in \mathbb{Z}_p^* \), \( a \in \mathbb{Z}_p^* \), and \( h \in G \). Then, let \( f(y) \) be the polynomial described as follows:

\[
f(y) = \prod_{i=1,2,\ldots,q} (y + c_i) = \sum_{i=0,1,\ldots,q} a_i y^i,
\]

where each \( a_i \in \mathbb{Z}_p^* \) is a coefficient of the polynomial \( f(y) \).

Let \( g = \prod_{i=0}^{q} (A_i)^{b_i} = g_1^{f(a)} \) and \( g^a = \prod_{i=1}^{q+1} (A_i)^{c_{q+1-i}} = g_1^{f(a)+a} \). For \( \forall att \in U \), \( u_{att} \in \mathbb{Z}_p^* \) is randomly picked, and let \( U_{att} = g^{u_{att}} \). For each node in the binary tree \( BT^* \), \( x_i \in \{0, 1\}^{l+1} \in \mathbb{Z}_2^* \) is randomly picked, and then, let \( y_i = g_1^{x_i^{2^{l-2}}} \).

Finally, \( A \) is given the public parameters

\[
APK = (g, h, e(g, g)^a, g^a, U_{att} = U, Y = \{y_i\}_{i=0}^{2^{l+1}-2} = g_1^{x_i^{2^{l+1}-2}}).
\]

Key Query: \( A \) submits \( (u_i, S_i) \) to \( B \) for petitioning the related decryption key. To respond, \( B \) proceeds as follows.

Note that \( i \leq q \) and when it goes on the \( i \)th query, let \( f_i(y) \) be the following polynomial

\[
f_i(y) = \frac{f(y)}{y + c_i} = \prod_{j=1,2,\ldots,q \neq i} (y + c_j) = \sum_{j=0,1,\ldots,q-1} \beta_j y^j,
\]

where \( \beta_j \in \mathbb{Z}_p^* \) and \( j = 0, 1, \ldots, q - 1 \) are the coefficients of polynomial \( f_i(y) \). Then, \( B \) sets


\[ s_i = \prod_{j=0,1,\ldots,q-1} (A_j)^{\delta_j} = g_{f_1}^{f(a)} = g_{f_1}^{(a)(\alpha+c)} = g_{1/(\alpha+c)}. \]  

(A4)

Suppose path \((i_d) = i_0, \ldots, i_d\), where \(i_0 = \text{root} \) and \(i_d\) is the leaf node related to \(u_i\). To calculate the decryption key related to \((u_i, S_i)\), \(B\) randomly picks \(r \in \mathbb{Z}_p\) and does as follows:

\[ K' = c_i, K = (s_i) \cdot H' = g_{\alpha/(\alpha+c)} H', \quad L = g_{f_2} L' = (g_{Rf} H')', \quad g_{f_2} \cdot y_{i_0} = U_{i_0}^{(a+c)} |_{\text{attr}S_i}, K_i = g_{f_2} |_{\text{attr}S_i} |_{\text{path}(i_d)}. \]

Finally, \(B\) sends the decryption key below to \(A\):

\[ SK_{u_i} = \langle K, K', L, L', |\text{attr}S_i, |D_i |_{\text{path}(i_d)} \rangle. \]  

(A5)

**Key Forgery:** \(A\) submits a forged decryption key \(SK_{u_i}\) to \(B\). Note that distributions of public parameters \(APK\) and decryption key \(SK_{u_i}\) in the above game are identical to the real scheme.

Let \(\epsilon_{SDH}\) denote the event that \(A\) wins the game; i.e., \(SK_{u_i}\) can pass the Key Sanity Check and \(K' \notin \{c_1, c_2, \ldots, c_q\}\).

If \(\epsilon_{SDH}\) does not happen, \(B\) chooses a random pair \((c_i, u_i) \in \mathbb{Z}_p \times G\) as the solution to the SDH problem.

If \(\epsilon_{SDH}\) happens, using long division \(B\) writes the polynomial \(f(y) = y(y + K) + y_\ldots + 1\) for some polynomial \(y = \sum_{i=0}^{q-1} y_i y^i\) and some \(y_\ldots + 1 \in \mathbb{Z}_p^\ast\). Since \(f(y) = \prod_{i=1}^q (y + c_i)\), \(c_i \in \mathbb{Z}_p^\ast\), and \(K' \notin \{c_1, c_2, \ldots, c_q\}\), as such \(y + K'\) cannot divide \(f(y)\). \(B\) computes a tuple \((c_i, u_i) \in \mathbb{Z}_p \times G\) as follows.

Suppose that \(L = g_{f_2}\), where \(r\) is unknown. According to the equality \(e(L', g) = e(L, g_{f_2})\) from Key Sanity Check, we have \(L' = g^r\). On the basis of the equality \(e(L, K' \Lambda) = e(L, g^r)\), we get \(\Lambda = g^r/(\alpha+c)\). Then, \(B\) computes

\[ \sigma \]

\[ \epsilon_{SDH} = \left\{ \sum_{i=0,1,\ldots,q-1} A_i^{-y_i} \right\} g_{f_2}^{1/(\alpha+c)}, \]  

(A6)

As \(e(g_{f_2}^i, w_i, r) = e(g_{f_2}^i, g_{f_2}^{-1}, g_{f_2}^{1/(\alpha+c)}) = e(g_{f_2}^i, g_{f_2}^{-1})\), the tuple \((c_i, u_i)\) is a solution to SDH hard problem.

Let \(\epsilon_{SDH}(c_i, u_i)\) denote the event in which \((c_i, u_i)\) is a solution for the SDH problem, which can be verified by checking whether \(e(g_{f_2}^i, c_i, u_i) = e(g_{f_2}^i, g_{f_2}^{-1}, g_{f_2}^{1/(\alpha+c)})\) happens. When \(B\) randomly selects a tuple \((c_i, u_i)\), \(\epsilon_{SDH}(c_i, u_i)\) happens with negligible probability and denotes 0 for simplicity. When it comes to the case that \((\sigma\text{wins} \land \gcd(y_\ldots + 1, p) = 1)\) happens, the probability of \((c_i, u_i)\) satisfying \(e(g_{f_2}^i, c_i, u_i) = e(g_{f_2}^i, g_{f_2}^{-1})\) is 1.

Hence, \(B\) solves the SDH problem with probability

\[ \Pr[\epsilon_{SDH}(c_i, u_i)] + \Pr[\epsilon_{SDH}(c_i, u_i) \land \sigma\text{wins} \land \gcd(y_\ldots + 1, p) \neq 1] \cdot \Pr[\sigma\text{wins} \land \gcd(y_\ldots + 1, p) \neq 1] = 0 + 0 + 1 \cdot \Pr[\sigma\text{wins} \land \gcd(y_\ldots + 1, p) = 1] = \Pr[\sigma\text{wins}] \cdot \Pr[\gcd(y_\ldots + 1, p) = 1] = \epsilon_{SDH}. \]  

(A7)

**Data Availability**

The data used to support the findings of this study are included within the article.

**Conflicts of Interest**

The authors declare that there are no conflicts of interest regarding the publication of this study.

**Acknowledgments**

This work was supported by National Key R&D Program of China (grant no. 2020YFB1806205) and Research Program of BJCA (BJCA2020-YF-0300).

**References**


