IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 21, 2021, accepted February 8, 2021, date of publication March 9, 2021, date of current version March 23, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3064863

A Novel Revocable and Identity-Based Conditional Proxy Re-Encryption Scheme With Ciphertext Evolution for Secure Cloud Data Sharing

SHIMAO YAO[1,2], RALPH VOLTAIRE J. DAYOT[2], HYUNG-JIN KIM[3], AND IN-HO RA[2], (Member, IEEE)

[1]School of Computer and Big Data Science, Jiujiang University, Jiangxi 332005, China
[2]School of Computer, Information and Communication Engineering, Kunsan National University, Gunsan 54150, South Korea
[3]Department of IT Applied System, Chonbuk National University, Jeonju 54896, South Korea

Corresponding authors: Hyung-Jin Kim (kim@jbnu.ac.kr) and In-Ho Ra (ihra@kunsan.ac.kr)

This work was supported in part by the National Natural Science Foundation of China under Project 61662039, Project 62062045, and Project 62041603, in part by the KETEP, Korean Government, Ministry of Trade, Industry, and Energy (MOTIE) under Project 20194010201800, and in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) under Project 2021R1A2C2014333.

ABSTRACT Proxy re-encryption (PRE), with the unique ciphertext transformation ability, enables various ciphertext authorization applications to be implemented efficiently. However, most existing PRE schemes mainly focus on access authorization while ignoring the situation where the key needs to be changed and the ciphertext needs to be evolved, making the scheme's practicability and security inadequate. Moreover, the few schemes that simultaneously combine ciphertext authorization, key update, and ciphertext evolution are not satisfactory in terms of security. For solving this problem, based on Xiong *et al.*'s scheme, this paper proposes an improved revocable and identity-based conditional proxy re-encryption scheme with ciphertext evolution (RIB-CPRE-CE) for secure and efficient cloud data sharing. The proposed scheme inherits the characteristics of multi-use, constant ciphertext length, fine-grained authorization, collision-resistance security, and chosen ciphertext attack (CCA) security from the original method. Also, it supports updating ciphertext to adapt to the new key after changing the identity (key) or achieves authorization revocation by evolving ciphertext. Two new algorithms, *URKeyGen* and *UpReEnc*, have been integrated into the original delegation scheme to support ciphertext evolution. The formal definition, security model, concrete construction, and security analysis of RIB-CPRE-CE have been presented. The comparison and analysis show that the proposed scheme is practical and secure. Although it adds a ciphertext evolution function for supporting key update and delegation revocation, its efficiency and security are not reduced. The proposed scheme can also be used in other access authorization systems that need to change the key or revoke the authorization. It has certain practicability and security.

INDEX TERMS Ciphertext evolution, cloud sharing, conditional re-encryption, identity-based proxy re-encryption, key update, revocable.

I. INTRODUCTION

Proxy re-encryption (PRE) enables a semi-trusted proxy to convert a ciphertext encrypted under one user's public key into a new ciphertext that can be decrypted by another user's private key without revealing the underlying plaintext, which is considered a promising solution for efficiently and securely delegating data access among users [1]. Since the first scheme [2] proposed by Blaze *et al.* in 1998, PRE has attracted much attention, and many schemes [3]–[11] with various features (as shown in Table 1) have been proposed. PRE has been widely used in encrypted email forwarding [2], [3],

The associate editor coordinating the review of this manuscript and approving it for publication was Fan-Hsun Tseng.

TABLE 1. Some schemes with various characteristics.

Schemes	Bidirectional	Multi-use	Security	Collusion-resistant	Identity-based encryption	Random oracle	Conditional
[2]	Yes	Yes	CPA	No	No	No	No
[3]	No	No	CPA	Yes	No	No	No
[4]	Yes	Yes	CCA	No	No	No	No
[5]	No	No	RCCA	Yes	No	No	No
[6]^a	Yes	Yes	CPA	Yes	Yes	Yes	No
[6]^b	Yes	No	CCA	Yes	Yes	Yes	No
[7]	No	Yes	CCA	No	Yes	No	No
[8]	No	Yes	CCA	Yes	Yes	No	No
[9]	No	No	CCA	Yes	No	Yes	Yes
[10]^a	Yes	Yes	CPA	Yes	No	No	No
[10]^b	Yes	No	CCA	Yes	No	No	No
[11]	No	Yes	CPA	Yes	Yes	No	No

digital rights management [12], cloud data sharing [13], and other delegation occasions [14], [15]. Being consistent with the original intention of PRE, the existing schemes mainly focus on access authorization, including single-hop authorization or multi-hop (multi-use) authorization, unidirectional authorization or bidirectional authorization, public key-based authorization or identity-based authorization, conditional authorization or universal authorization. If the PRE scheme is used for short-time delegation situations (e.g., encrypted mail forwarding), it may be sufficient to consider authorization only since the probability that the key needs to be changed over a short period is negligible. But for a long-time delegation application such as secure cloud data sharing or digital rights management, it will not be practical and secure enough without considering the delegation change caused by key change.

According to Special Publication 800-57 [16], issued by the National Institute of Standards and Technology (NIST), any scheme based on cryptography must periodically update keys. In addition, for a cryptography application, once the user's key has been compromised, it must be replaced immediately. And after the key changes, the old ciphertext must be deleted or updated to avoid the adversary decrypting it with the old key. Therefore, considering functions of the key update, authorization revocation, and ciphertext evolution in the secure data sharing scheme using PRE for delegation is necessary. However, as far as we know, most of the existing PRE schemes or applications only consider authorization and do not include key change. Also, the few schemes [17], [24], [25] that consider both authorization and key update are not ideal for security and practicality. The ciphertext length in Liang *et al.*'s scheme [17] grows with the re-encryption number, making the scheme unpractical. The re-keying algorithm in Shafagh *et al.*'s scheme [24] computes the re-encryption key as $rk = x'/x$, where x' and x are the new key and old key, respectively. This re-encryption key generation algorithm cannot resist collusion attack as the adversary who knows the old key x and the re-encryption key rk can derive the new key x', which will make the key update meaningless. Yao *et al.*'s scheme [25] is proved secure under the random oracle model but maybe insecure in a real application. Moreover, the above three schemes are only secure against chosen-plaintext attack (CPA), but cloud data sharing application usually needs to be secure against chosen-ciphertext attack (CCA). Thus, this motivates us to propose a CCA-secure PRE scheme that supports key update and ciphertext evolution for cloud data sharing.

From the perspective of ciphertext transformation, ciphertext evolution is also a process of transforming a ciphertext encrypted by one user's public key into a new ciphertext that another private key can decrypt, which coincides with proxy re-encryption. Therefore, intuitively, a ciphertext update function could be added directly to the existing PRE delegation scheme. However, it is essential to note that this extension is not trivial because the update function needs to be perfectly integrated with the authorization function. Otherwise, the authorization function will be affected, so not all PRE schemes would be suitable. First of all, it is strongly requested that the scheme must be multi-hop to ensure that the updated ciphertext can be updated again, so the single-hop scheme is not suitable. Secondly, for the sake of efficiency and practicability, the length of ciphertext and the complexity of decryption cannot be increased linearly with the number of re-encryption times. In this aspect, the multi-hop schemes [2], [7], [8], [17] based on the GA (Green and Ateniese [6]) paradigm are not suitable due to the increment of ciphertext length. Only the scheme of which ciphertext length does not increase with re-encryption is appropriate. Third, the scheme also needs to ensure the security of collusion-resistance. Because in the case that the user's key is compromised or breached and needs to be updated, the adversary knows the previous key, and the re-encryption key for updating ciphertext can deduce the new key in collaboration with the proxy, which will make the key update and ciphertext evolution meaningless. Finally, the CCA security is a critical evaluation criterion for cryptography schemes. The chosen scheme should be CCA-secure.

To sum up, a PRE scheme that supports key update and ciphertext evolution should have the characteristics of multi-hop, a constant ciphertext length, collusion-resistance, and CCA security. To the best of our knowledge, only two schemes meet the above requirements so far. One is the multi-hop CCA-secure PRE scheme proposed by Lai *et al.* [18] using the recent advances in indistinguishability obfuscation.

The other is the secure cloud access authorization scheme in cloud computing presented by Xiong *et al.* [19]. Since the former does not give an efficiency analysis, its practicability needs to be verified. So, the latter is chosen as the underlying scheme to be improved.

A. CONTRIBUTIONS

The major contributions of this paper are as follows:

1) To improve Xiong *et al.*'s scheme [19] by adding a ciphertext evolution function to support key change and delegation revocation. Also, a revocable and identity-based conditional proxy re-encryption scheme with ciphertext evolution (RIB-CPRE-CE) for more secure and efficient cloud data sharing is proposed.

2) To integrate two new algorithms, *URKeyGen* for generating update re-encryption key and *UpReEnc* for evolving ciphertext, into the original delegation scheme.

3) To give a formal definition of RIB-CPRE-CE and its security model.

4) To give a concrete construction and prove its CCA-security in the standard model under the Decisional Bilinear Diffie-Hellman assumption.

B. RELATED WORK

Since the concept of cloud computing was put forward, ensuring its data security has always been the focus of research. Many researchers have proposed a large number of security solutions to suit various application scenarios (requirements). The traditional symmetric encryption algorithms (i.e., AES), hybrid encryption algorithm, or asymmetric encryption algorithm (i.e., RSA) fails to meet user expectations on ciphertext optimality and key optimality. For example, a data owner (Alice) only wants to store one copy of ciphertext in the cloud server, which can be accessed by multiple users (Alice and Bob), and the data visitor (Alice or Bob) only needs her/his own private key to access the encrypted data. They do not want to keep any additional decryption keys. The technology that can gracefully address this challenge is proxy re-encryption. At present, there are a large number of schemes using proxy re-encryption to achieve secure data sharing. But here, only the PRE schemes with properties are reviewed.

Blaze *et al.* [2] proposed the first bidirectional multi-hop proxy re-encryption scheme with a constant ciphertext length, which cannot resist collusion attack and chosen-ciphertext attack. Canetti and Hohenberger [4] realized the CCA security based on the Blaze's scheme but cannot resist collusion attack either. Weng and Zhao [10] proposed a bidirectional multi-hop PRE scheme with constant ciphertext length and anti-collusion security, but it is only CPA-secure. Luo *et al.* [11] proposed an identity-based proxy re-encryption scheme with the characteristics of constant ciphertext length, unidirectionality, multi-hop, and master secret security, but it is also CPA-secure. Liang *et al.* [20] proposed an identity-based conditional PRE scheme with constant ciphertext length, and it is secure against collusion attack and chosen-ciphertext attack. However, He *et al.* [21] pointed out that their scheme cannot ensure the CCA-security by giving specific attacks. Recently, Xiong *et al.* [19] proposed a flexible, efficient, and secure access authorization scheme in cloud computing based on Luo *et al.*'s scheme [11]. Their scheme also has the characteristics of unidirectionality, multi-hop, constant ciphertext length, collusion-safe, and CCA-secure, and supports fine-grained access authorization. Lai *et al.* [18] used the recent advances in indistinguishability obfuscation to propose a unidirectional, multi-hop, constant ciphertext length PRE scheme that is secure against collusion and chosen-ciphertext attack. They claimed that the scheme could be used in an identity-based encryption infrastructure. But the solution did not provide a comparative analysis of the implementation and efficiency on the proxy server, and its practicality needs to be verified. These schemes mainly focus on access authorization and have not considered the key update situation.

In 2014, Liang *et al.* [17] proposed a PRE scheme to update the key and ciphertext. However its ciphertext length increases with the number of re-encryption times, and it only ensures CPA-secure. In 2016, Wang *et al.* [22] pointed out that Liang *et al.*'s scheme [17] cannot resist re-encryption key forgery attacks and conspiracy attacks. They proposed a cloud-assisted, scalable, and revocable identity-based encryption scheme with ciphertext evolution, which can resist collusion attack and CPA, and its ciphertext length is constant. In 2018, Sun *et al.* [23] proposed a CCA secure revocable identity-based encryption scheme with ciphertext evolution in a cloud. The scheme first uses the user's unique identifier as the identity to encrypt data to obtain the first-layer of ciphertext, then uses the identifier combined with the time as the identity to re-encrypt the first-layer ciphertext to obtain the second-layer ciphertext. When updating the ciphertext, the key corresponding to the combined identity of identifier and time is used to decrypt the second-layer ciphertext, then the identity corresponding to the new time is used to encrypt the first-layer ciphertext obtained. In fact, the first-layer key corresponding to the user's unique identification remains unchanged, and no real key update is realized. In addition, Wang *et al.*'s scheme [22] and Sun *et al.*'s scheme [23] utilize identity-based encryption (IBE), not PRE. When the data stored in the cloud need to be accessed by multiple users, different identities are used to encrypt the data and obtain different ciphertexts, which requires numerous storage resources.

Shafagh *et al.* [24] proposed a secure IoT data sharing solution with the key update. But it only achieves CPA security and cannot resist collusion attacks because it is based on Blaze *et al.*'s PRE scheme [2]. Yao *et al.* [25] improved Green *et al.*'s scheme [6] and proposed a PRE scheme, which has the function of ciphertext evolution and is collusion resistant. The calculation of ciphertext update in their scheme only requires one multiplication operation, which is very efficient. Unfortunately, their scheme only achieves CPA security in the random oracle model, and one ciphertext component needs to be downloaded in advance.

C. PAPER ORGANIZATION

The rest of the paper is organized as follows. The system model and some assumptions are given in Section II. Section III reviews some mathematical preliminaries and provides the definition and security model of a revocable identity-based conditional proxy re-encryption scheme with ciphertext evolution (RIB-CPRE-CE). Section IV presents the concrete construction, the correctness, and the security analysis of the proposed scheme. Section V shows the simple implementation of the proposed scheme. In Section VI, we analyze the efficiency and functions by comparing them with other related representative works. Finally, Section VII concludes the paper.

II. SYSTEM MODEL AND ASSUMPTIONS

A. SYSTEM MODEL

As shown in Fig. 1, the secure cloud sharing system consists of a trusted key generation center (private key generator, PKG), data owner, data requestor, and cloud server (storage server and proxy server).

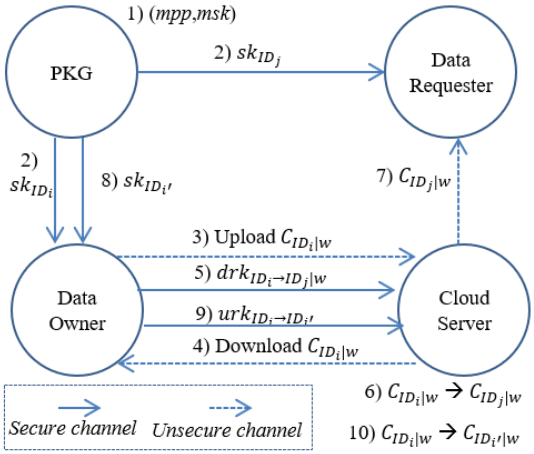

FIGURE 1. System model.

The private key generator (PKG) is a trusted third party responsible for generating the system parameters and the user's private key.

The data owner is the user as the delegator in the system, who rents computing and storage resources from cloud service providers, encrypts sensitive data, and uploads it to the cloud for storage. Furthermore, he/she negotiates identity update and key update with PKG, generates update or delegation re-encryption keys, and issues ciphertext update or delegation instructions to the proxy server.

The data requester is the user as the delegatee in the system, who needs to access the encrypted data stored in the cloud server by the data owner.

The storage server is a part of the cloud server, which stores user data and responds to data access requests from data owner or data requestor. Note that only one copy of the data is stored on the cloud server, usually the latest ciphertext corresponding to the data owner's newest key.

The proxy server, instantiated by the cloud server, is semi-trusted. That is, it will comply with the protocol and will not actively change the user's data. It needs to authenticate the user's identity, receive the user's re-encryption key through a secure channel, and perform legitimate ciphertext update operations or generate delegation ciphertext.

In addition, Fig. 1 shows ten types of operations among system components. The processings are shown as follows:

1) PKG generates the master public parameters (mpp) and the master secret key (msk). The former is distributed to all the participants, while the latter is kept secret. Moreover, PKG maintains a database of user identity information, which indicates the user's current valid identity.

2) After receiving a user's private key generation request, PKG first validates the request to ensure that it comes from a valid user and the key period of the identity is reasonable. If the check fails, PKG rejects the request. Otherwise, PKG generates the private key for the corresponding identity and returns it to the user over a secure channel.

3) The data owner, Alice, encrypts the sensitive data under her identity (e.g., $ID_i = Alice|20200720$) and condition w to generate an original ciphertext $C_{ID_i|w}$, which is uploaded to the cloud for storage.

4) The data owner can download the ciphertext $C_{ID_i|w}$ from the cloud and decrypts it with the private key sk_{ID_i}.

5) If Alice wants to share her data to a requester of identity ID_j, she generates a delegation token $drk_{ID_i \to ID_j|w}$ using her private key sk_{ID_i} and the requester's identity ID_j. The delegation token is sent to the proxy securely.

6) When the legitimate requester (say Bob) wants to access the data, the proxy inputs delegation token $drk_{ID_i \to ID_j|w}$ and ciphertext $C_{ID_i|w}$ into the re-encryption algorithm to generate a new ciphertext $C_{ID_j|w}$.

7) The proxy returns the delegation ciphertext $C_{ID_j|w}$ to the requester, the latter can decrypt it with the private key sk_{ID_j}.

8) When the user's identity expires or the private key is leaked, a new identity (e.g., $ID'_i = Alice|20200801$) will be sent to PKG to request a new private key. If the request is legitimate, PKG generates a new private key $sk_{ID'_i}$ corresponding to the new identity ID'_i and returns it to the user via a secure channel.

9) After the data owner (Alice) changes the identity and private key, she generates an update token $urk_{ID_i \to ID'_i}$ by using the old private key sk_{ID_i} and the new identity ID'_i. And the update token will be securely sent to the proxy for ciphertext evolution.

10) After receiving a legitimate ciphertext update request, the proxy transforms the previous ciphertext to be a new one $C_{ID'_i}$, and the previous one C_{ID_i} will be deleted. The cloud stores only one copy of the data, which usually is the latest one.

Next, the data owner can download the updated ciphertext and decrypt it with the new private key. The operation is the same as in step 4). Note that the old delegation token $drk_{ID_i \to ID_j|w}$ cannot be used to delegate the updated

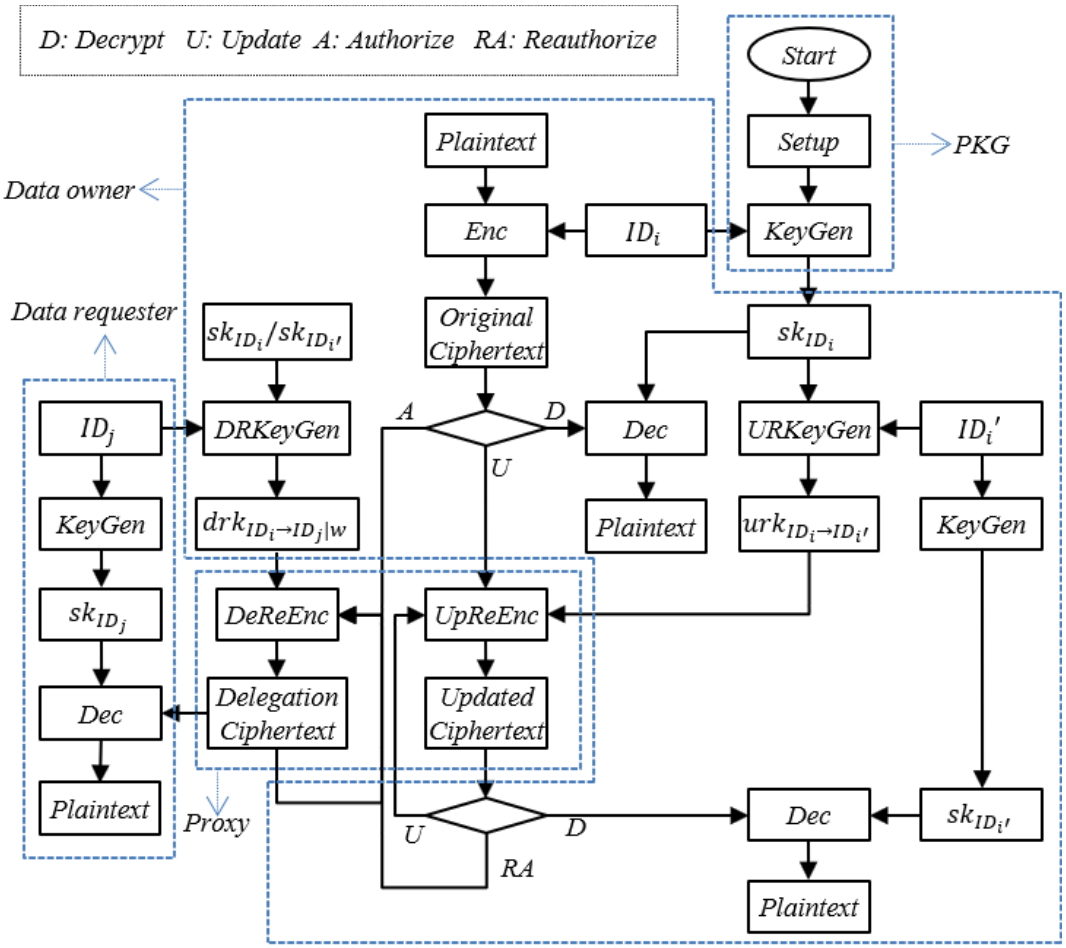

FIGURE 2. Process flow of RIB-CPRE-CE.

ciphertext $C_{ID_i'}$ anymore, which means that the delegation has been revoked by evolving the ciphertext. If the data owner wants to continue to share the data with the requester, a new delegation token needs to be generated. The process is the same as in step 5). In addition, the updated ciphertext can be updated as needed, and the delegation ciphertext can also be delegated multiple times. The work process is shown as in Fig. 2.

B. EXPLANATIONS AND ASSUMPTIONS

In order to facilitate the introduction of system functions, some explanations and assumptions are given here.

1) The specification of user identity, defined in master public parameters *mpp*, is described as below:

- User identity consists of a unique identifier (e.g., user name) and a validity period. For example, Alice|20200701-20200731 represents Alice's identity *ID* from July 1, 2020, to July 31, 2020.
- The validity period is limited in a duration window. Assuming that the identity interval is one month, the maximum validity period of the identity is from the first day to the last day of the month, and cross-cycle identities are not allowed. For example, Alice|20200720-20200820 is illegal, but Alice|20200705-20200726 is legal. For the sake of description, the identity's expiration date is set to be the last day of every month. The validity of identity will be denoted with the starting time, so Alice|20200801-20200831 can be simplified to Alice|20200801. And D_t is used to represent the validity window for identity *ID* with a validity period t. The validity window of Alice|20200801 is $D_t = 202008$.
- Normally, user Alice's identity *ID* will be updated to Alice|2020080120200831 in the next cycle. However, at a special time (e.g., July 20, 2020), Alice notices that the secret key has been compromised. She must immediately update her identity and key. The new identity's validity period will be set from the current time until the end of the month, such as Alice|20200720-20200731 (abbreviated as Alice|20200720).

2) Assuming PKG can authenticate the user's identity correctly when receiving a private key generation request. For the convenience of discussion, this authentication process is omitted in this system.

3) Assuming the proxy can authenticate the request of a ciphertext evolution. If the adversary forges an update token

and requests the proxy to update the ciphertext, it can detect the attack and reject it. Even it can send a notification about the attack to the data owner. The processing of this authentication is not discussed in this system, either.

4) Assuming that the proxy can secretly keep the legitimate delegation token and delete the expired one. And the proxy does not store an update token and old ciphertext after transforming ciphertext.

5) Assuming that all the private keys and re-encryption keys are transported via secure channels and kept securely.

6) Assuming all entities in the system follow the communication protocol and respond to the legitimate request correctly.

III. PRELIMINARIES

This section briefly introduces some foundational concepts and technologies used throughout this paper and provides the scheme's formal definition and security model.

A. BILINEAR MAPS

Groups (G, G_T) of prime order q are called *bilinear map groups* if there is a mapping $e : G \times G \to G_T$ with properties of 1) Bilinearity: $\forall a, b \in Z_q^*$, $\forall g, h \in G$, $e(g^a, h^b) = e(g, h)^{ab}$; 2) Non-degeneracy: $\forall g, h \neq 1_G$, $e(g, h) \neq 1_{G_T}$; 3) Computability: $\forall g, h \in G$, $e(g, h)$ is efficiently computable [5].

B. DECISIONAL BILINEAR DIFFIE-HELLMAN ASSUMPTION

Given an tuple $(p, p^a, p^b, p^c, Q) \in G^4 \times G_T$, the decisional bilinear Diffie-Hellman (DBDH) problem is to decide whether $Q = e(p, p)^{abc}$ or not. Define

$$Adv_{\mathcal{A}}^{DBDH} = |Pr[\mathcal{A}(p, p^a, p^b, p^c, e(p, p)^{abc}) = 0] - Pr[\mathcal{A}(p, p^a, p^b, p^c, Q) = 0]|$$

as the advantage of the adversary $\mathcal{A}$ in winning the DHDH problem. Obviously the DBDH assumption holds if no probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$ has non-negligible advantage in solving the DBDH problem [19].

C. STRONGLY UNFORGEABLE ONE-TIME SIGNATURES

A one-time signature (OTS) [26] consists of a triple of algorithms $Sig = (SKG, \mathcal{S}, \mathcal{V})$ such that: 1) on input a security parameter k, SKG outputs a OTS key pair (svk, ssk); 2) on input a message m and a signing key ssk, $\mathcal{S}(ssk, m)$ outputs a signature δ; 3) on input a verification key svk, a message m, and a signature δ, $\mathcal{V}(ssk, m, \delta)$ outputs 1 if δ is a valid signature of m, or outputs 0 otherwise. $Sig = (SKG, \mathcal{S}, \mathcal{V})$ is a strongly unforgeable one-time signature if the probability

$$\begin{aligned} &Adv_{\mathcal{A}}^{OTS}(1^k) \\ &= Pr[\mathcal{V}(ssk, \delta^*, m^*) = 1 : (ssk, svk) \\ &\leftarrow SKG(1^k); (m, ST) \leftarrow \mathcal{A}(svk); \delta \leftarrow \mathcal{S}(ssk, m); (m^*, \delta^*) \\ &\leftarrow \mathcal{A}(svk, \delta, ST); (m^*, \delta^*) \neq (m, \delta)], \end{aligned}$$

where ST is the state information, is negligible for any PPT adversary $\mathcal{A}$.

D. SCHEME DEFINITION

Definition 1: A revocable and identity-based conditional proxy re-encryption with ciphertext evolution (RIB-CPRE-CE) scheme consists of eight algorithms as follows:

1) $Setup(1^\lambda)$: On input a security parameter λ, this algorithm outputs a master public parameters (mpp) distributed to all participants of the system, and a master secret key (msk) kept secretly by the private key generator (PKG). Note that mpp is implicitly included in each of following algorithms.

2) $KeyGen(msk, ID)$: On input msk and a combination identity $ID = (I, t) \in Z_q^*$, where I is the unique identification of the user, and t is the validity period of ID, this algorithm outputs a corresponding private key sk_{ID}.

3) $Enc(ID, w, m)$: On input ID, the condition $w \in Z_q^*$, and the message $m \in G_T$, this algorithm outputs the original ciphertext $C_{ID|w}$ with a condition.

4) $Dec(sk_{ID}, C_{ID|*})$: On input sk_{ID} and the ciphertext $C_{ID|*}$ under identity ID and any condition, this algorithm outputs a message m or an error flag $\perp$.

5) $DRKeyGen(sk_{ID_i}, ID_j, w)$: On input sk_{ID_i}, the delegatee's identity ID_j, and w, this algorithm outputs a re-encryption key $drk_{ID_i \to ID_j|w}$ for delegation.

6) $DeReEnc(drk_{ID_i \to ID_j|w}, C_{ID_i|w})$: On input delegation re-encryption key $drk_{ID_i \to ID_j|w}$ and $C_{ID_i|w}$, this algorithm outputs the re-encrypted ciphertext $C_{ID_j|w}$ or a error flag $\perp$ which means the ciphertext $C_{ID_i|w}$ is invalid or the condition is not matched.

7) $URKeyGen(sk_{ID}, ID')$: On input the old private key sk_{ID} and the new identity $ID' = (I, t')$, this algorithm outputs a re-encryption key $urk_{ID \to ID'}$ for ciphertext update.

8) $UpReEnc(urk_{ID \to ID'}, C_{ID|*})$: On input $urk_{ID \to ID'}$ and the old ciphertext $C_{ID|*}$ under the old identity ID and any condition, this algorithm outputs a new ciphertext $C_{ID'|*}$ or a error flag $\perp$ which means the ciphertext $C_{ID|*}$ is invalid.

Correctness: For any security parameter λ, any identity $\{ID_k, ID_l \in Z_q^* | 1 \leq k \leq l - 1 \leq 2^\lambda - 1\}$, any condition $w \in Z_q^*$, and any message $m \in G_T$, if $(mpp, msk) \leftarrow Setup(1^\lambda)$, $sk_{ID_i} \leftarrow KeyGen(msk, ID_i)$, $drk_{ID_k \to ID_l|w} \leftarrow DRKeyGen(sk_{ID_k}, ID_l, w)$, and $urk_{ID_k \to ID_{k+1}} \leftarrow URKeyGen(sk_{ID_k}, ID_{k+1})$, the following conditions hold:

1) $Dec(sk_{ID_k}, Enc(ID_k, m)) = m$.

2) $Dec(sk_{ID_k}, UpReEnc(urk_{ID_{k-1} \to ID_k}, UpReEnc(urk_{ID_{k-2} \to ID_{k-1}}, \ldots, UpReEnc(urk_{ID_1 \to ID_2}, Enc(ID_1, w, m))))) = m$.

3) $Dec(sk_{ID_l}, DeReEnc(drk_{ID_{l-1} \to ID_l|w}, DeReEnc(drk_{ID_{l-2} \to ID_{l-1}|w}, \ldots, DeReEnc(drk_{ID_k \to ID_{k+1}|w}, Enc(ID_k, w, m))))) = m$.

E. SECURITY MODEL

As with scheme [19], some important terms for the security model are defined firstly as follows.

Corrupt/Honest Identity: If the private key corresponding to the user name I and a period t can be known by an

adversary, the combination identity $ID = (I, t)$ is corrupted. Otherwise, it is honest. And CI and HI is used to denote as the set of all corrupt identities and honest identities, respectively.

Directed Graph: The directed graph $DG_{urk} = (V_{urk}, A_{urk})$ is used to record the relationship between the identities and the update re-encryption keys involved in the generation of update re-encryption key. $DG_{ure} = (V_{ure}, A_{ure})$ is used to record the relationship between the identities and the update re-encryption keys involved in the update re-encryption. $DG_{drk} = (V_{drk}, A_{drk})$ is used to record the relationship between the identities and the delegation re-encryption keys involved in the generation of delegation re-encryption key. And $DG_{dre} = (V_{dre}, A_{dre})$ is used to records the relationship between the identities and the delegation re-encryption keys involved in the delegation re-encryption. In a $DG = (V, A)$, V is the set of vertices of DG, A is the set of arcs of DG and each edge a of A joins a head vertex v with a tail vertex u. In this secure model, V represents the set of identities and A represents the set of re-encryption keys.

Definition 2: A RIB-CPRE-CE scheme is CCA secure, if no probabilistic polynomial time (PPT) adversasry $\mathcal{A}$ wins the following game with a non-negligible advantage. In this game, $\mathcal{A}$ is the adversary and $\mathcal{C}$ is the challenger.

Init. Adversary $\mathcal{A}$ selects the challenge identity $ID^* = (I^*, t^*)$, condition w^* and sends them to the challenger $\mathcal{C}$.

Setup. Challenger $\mathcal{C}$ sets up the master public parameters mpp and the master secret key msk. The mpp is sent to $\mathcal{A}$ while the msk is kept secretly by $\mathcal{C}$. Besides, $\mathcal{C}$ maintains four directed graphs $DG_{urk} = (V_{urk}, A_{urk})$, $DG_{ure} = (V_{ure}, A_{ure})$, $DG_{drk} = (V_{drk}, A_{drk})$, and $DG_{dre} = (V_{dre}, A_{dre})$ and a queried identity set QI. Any vertex in graph DG_{urk}, DG_{ure}, DG_{drk}, and DG_{dre} represents an identity with key period in $HI \cup CI$. In graph DG_{urk}, each arc (directed edge) in A_{urk} represents a update re-encryption key of an identity from one key period to another. For example, $a_{((I,t_i),(I,t_j))}$ represents the re-encryption key $urk_{(I,t_i)\to(I,t_j)}$. In the graph DG_{ure}, each arc in A_{ure} represents an update query of ciphertext under an identity from one key period to another. For example, $a_{((I,t_i),(I,t_j))}$ represents a query about $\mathcal{O}_{ure}((I, t_i), (I, t_j), C_{(I,t_i)|w^*})$. In the graph DG_{drk}, each arc in A_{drk} represents a delegation re-encryption key of an identity to another identity in same validity window and under condition w^*. For example, $a_{((I_k,t_i),(I_l,t_j))}$ represents $drk_{(I_k,t_i)\to(I_l,t_j)|w^*}$, where $D_{t_i} = D_{t_j}$. In the graph DG_{dre}, each arc in A_{dre} represents a delegation re-encryption query from an identity to another identity in same validity window and under condition w^*. For example, $a_{((I_k,t_i),(I_l,t_j))}$ represents $\mathcal{O}_{dre}((I_k, t_i), (I_l, t_j), C_{(I_k,t_i)|w^*}, w^*)$, where $D_{t_i} = D_{t_j}$. QI denotes the set of the identities whose private key have been queried. Initially A_{urk}, A_{ure}, A_{drk}, A_{dre}, and QI are empty.

Phase 1. The adversary $\mathcal{A}$ can issue following queries adaptively.

- Private key query $\mathcal{O}_{sk}$: On input $ID = (I, t)$ by the adversary, $\mathcal{C}$ performs the following processing:
  1) If $I = I^*$ and $t = t^*$, $\mathcal{C}$ outputs $\perp$ and aborts.
  2) If $I = I^*$, $t > t^*$, and there is a path in DG_{urk} from vertex (I^*, t^*) to vertex (I, t), as shown in Fig. 3(a), $\mathcal{C}$ outputs $\perp$ and aborts.
  3) If $I \neq I^*$ and there is a path in $DG_{urk} \cup DG_{drk}$ from vertex (I^*, t^*) to vertex (I, t), as shown in Fig. 3(b), $\mathcal{C}$ outputs $\perp$ and aborts. Otherwise $\mathcal{C}$ returns $sk_{(I,t)} \leftarrow KeyGen(msk, (I, t))$.
- Update re-encryption key query $\mathcal{O}_{urk}$: On input $((I, t_i), (I, t_j))$ by the adversary, where $t_i < t_j$, $\mathcal{C}$ performs the following processing:
  1) If $I = I^*$, and after $a_{((I,t_i),(I,t_j))}$ being added to A_{urk}, there is a path in DG_{urk} from vertex (I^*, t^*) to vertex (I, t), where $t_j \leq t$, and $(I, t) \in QI \cup CI$, as shown in Fig. 3(c), $\mathcal{C}$ outputs $\perp$ and aborts.
  2) If $I = I^*$, and after $a_{((I,t_i),(I,t_j))}$ being added to A_{urk}, there is a path in $DG_{urk} \cup DG_{drk}$ from vertex (I^*, t^*) to vertex (I', t), where $D_{t_j} \leq D_t$, and $(I', t) \in QI \cup CI$, as shown in Fig. 3(d), $\mathcal{C}$ outputs $\perp$ and aborts.
  3) If $I \neq I^*$, and after $a_{((I,t_i),(I,t_j))}$ being added to A_{urk}, there is a path in $DG_{urk} \cup DG_{drk}$ from vertex (I^*, t^*) to vertex (I', t), where $D_{t^*} \leq D_{t_i} \leq D_{t_j} \leq D_t$, and $(I', t) \in QI \cup CI$, as shown in Fig. 3(e), $\mathcal{C}$ outputs $\perp$ and aborts. Otherwise, $\mathcal{C}$ returns $urk_{(I,t_i)\to(I,t_j)} \leftarrow URKeyGen(sk_{(I,t_i)}, (I, t_j))$ to $\mathcal{A}$ and adds $urk_{(I,t_i)\to(I,t_j)}$ to A_{urk}, where $sk_{(I,t_i)} \leftarrow KeyGen(msk, (I, t_i))$.
- Delegation re-encryption key query $\mathcal{O}_{drk}$: On input $((I_k, t_i), (I_l, t_j), w)$ by the adversary, where $D_{t_i} = D_{t_j}$, if $w \neq w^*$, $\mathcal{C}$ runs $DRKeyGen(sk_{(I_k,t_i)}, (I_l, t_j), w)$ to generate $drk_{(I_k,t_i)\to(I_l,t_j)|w}$. Otherwise, $\mathcal{C}$ performs the following processing:
  1) If $I_k = I^*$, and after $a_{((I_k,t_i),(I_l,t_j))}$ being added to A_{drk}, there is a path in $DG_{urk} \cup DG_{drk}$ from vertex (I^*, t^*) to vertex (I, t), and $(I, t) \in QI \cup CI$, where $D_{t^*} \leq D_{t_i}$ and $D_{t_j} \leq D_t$, as shown in Fig. 3(f), $\mathcal{C}$ outputs $\perp$ and aborts.
  2) If $I_k \neq I^*$, and after $a_{((I_k,t_i),(I_l,t_j))}$ being added to A_{drk}, there is a path in $DG_{urk} \cup DG_{drk}$ from vertex (I^*, t^*) to vertex (I, t), and $(I, t) \in QI \cup CI$, where $D_{t^*} \leq D_{t_i}$ and $D_{t_j} \leq D_t$, as shown in Fig. 3(g), $\mathcal{C}$ outputs $\perp$ and aborts. Otherwise, $\mathcal{C}$ runs $DRKeyGen(sk_{(I_k,t_i)}, (I_l, t_j), w)$ to generate $drk_{(I_k,t_i)\to(I_l,t_j)|w^*}$ and adds it into A_{drk}, where $sk_{(I_k,t_i)} \leftarrow KeyGen(msk, (I_k, t_i))$.
- Ciphertext update re-encryption query $\mathcal{O}_{ure}$: On input $((I, t_i), (I, t_j), C_{(I,t_i)|*})$ by the adversary, where $t_i < t_j$ and $*$ in the ciphertext $C_{(I,t_i)|*}$ denotes any condition, $\mathcal{C}$ runs $UpReEnc(urk_{(I,t_i)\to(I,t_j)}, C_{(I,t_i)|*})$ to generate $C_{(I,t_j)|*}$, where $urk_{(I,t_i)\to(I,t_j)} \leftarrow URKeyGen(sk_{(I,t_i)}, (I, t_j))$, and $sk_{(I,t_i)}$ is generated by $KeyGen(msk, (I, t_i))$.
- Ciphertext delegation re-encryption query $\mathcal{O}_{dre}$: On input $((I_k, t_i), (I_l, t_j), C_{(I_k,t_i)|w}, w)$ by the adversary, where $D_{t_i} = D_{t_j}$, $\mathcal{C}$ returns $C_{(I_l,t_j)|w} \leftarrow DeReEnc(drk_{(I_k,t_i)\to(I_l,t_j)|w}, C_{(I_k,t_i)|w})$, where $\mathcal{C}$ runs $DRKeyGen(sk_{(I_k,t_i)}, (I_l, t_j), w)$ to generate $drk_{(I_k,t_i)\to(I_l,t_j)|w}$ and $KeyGen(msk, (I_k, t_i))$ to generated $sk_{(I_k,t_i)}$.

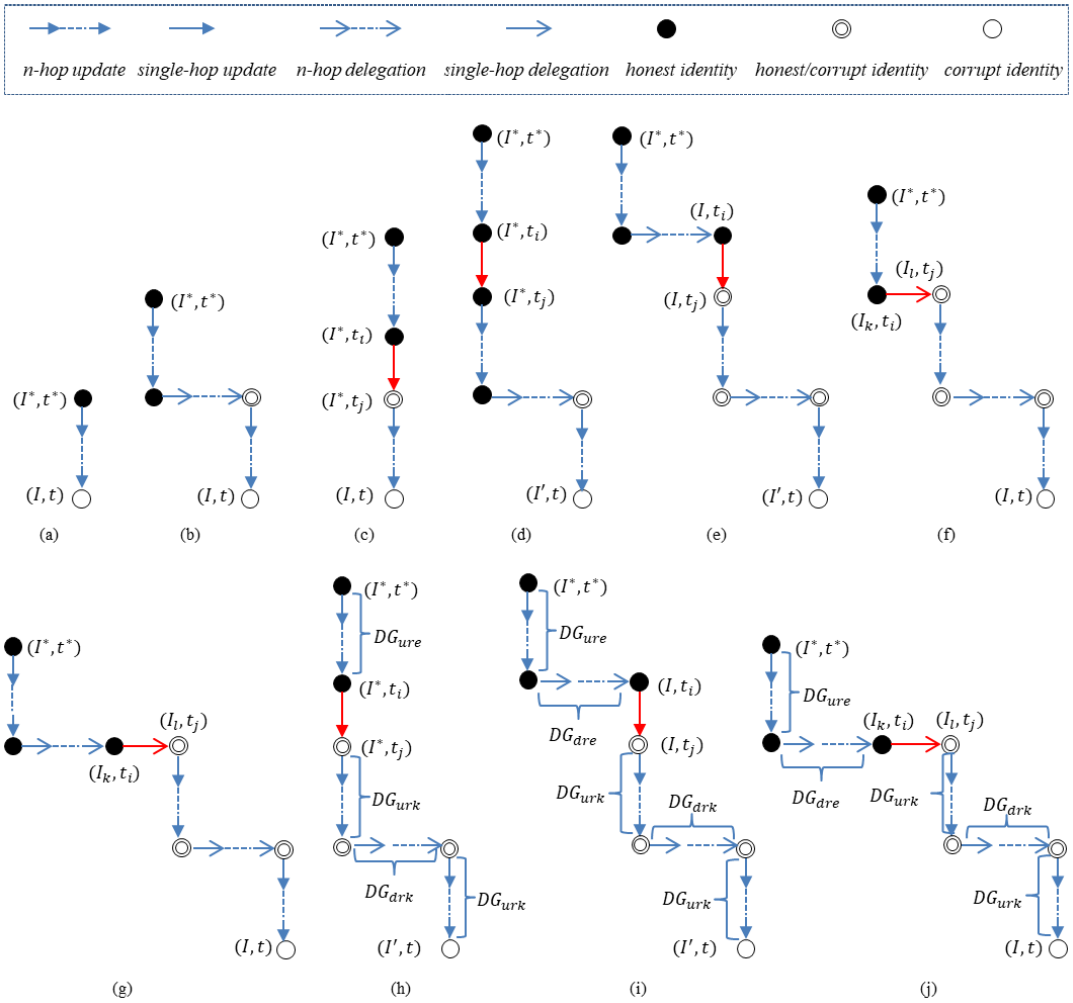

FIGURE 3. Directed graph.

- Decryption query $\mathcal{O}_{de}$: On input $((I, t), C_{(I,t)|*})$ by the adversary, $\mathcal{C}$ returns $m \leftarrow Dec(sk_{(I,t)}, C_{(I,t)|*})$, where $sk_{(I,t)} \leftarrow KeyGen(msk, (I, t))$.

Challenge. $\mathcal{A}$ submits two messages m_0, m_1 of equal length. $\mathcal{C}$ returns $C^*_{I^*,t^*|w^*} = Enc((I^*, t^*), w^*, m_d)$ to $\mathcal{A}$, where $d \in \{0, 1\}$.

Phase 2. $\mathcal{A}$ continues issuing queries.

- $\mathcal{O}_{sk}(I, t)$: $\mathcal{C}$ processes as Phase 1.
- $\mathcal{O}_{urk}((I, t_i), (I, t_j))$: $\mathcal{C}$ processes as Phase 1.
- $\mathcal{O}_{drk}((I_k, t_i), (I_l, t_j), w)$: $\mathcal{C}$ processes as Phase 1.
- $\mathcal{O}_{ure}((I, t_i), (I, t_j), C_{(I,t_i)|*})$: If $C_0 \neq svk^*$, $\mathcal{C}$ processes as Phase 1. Otherwise, $\mathcal{C}$ performs the following processing:
  1) If $I = I^*$, and after $a_{((I,t_i),(I,t_j))}$ being added to A_{ure}, there is a path in $DG_{urk} \cup DG_{ure} \cup DG_{drk} \cup DG_{dre}$ from vertex (I^*, t^*) to vertex (I', t), where $D_{t_j} \leq D_t$, and at least one vertex in the path from vertex (I^*, t_j) to vertex (I', t) is in $QI \cup CI$, as shown in Fig. 3(h), $\mathcal{C}$ outputs $\perp$ and aborts.
  2) If $I \neq I^*$, and after $a_{((I,t_i),(I,t_j))}$ being added to A_{ure}, there is a path in $DG_{urk} \cup DG_{ure} \cup DG_{drk} \cup DG_{dre}$ from vertex (I^*, t^*) to vertex (I', t), where $D_{t_j} \leq D_t$, and at least one vertex in the path from vertex (I^*, t_j) to vertex (I', t) is in $QI \cup CI$, as shown in Fig. 3(i), $\mathcal{C}$ outputs $\perp$ and aborts. Otherwise, $\mathcal{C}$ returns $C_{(I,t_j)|*}$ to $\mathcal{A}$ as Phase 1 and adds $a_{(I,t_i),(I,t_j)}$ into A_{ure}.
- $\mathcal{O}_{dre}((I_k, t_i), (I_l, t_j), C_{(I_k,t_i)|w}, w)$: If C_0 in $C_{(I_k,t_i)|w}$ is not equal to svk^* or $w \neq w^*$, $\mathcal{C}$ processes as Phase 1. Otherwise, $\mathcal{C}$ checks that, if after $a_{((I_k,t_i),(I_l,t_j))}$ being added to A_{dre}, there is a path in $DG_{urk} \cup DG_{ure} \cup DG_{drk} \cup DG_{dre}$ from vertex (I^*, t^*) to vertex (I, t), where $D_{t_j} \leq D_t$, and at least one vertex in the path from vertex (I_l, t_j) to vertex (I, t) is in $QI \cup CI$, as shown in Fig. 3(j), $\mathcal{C}$ outputs $\perp$ and aborts. Otherwise, $\mathcal{C}$ returns $C_{(I_l,t_j)|w}$ to $\mathcal{A}$ as Phase 1 and adds $a_{((I_k,t_i),(I_l,t_j))}$ into A_{dre}.
- $\mathcal{O}_{de}((I, t), C_{(I,t)|*})$: If C_0 in $C_{(I,t)|*}$ is equal to the OTS verification key of the challenge ciphertext, $\mathcal{C}$ outputs $\perp$ and aborts. Otherwise, $\mathcal{C}$ processes as Phase 1.

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

IV. PROPOSAL

In this section, the generic construction, concrete construction, correctness analysis, security analysis, and security proof of the scheme are presented.

A. THE GENERIC CONSTRUCTION OF THE SCHEME

The general proxy re-encryption scheme consists of six algorithms: PRE_{Setup}, PRE_{KeyGen}, PRE_{Enc}, PRE_{Dec}, $PRE_{DRKeyGen}$, and $PRE_{DeReEnc}$. Two new algorithms, $PRE_{URKeyGen}$ for calculating update tokens and $PRE_{UpReEnc}$ for updating ciphertexts, are integrated into Xiong *et al.*'s PRE scheme to construct a novel revocable and identity-based conditional proxy re-encryption scheme with ciphertext evolution (RIB-CPRE-CE). The generic algorithm construction is shown in Fig. 4.

- $Setup(1^\lambda)$:

$PRE_{Setup}(1^\lambda) \rightarrow (mpp, msk)$

- $KeyGen(msk, ID)$:

$PRE_{KeyGen}(mpp, msk, ID) \rightarrow sk_{ID}$

- $Enc(ID, w, m)$:

$PRE_{Enc}(mpp, ID, w, m) \rightarrow C_{ID|w}$

- $Dec(sk_{ID}, C_{ID|w})$:

$PRE_{Dec}(mpp, sk_{ID}, C_{ID|w}) \rightarrow m$

- $ReKeyGen(sk_{ID_i}, ID_j, w)$:

$PRE_{DRKeyGen}(mpp, sk_{ID_i}, ID_j, w) \rightarrow drk_{ID_i \rightarrow ID_j|w}$

- $ReEnc\left(drk_{ID_i \rightarrow ID_j|w}, C_{ID_i|w}\right)$:

$PRE_{DeReEnc}(mpp, drk_{ID_i \rightarrow ID_j|w}, C_{ID_i|w}) \rightarrow C_{ID_j|w}$

- $UpKeyGen(sk_{ID}, ID')$:

$PRE_{URKeyGen}(mpp, sk_{ID}, ID') \rightarrow urk_{ID \rightarrow ID'}$

- $ReEnc(urk_{ID \rightarrow ID'}, C_{ID|*})$:

$PRE_{UpReEnc}(mpp, urk_{ID_i \rightarrow ID_j|w}, C_{ID_i|*}) \rightarrow C_{ID_j|*}$

FIGURE 4. Generic algorithm construction of RIB-CPRE-CE.

B. THE CONCRETE CONSTRUCTION OF THE SCHEME

The concrete construction is described as follows:

- $Setup(1^\lambda)$: On input a security parameter λ, this algorithm first chooses two multiplicative groups G and G_T of prime order q such that a bilinear map $e : G \times G \rightarrow G_T$, then randomly picks $a, b, c \in Z_q^*$, a generator g of G, five group elements $h, g_1, g_2, g_3, g_4 \in G$, and a one-time strong unforgeable signature scheme $Sig = (SKG, S, V)$. Next it computes $u_a = e(g, h)^a$, $u_b = e(g, h)^b$, $v_c = g^c$ and defines Z_q^* and G_T as the identity space and message space, respectively. This algorithm sets the master secret key $msk = (a, b, c)$ which will be kept secretly by the private key generator (PKG), and outputs the master public parameters $mpp = (q, e, G, G_T, g, h, g_1, g_2, g_3, g_4, Sig, u_a, u_b, v_c)$ to all participants in the system. For the convenience of description, the following algorithms implicitly include mpp.
- $KeyGen(msk, ID_i)$: Taking as input msk and an identity $ID_i \in Z_q^*$, this algorithm chooses $s_i, \alpha_i, \beta_i, \gamma_i \in Z_q^*$ randomly, computes $sk_{i,1} = \frac{c+s_i}{a+b \cdot ID_i}$, $sk_{i,2} = h^{s_i}$, $sk_{i,3} = g^{s_i}$, $sk_{i,4} = \frac{a+\alpha_i}{a+b \cdot ID_i}$, $sk_{i,5} = \frac{b+\beta_i}{a+b \cdot ID_i}$, $sk_{i,6} = \frac{\gamma_i}{a+b \cdot ID_i}$, $sk_{i,7} = h^{\alpha_i}$, $sk_{i,8} = h^{\beta_i}$, $sk_{i,9} = h^{\gamma_i}$, and outputs the private key $sk_{ID_i} = (sk_{i,1}, sk_{i,2}, sk_{i,3}, sk_{i,4}, sk_{i,5}, sk_{i,6}, sk_{i,7}, sk_{i,8}, sk_{i,9})$.
- $Enc(ID_i, w, m)$: Taking as input ID_i, the condition $w \in Z_q^*$, and the message $m \in G_T$, this algorithm first runs $SKG(1^\lambda)$ to generate a one-time signature key pair (ssk, svk), where $svk \in Z_q^*$, sets $C_0 = svk$. Next, it chooses $r \in_R Z_q^*$ and computes $C_1 = g^r$, $C_2 = m \cdot e(v_c, h)^r$, $C_3 = (u_a \cdot u_b^{ID_i})^r$, $C_4 = ({g_1}^w \cdot g_2)^r$, $C_5 = ({g_3}^{svk} \cdot g_4)^r$. At last, it generates a signature $C_6 = S(ssk, (w, C_1, C_2, C_4, C_5))$ and outputs $C_{ID_i|w} = (w, C_0, C_1, C_2, C_3, C_4, C_5, C_6)$ as the original ciphertext.
- $Dec(sk_{ID_x}, C_{ID_x|w})$: Taking as input sk_{ID_x} and $C_{ID_x|w}$, the this algorithm first checks the validity of $C_{ID_x|w}$ by verifying whether the following equalities hold:

$$e(g, C_4) = e(C_1, {g_1}^w \cdot g_2), \tag{1}$$

$$e(g, C_5) = e(C_1, {g_3}^{C_0} \cdot g_4), \tag{2}$$

$$V(C_0, C_6, (w, C_1, C_2, C_4, C_5)) = 1. \tag{3}$$

  If any of the above equations fails, this algorithm outputs an error flag $\perp$ and aborts. Otherwise, it computes and outputs $m = C_2 \cdot e(C_1, sk_{x,2}) / {C_3}^{sk_{x,1}}$.
- $DRKeyGen(sk_{ID_i}, ID_j, w)$: Taking as input $sk_{ID_i} = (sk_{i,1}, sk_{i,2}, sk_{i,3}, sk_{i,4}, sk_{i,5}, sk_{i,6}, sk_{i,7}, sk_{i,8}, sk_{i,9})$, delegatee's identity ID_j and w, this algorithm chooses $dk_1, dk_2, dk_3 \in {}_R Z_q^*$, computes $drk_1 = (sk_{i,4} + dk_1 \cdot sk_{i,6}) + (sk_{i,5} + dk_2 \cdot sk_{i,6}) \cdot ID_j$, $drk_2 = (sk_{i,7} \cdot {sk_{i9}}^{dk_1}) \cdot (sk_{i,8} \cdot {sk_{i,9}}^{dk_2})^{ID_j} \cdot ({g_1}^w \cdot g_2)^{dk_3}$, $drk_3 = g^{dk_3}$ and outputs the re-encryption key $drk_{ID_i \rightarrow ID_j|w} = (drk_1, drk_2, drk_3)$.
- $DeReEnc(drk_{ID_i \rightarrow ID_j|w}, C_{ID_i|w})$: Taking the delegation re-encryption key $drk_{ID_i \rightarrow ID_j|w}$, and ciphertext $C_{ID_i|w} = (w, C_{i,0}, C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4}, C_{i,5}, C_{i,6})$ as input, this algorithm checks the validity of $C_{ID_i|w}$ by (1), (2), and (3). If all the equations hold, this algorithm computes $C_{j,3} = {C_{i,3}}^{drk_1} \cdot e(drk_3, C_{i,4}) / e(drk_2, C_{i,1})$, $C_{j,0} = C_{i,0}$, $C_{j,1} = C_{i,1}$, $C_{j,2} = C_{i,2}$, $C_{j,4} = C_{i,4}$, $C_{j,5} = C_{i,5}$, and $C_{j,6} = C_{i,6}$, then outputs the delegation ciphertext $C_{ID_j|w} = (w, C_{j,0}, C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}, C_{j,5}, C_{j,6})$. Otherwise, outputs $\perp$ and aborts.
- $URKeyGen(sk_{ID_i}, ID_i')$: Taking as input an old private key sk_{ID_i}, and a new identity ID_i', this algorithm chooses $uk_1, uk_2 \in {}_R Z_q^*$, computes $urk_1 = (sk_{i,4} + uk_1 \cdot sk_{i,6}) + (sk_{i,5} + uk_2 \cdot sk_{i,6}) \cdot ID_i'$, $urk_2 = (sk_{i,7} \cdot {sk_{i,9}}^{uk_1}) \cdot (sk_{i,8} \cdot {sk_{i,9}}^{uk_2})^{ID_i'}$, and outputs the re-encryption key $urk_{ID_i \rightarrow ID_i'} = (urk_1, urk_2)$ for updating ciphertext.
- $UpReEnc(urk_{ID_i \rightarrow ID_i'}, C_{ID_i|*})$: Taking as input an update key $urk_{ID_i \rightarrow ID_i'}$, an old ciphertext $C_{ID_i|*} = (C_0, C_1, C_2, C_3, C_4, C_5, C_6)$ encrypted under identity ID_i and any condition, this algorithm first checks the

validity of $C_{ID_i|*}$ as algorithm *DeReEnc* and *Dec*. If the check is passed, this algorithm computes $C_3' = {C_3}^{urk_1}/e(urk_2, C_1)$, $C_0' = C_0$, $C_1' = C_1$, $C_2' = C_2$, $C_4' = C_4$, $C_5' = C_5$, and $C_6' = C_6$, then outputs the delegation ciphertext $C_{ID_i'|*} = (w, C_0', C_1', C_2', C_3', C_4', C_5', C_6')$. Otherwise, it outputs $\perp$ and aborts.

C. THE CORRECTNESS OF THE SCHEME

In this subsecion, the correctness of the proposed scheme is shown.

1) THE CORRECTNESS OF THE ORIGINAL CIPHERTEXT

Parse $C_{ID_i|w} = (w, C_{i,0}, C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4}, C_{i,5}, C_{i,6})$ and $sk_{ID_i} = (sk_{i,1}, sk_{i,2}, \ldots, sk_{i,9})$. The decryption of the original ciphertext is processed as follows:

$$\begin{aligned} m &= \frac{C_{i,2} \cdot e(C_{i,1}, sk_{i,2})}{{C_{i,3}}^{sk_{i,1}}} \\ &= \frac{m \cdot e(g,h)^{c \cdot r} \cdot e(g^r, h^{s_i})}{(e(g,h)^{(a+b \cdot ID_i) \cdot r})^{\frac{c+s_i}{a+b \cdot ID_i}}} \\ &= \frac{m \cdot e(g,h)^{c \cdot r} \cdot e(g,h)^{r \cdot s_i}}{e(g,h)^{r \cdot (c+s_i)}} \\ &= m. \end{aligned}$$

2) THE CORRECTNESS OF THE DELEGATION

i) The delegation re-encryption key $drk_{ID_i \to ID_j|w} = (drk_1, drk_2, drk_3)$ for ciphertext delegation is processed as follows:

$$\begin{aligned} drk_1 &= (sk_{i,4} + dk_1 \cdot sk_{i,6}) + (sk_{i,5} + dk_2 \cdot sk_{i,6}) \cdot ID_j \\ &= \frac{a + b \cdot ID_j + \alpha_i + dk_1 \cdot \gamma_i + (\beta_i + dk_2 \cdot \gamma_i) \cdot ID_j}{a + b \cdot ID_i}, \\ drk_2 &= (sk_{i,7} \cdot {sk_{i,9}}^{dk_1})({sk_{i,8}} \cdot {sk_{i,9}}^{dk_2})^{ID_j} ({g_1}^w \cdot g_2)^{dk_3} \\ &= (h^{\alpha_i} \cdot (h^{\gamma_i})^{dk_1})(h^{\beta_i} \cdot (h^{\gamma_i})^{dk_2})^{ID_j} ({g_1}^w \cdot g_2)^{dk_3} \\ &= h^{(\alpha_i + dk_1 \cdot \gamma_i + (\beta_i + dk_2 \cdot \gamma_i) \cdot ID_j)} \cdot ({g_1}^w \cdot g_2)^{dk_3}, \\ drk_3 &= g^{dk_3}. \end{aligned}$$

ii) To transform $C_{ID_i|w}$ into $C_{ID_j|w}$, it is processed as follows:

$$\begin{aligned} C_{j,3} &= \frac{{C_{i,3}}^{drk_1} \cdot e(drk_3, C_{i,4})}{e(drk_2, C_{i,1})} \\ &= \frac{(e(g,h)^{(a+b \cdot ID_i) \cdot r})^{drk_1} \cdot e(g^{dk_3}, ({g_1}^w \cdot g_2)^r)}{e(h^{(\alpha_i + dk_1 \cdot \gamma_i + (\beta_i + dk_2 \cdot \gamma_i) \cdot ID_j)} \cdot ({g_1}^w \cdot g_2)^{dk_3}, g^r)} \\ &= \frac{(e(g,h)^{(a+b \cdot ID_i) \cdot r})^{\frac{a+b \cdot ID_j + \alpha_i + dk_1 \cdot \gamma_i + (\beta_i + dk_2 \cdot \gamma_i) \cdot ID_j}{a+b \cdot ID_i}}}{e(h^{(\alpha_i + dk_1 \cdot \gamma_i + (\beta_i + dk_2 \cdot \gamma_i) \cdot ID_j)}, g^r)} \\ &= e(g,h)^{r \cdot (a + b \cdot ID_j)}. \end{aligned}$$

iii) The decryption of delegation ciphertext $C_{ID_j|w}$ under private key $sk_{ID_j} = (sk_{j,1}, sk_{j,2}, \ldots, sk_{j,9})$, where $sk_{j,1} = \frac{c+s_j}{a+b \cdot ID_j}$, $sk_{j,2} = h^{s_j}$, $sk_{j,3} = g^{s_j}$, $sk_{j,4} = \frac{a+\alpha_j}{a+b \cdot ID_j}$, $sk_{i,5} = \frac{b+\beta_i}{a+b \cdot ID_i}$, $sk_{i,6} = \frac{\gamma_i}{a+b \cdot ID_i}$, $sk_{j,7} = h^{\alpha_j}$, $sk_{j,8} = h^{\beta_j}$, $sk_{j,9} = h^{\gamma_j}$, is shown as follows:

$$\begin{aligned} m &= \frac{C_{j,2} \cdot e(C_{j,1}, sk_{j,2})}{{C_{j,3}}^{sk_{j,1}}} \\ &= \frac{m \cdot e(g,h)^{c \cdot r} \cdot e(g^r, h^{s_j})}{(e(g,h)^{(a+b \cdot ID_j) \cdot r})^{\frac{c+s_j}{a+b \cdot ID_j}}} \\ &= \frac{m \cdot e(g,h)^{c \cdot r} \cdot e(g,h)^{r \cdot s_j}}{e(g,h)^{r \cdot (c+s_j)}} \\ &= m. \end{aligned}$$

3) THE CORRECTNESS OF CIPHERTEXT UPDATE

i) The re-encryption key $urk_{ID_i \to ID_i'} = (urk_1, urk_2)$ for ciphertext update is processed as follows:

$$\begin{aligned} urk_1 &= (sk_{i,4} + uk_1 \cdot sk_{i,6}) + (sk_{i,5} + uk_2 \cdot sk_{i,6}) \cdot ID_i' \\ &= \frac{a + b \cdot ID_i' + \alpha_i + uk_1 \cdot \gamma_i + (\beta_i + uk_2 \cdot \gamma_i) \cdot ID_i'}{a + b \cdot ID_i}, \\ urk_2 &= (sk_{i,7} \cdot {sk_{i,9}}^{uk_1}) \cdot ({sk_{i,8}} \cdot {sk_{i,9}}^{uk_2})^{ID_i'} \\ &= (h^{\alpha_i} \cdot (h^{\gamma_i})^{uk_1}) \cdot (h^{\beta_i} \cdot (h^{\gamma_i})^{uk_2})^{ID_i'} \\ &= h^{(\alpha_i + uk_1 \cdot \gamma_i + (\beta_i + uk_2 \cdot \gamma_i) \cdot ID_i')}. \end{aligned}$$

ii) To transform $C_{ID_i|*}$ into $C_{ID_i'|*}$, it is processed as follows:

$$\begin{aligned} C_3' &= \frac{{C_3}^{urk_1}}{e(urk_2, C_1)} \\ &= \frac{(e(g,h)^{(a+b \cdot ID_i) \cdot r})^{\frac{(a+b \cdot ID_i') + (\alpha_i + uk_1 \cdot \gamma_i + (\beta_i + uk_2 \cdot \gamma_i) \cdot ID_i')}{a+b \cdot ID_i}}}{e(h^{(\alpha_i + uk_1 \cdot \gamma_i + (\beta_i + uk_2 \cdot \gamma_i) \cdot ID_i')}, g^r)} \\ &= e(g,h)^{r \cdot (a + b \cdot ID_i')}. \end{aligned}$$

iii) The decryption of updated ciphertext $C_{ID_i'|*}$ under private key $sk_{ID_i}' = (sk_{i,1}', sk_{i,2}', \ldots, sk_{i,9}')$, where $sk_{i,1}' = \frac{c+s_i'}{a+b \cdot ID_i'}$, $sk_{i,2}' = h^{s_i'}$, $sk_{i,3}' = g^{s_i'}$, $sk_{i,4}' = \frac{a+\alpha_i'}{a+b \cdot ID_i'}$, $sk_{i,5}' = \frac{b+\beta_i'}{a+b \cdot ID_i'}$, $sk_{i,6}' = \frac{\gamma_i'}{a+b \cdot ID_i'}$, $sk_{i,7}' = h^{\alpha_i'}$, $sk_{i,8}' = h^{\beta_i'}$, $sk_{i,9}' = h^{\gamma_i'}$, is shown as follows:

$$\begin{aligned} m &= \frac{C_{i,2}' \cdot e(C_{i,1}', sk_{i,2}')}{{C_{i,3}'}^{sk_{i,1}'}} \\ &= \frac{m \cdot e(g,h)^{c \cdot r} \cdot e(g^r, h^{s_i'})}{(e(g,h)^{(a+b \cdot ID_i') \cdot r})^{\frac{c+s_i'}{a+b \cdot ID_i'}}} \\ &= \frac{m \cdot e(g,h)^{c \cdot r} \cdot e(g,h)^{r \cdot s_i'}}{e(g,h)^{r \cdot (c+s_i')}} \\ &= m. \end{aligned}$$

D. THE SECURITY ANALYSIS OF THE SCHEME

This subsection first analyzes the security of the updated ciphertext and the new private key. Then the impact of the update re-encryption to the delegation is also analyzed. Finally, a security proof of the proposed scheme is given.

1) THE SECURITY OF UPDATED CIPHERTEXT

After the ciphertext is updated, the old key sk_{ID_i} cannot decrypt the updated ciphertext $C_{ID_i'|*}$. The verification is shown as follows:

$$\frac{C'_{i,2}\cdot e(C'_{i,1}, sk_{i,2})}{{C'_{i,3}}^{sk_{i,1}}} = \frac{m\cdot e(g,h)^{c\cdot r}\cdot e(g^r, h^{s_i})}{(e(g,h)^{(a+b\cdot ID'_i)\cdot r})^{\frac{c+s_i}{a+b\cdot ID_i}}} \neq m.$$

2) THE SECURITY OF NEW PRIVATE KEY

When the proxy receives the data owner's update re-encryption key $urk_{ID_i\to ID_i'}$ and knows the old private key sk_{ID_i}, it cannot derive the new private key $sk_{ID_i'}$. Also, it is impossible to derive the date owner's private key from the delegation re-encryption key even the requester colludes with the proxy. It is straightforward to verify this security, which is also termed as master secret security or collision-resistance.

3) THE IMPACT OF THE UPDATE RE-ENCRYPTION TO THE DELEGATION

After the ciphertext C_{ID_i} has been updated to $C_{ID_i'}$, the proxy cannot use the old delegation re-encryption key $drk_{ID_i\to ID_j}$ to delegate access to the requester (user ID_j) any more. The verification is shown as follows:

i) To transform $C_{ID_i'|w}$ into $C_{ID_j|w}$ with $drk_{ID_i\to ID_j}$. Here only the conversion of $C_{j,3}$ is given.

$$\begin{aligned}
C_{j,3} &= \frac{{C'_{i,3}}^{drk_1}\cdot e(drk_3, C'_{i,4})}{e(drk_2, C'_{i,1})}\\
&= \frac{(e(g,h)^{(a+b\cdot ID'_i)\cdot r})^{drk_1}\cdot e(g^{dk_3}, (g_1{}^w\cdot g_2)^r)}{e(h^{(\alpha_i+dk_1\cdot\gamma_i+(\beta_i+dk_2\cdot\gamma_i)\cdot ID_j)}\cdot(g_1{}^w\cdot g_2)^{dk_3}, g^r)}\\
&= \frac{(e(g,h)^{(a+b\cdot ID'_i)\cdot r})^{\frac{a+b\cdot ID_j+\alpha_i+dk_1\cdot\gamma_i+(\beta_i+dk_2\cdot\gamma_i)\cdot ID_j}{a+b\cdot ID_i}}}{e(h^{(\alpha_i+dk_1\cdot\gamma_i+(\beta_i+dk_2\cdot\gamma_i)\cdot ID_j)}, g^r)}\\
&= \frac{e(g,h)^{\frac{a+b\cdot ID'_i}{a+b\cdot ID_i}\cdot(a+b\cdot ID_j+\alpha_i+dk_1\cdot\gamma_i+(\beta_i+dk_2\cdot\gamma_i)\cdot ID_j)\cdot r}}{e(g,h)^{(\alpha_i+dk_1\cdot\gamma_i+(\beta_i+dk_2\cdot\gamma_i)\cdot ID_j)\cdot r}}
\end{aligned}$$

ii) To decrypt the delegation ciphertext $C_{ID_j|w}$ with private key sk_{ID_j} cannot get the plaintext m. The process is shown as follows:

$$\begin{aligned}
&\frac{C_{j,2}\cdot e(C_{j,1}, sk_{j,2})}{{C_{j,3}}^{sk_{j,1}}}\\
&= \frac{m\cdot e(g,h)^{c\cdot r}\cdot e(g^r, h^{s_j})}{(C_{j,3})^{\frac{c+s_j}{a+b\cdot ID_j}}}\\
&\neq m.
\end{aligned}$$

Note that the ciphertext update also implies the revocation of the previous authorization. Therefore, if the data needs to continue to be shared with the requester, a new delegate key needs to be generated by the data owner.

Next, it is shown that the proxy holding the update re-encryption key $urk_{ID_i\to ID_i'}$ cannot derive a new delegation re-encryption key $drk_{ID_i'\to ID_j'} = (drk'_1, drk'_2, drk'_3)$, where

$$\begin{aligned}
drk'_1 &= sk'_{i,4} + dk'_1\cdot sk'_{i,6} + (sk'_{i,5} + dk'_2\cdot sk'_{i,6})\cdot ID'_j\\
&= \frac{a + b\cdot ID'_j + \alpha'_i + dk'_1\cdot\gamma'_i + (\beta'_i + dk'_2\cdot\gamma'_i)\cdot ID'_j}{a + b\cdot ID'_i},\\
drk'_2 &= (sk'_{i,7}\cdot {sk'_{i,9}}^{dk'_1})(sk'_{i,8}\cdot {sk'_{i,9}}^{dk'_2})^{ID'_j}(g_1{}^w\cdot g_2)^{dk'_3}\\
&= (h^{\alpha'_i}\cdot(h^{\gamma'_i})^{dk'_1})(h^{\beta'_i}\cdot(h^{\gamma'_i})^{dk'_2})^{ID'_j}(g_1{}^w\cdot g_2)^{dk'_3}\\
&= h^{(\alpha'_i+dk'_1\cdot\gamma'_i+(\beta'_i+dk'_2\cdot\gamma'_i)\cdot ID'_j)}\cdot(g_1{}^w\cdot g_2)^{dk'_3},\\
drk_3 &= g^{dk'_3}.
\end{aligned}$$

It can be found from the generation of $drk_{ID_i'\to ID_j'}$ and $urk_{ID_i\to ID_i'}$ that it is obviously impossible to derive the latter directly from the former, because $\alpha_i, \beta_i, \gamma_i$ are different from $\alpha'_i, \beta'_i, \gamma'_i$.

4) THE SECURITY PROOF OF THE PROPOSED SCHEME

Theorem 1: Assuming that the one-time signature is strongly unforgeable and the DBDH assumption holds, the proposed RIB-CPRE-CE scheme is CCA-secure in the standard model.

Proof. Assume there exists an adversary $\mathcal{A}$ that can break the CCA security of the proposed scheme, then it possibly builds another algorithm $\mathcal{C}$ that can break the DBDH assumption (i.e., given p, p^a, p^b, p^c, Q, it is hard to decide $Q = e(p,p)^{abc}$) by playing a CCA game with $\mathcal{A}$. The details are shown as follows.

Init. Adversary $\mathcal{A}$ chooses the challenge identity $ID^* = (I^*, t^*)$ and condition w^*.

Setup. Let (p, p^a, p^b, p^c, Q) as the DBDH instance. $\mathcal{C}$ selects $a_0, b_0, c_0 \in_R Z_q^*$, sets $g = p^a$, $h = p^b$, $g_1 = p^{s_1}$, $g_2 = p^{s_2}$, $g_3 = p^{s_3}$, $g_4 = p^{s_4}$, where $s_1, s_2, s_3, s_4 \in_R Z_q^*$, picks a strongly unforgeable one-time signature scheme $Sig = (SKG, S, V)$, computes $u_a = e(g,h)^{a_0}$, $u_b = e(g,h)^{b_0}$, $v_c = g^{c_0}$, and outputs the scheme's system parameters $(q, g, G, G_T, e, h, g_1, g_2, g_3, g_4, u_a, u_b, v_c, Sig)$. $\mathcal{C}$ also maintains four directed graphs $DG_{urk} = (V_{urk}, A_{urk})$, $DG_{ure} = (V_{ure}, A_{ure})$, $DG_{drk} = (V_{drk}, A_{drk})$, $DG_{dre} = (V_{dre}, A_{dre})$ and a queried identity set QI defined in the security model. Initially, $A_{urk}, A_{ure}, A_{drk}, A_{dre}$ and QI are empty.

Phase 1. In this stage, $\mathcal{C}$ responds $\mathcal{A}$'s following queries as following:

- $\mathcal{O}_{sk}$: On input an identity (I, t), $\mathcal{C}$ performs the following processing:
  1) If $(I, t) = (I^*, t^*)$, $\mathcal{C}$ outputs $\perp$ and aborts.
  2) If $I = I^*$, $t > t^*$, and there is a path in DG_{urk} from vertex (I^*, t^*) to vertex (I, t), as shown in Fig. 3(a), $\mathcal{C}$ outputs $\perp$ and aborts.
  3) If $I \neq I^*$ and there is a path in $DG_{urk} \cup DG_{drk}$ from vertex (I^*, t^*), as shown in Fig. 3(b), $\mathcal{C}$ outputs $\perp$ and aborts. Otherwise, $\mathcal{C}$ returns $sk_{(I,t)} \leftarrow KeyGen(msk, (I, t))$ to $\mathcal{A}$ and adds (I, t) to QI, where $msk = (a_0, b_0, c_0)$.
- $\mathcal{O}_{urk}$: On input $((I, t_i), (I, t_j))$, where $t_i < t_j$, $\mathcal{C}$ performs the following processing:
  1) If $I = I^*$, and after $a_{((I,t_i),(I,t_j))}$ being added to A_{urk}, there is a path in DG_{urk} from vertex (I^*, t^*) to vertex (I, t), where $t_j \leq t$, and $(I, t) \in QI \cup CI$, as shown in Fig. 3(c), $\mathcal{C}$ outputs $\perp$ and aborts.

2) If $I = I^*$, and after $a_{((I,t_i),(I,t_j))}$ being added to A_{urk}, there is a path in $DG_{urk} \cup DG_{drk}$ from vertex (I^*, t^*) to vertex (I', t), where $D_{t_j} \leq D_t$, and $(I', t) \in QI \cup CI$, as shown in Fig. 3(d), $\mathcal{C}$ outputs $\perp$ and aborts.
3) If $I \neq I^*$, and after $a_{((I,t_i),(I,t_j))}$ being added to A_{urk}, there is a path in $DG_{urk} \cup DG_{drk}$ from vertex (I^*, t^*) to vertex (I', t), where $D_{t^*} \leq D_{t_i} \leq D_{t_j} \leq D_t$, and $(I', t) \in QI \cup CI$, as shown in Fig. 3(e), $\mathcal{C}$ outputs $\perp$ and aborts. Otherwise, $\mathcal{C}$ returns $urk_{(I,t_i)\to(I,t_j)} \leftarrow URKeyGen(sk_{(I,t_i)}, (I, t_j))$ to $\mathcal{A}$ and adds $urk_{(I,t_i)\to(I,t_j)}$ to A_{urk}, where $sk_{(I,t_i)} \leftarrow KeyGen(msk, (I, t_i))$.

- $\mathcal{O}_{drk}$: On input $((I_k, t_i), (I_l, t_j), w)$, where $D_{t_i} = D_{t_j}$, if $w \neq w^*$, $\mathcal{C}$ runs $DRKeyGen(sk_{(I_k,t_i)}, (I_l, t_j), w)$ to generate $drk_{(I_k,t_i)\to(I_l,t_j)|w}$. Otherwise, $\mathcal{C}$ performs the following processing:
1) If $I_k = I^*$, and after $a_{((I_k,t_i),(I_l,t_j))}$ being added to A_{drk}, there is a path in $DG_{urk} \cup DG_{drk}$ from vertex (I^*, t^*) to vertex (I, t), and $(I, t) \in QI \cup CI$, where $D_{t^*} \leq D_{t_i}$ and $D_{t_j} \leq D_t$, as shown in Fig. 3(f), $\mathcal{C}$ outputs $\perp$ and aborts.
2) If $I_k \neq I^*$, and after $a_{((I_k,t_i),(I_l,t_j))}$ being added to A_{drk}, there is a path in $DG_{urk} \cup DG_{drk}$ from vertex (I^*, t^*) to vertex (I, t), and $(I, t) \in QI \cup CI$, where $D_{t^*} \leq D_{t_i}$ and $D_{t_j} \leq D_t$, as shown in Fig. 3(g), $\mathcal{C}$ outputs $\perp$ and aborts. Otherwise, $\mathcal{C}$ runs $DRKeyGen(sk_{(I_k,t_i)}, (I_l, t_j), w)$ to generate $drk_{(I_k,t_i)\to(I_l,t_j)|w^*}$ and add it into A_{drk}, where $sk_{(I_k,t_i)} \leftarrow KeyGen(msk, (I_k, t_i))$.
- $\mathcal{O}_{ure}$: On input $((I, t_i), (I, t_j), C_{(I,t_i)|*})$, where $t_i < t_j$, $\mathcal{C}$ returns $UpReEnc(urk_{(I,t_i)\to(I,t_j)}, C_{(I,t_i)|*})$ to generate $C_{(I,t_j)|*}$, where $urk_{(I,t_i)\to(I,t_j)}$ is generated by $URKeyGen(sk_{(I,t_i)}, (I, t_j))$, and $sk_{(I,t_i)}$ is generated by $KeyGen(msk, (I, t_i))$.
- $\mathcal{O}_{dre}$: On input $((I_k, t_i), (I_l, t_j), C_{(I_k,t_i)|w}, w)$, where $D_{t_i} = D_{t_j}$, $\mathcal{C}$ returns $C_{(I_l,t_j)|w}$ generated from $DeReEnc(drk_{(I_k,t_i)\to(I_l,t_j)|w}, C_{(I_k,t_i)|w})$, where the delegation re-encryption key $drk_{(I_k,t_i)\to(I_l,t_j)|w}$ is generated from $DRKeyGen(sk_{(I_k,t_i)}, (I_l, t_j), w)$ and the private key $sk_{(I_k,t_i)}$ is generated from $KeyGen(msk, (I_k, t_i))$.
- $\mathcal{O}_{de}$: On input $((I, t), C_{(I,t)|w})$, $\mathcal{C}$ returns $m \leftarrow Dec(sk_{(I,t)}, C_{(I,t)|w})$, where the private key $sk_{(I,t)}$ is generated from $KeyGen(msk, (I, t))$.

Challenge. $\mathcal{A}$ submits two messages m_0, m_1 of equal length. $\mathcal{C}$ returns $C^*_{ID^*|w^*} = Enc(ID^*, w^*, m_d)$ to $\mathcal{A}$, where $C^*_{ID^*|w^*} = (C^*_0, C^*_1, C^*_2, C^*_3, C^*_4, C^*_5, C^*_6)$, $C^*_0 = svk^*$, $C^*_1 = g^{\frac{c}{c_0}}$, $C^*_2 = m_d \cdot Q$, $C^*_3 = Q^{\frac{a_0+b_0 \cdot ID^*}{c_0}}$, $C^*_4 = (p^{c \cdot s_1 \cdot w^*} \cdot p^{c \cdot s_2})^{\frac{1}{c_0}} = (g_1{}^{w^*} \cdot g_2)^{\frac{c}{c_0}}$, $C^*_5 = (p^{c \cdot s_3 \cdot svk^*} \cdot p^{c \cdot s_4})^{\frac{1}{c_0}} = (g_3{}^{svk^*} \cdot g_4)^{\frac{c}{c_0}}$, and $d \in \{0, 1\}$.

Phase 2. $\mathcal{A}$ continues issuing queries and $\mathcal{C}$ responds as follows.

- $\mathcal{O}_{sk}(I, t)$: $\mathcal{C}$ processes as Phase 1.
- $\mathcal{O}_{urk}((I, t_i), (I, t_j))$: $\mathcal{C}$ processes as Phase 1.
- $\mathcal{O}_{drk}((I_k, t_i), (I_l, t_j), w)$: $\mathcal{C}$ processes as Phase 1.
- $\mathcal{O}_{ure}((I, t_i), (I, t_j), C_{(I,t_i)|*})$: If $C_0 \neq svk^*$, $\mathcal{C}$ processes as Phase 1. Otherwise, $\mathcal{C}$ performs the following processing:
1) If $I = I^*$, and after $a_{((I,t_i),(I,t_j))}$ being added to A_{ure}, there is a path in $DG_{urk} \cup DG_{ure} \cup DG_{drk} \cup DG_{dre}$ from vertex (I^*, t^*) to vertex (I', t), where $D_{t_j} \leq D_t$, and at least one vertex in the path from vertex (I^*, t_j) to vertex (I', t) is in $QI \cup CI$, as shown in Fig. 3(h), $\mathcal{C}$ outputs $\perp$ and aborts.
2) If $I \neq I^*$, and after $a_{((I,t_i),(I,t_j))}$ being added to A_{ure}, there is a path in $DG_{urk} \cup DG_{ure} \cup DG_{drk} \cup DG_{dre}$ from vertex (I^*, t^*) to vertex (I', t), where $D_{t_j} \leq D_t$, and at least one vertex in the path from vertex (I^*, t_j) to vertex (I', t) is in $QI \cup CI$, as shown in Fig. 3(i), $\mathcal{C}$ outputs $\perp$ and aborts. Otherwise, $\mathcal{C}$ returns $C_{(I,t_j)|*}$ to $\mathcal{A}$ as Phase 1 and adds $a_{(I,t_i),(I,t_j)}$ into A_{ure}.
- $\mathcal{O}_{dre}((I_k, t_i), (I_l, t_j), C_{(I_k,t_i)|w}, w)$: If C_0 in $C_{(I_k,t_i)|w}$ is not equal to svk^* or $w \neq w^*$, $\mathcal{C}$ processes as Phase 1. Otherwise, $\mathcal{C}$ checks that, if after $a_{((I_k,t_i),(I_l,t_j))}$ being added to A_{dre}, there is a path in $DG_{urk} \cup DG_{ure} \cup DG_{drk} \cup DG_{dre}$ from vertex (I^*, t^*) to vertex (I, t), where $D_{t_j} \leq D_t$, and at least one vertex in the path from vertex (I_l, t_j) to vertex (I, t) is in $QI \cup CI$, as shown in Fig. 3(j), $\mathcal{C}$ outputs $\perp$ and aborts. Otherwise, $\mathcal{C}$ returns $C_{(I_l,t_j)|w}$ to $\mathcal{A}$ as Phase 1 and adds $a_{((I_k,t_i),(I_l,t_j))}$ into A_{dre}.
- $\mathcal{O}_{de}((I, t), C_{(I,t)|*})$: If C_0 in $C_{(I,t)|*}$ is equal to svk^*, $\mathcal{C}$ outputs $\perp$ and aborts. Otherwise, $\mathcal{C}$ processes as Phase 1.

Guess. $\mathcal{A}$ outputs a guess $d' \in \{0, 1\}$. If $d' = d$ then $\mathcal{C}$ decides that $Q = e(p, p)^{abc}$; else, Q is a random element in G_T.

If $\mathcal{C}$ does not abort during the simulation, adversary $\mathcal{A}$ will not be able to distinguish the simulation from the real attack. Assuming that $\mathcal{A}$ can break this scheme with a non-negligible advantage, then $\mathcal{C}$ can solve the DBDH problem with non-negligible advantage.

V. IMPLEMENTATION FOR SECURE CLOUD SHARING

In this section, the proposed scheme is applied to a secure cloud sharing scenario, including managements of secure data storage, data sharing (authorization), ciphertext update after changing identity (key), authorization revocation, and re-authorization. Because data encryption, decryption, and sharing are the same as other proxy re-encryption schemes. Therefore, this section mainly introduces the implementation of ciphertext update, authorization revocation, and re-authorization. In addition, for the sake of introduction, the identity update and the private key update are explained beforehand.

A. IDENTITY UPDATE AND KEY UPDATE

There are two types of identity update situations, active update and passive update. The former is due to the expiration of the identity cycle, in which the user's identity needs to be changed to the next validity period. In contrast, the latter is triggered by key leakage or compromise.

As described in Section II, the user's identity consists of a unique identifier and an expiration date. It is recorded by the PKG. When the expiration date expires, the user

TABLE 2. Characteristics comparison.

Characteristics	[17]	[19]	[22]	[23]	[24]	[25]	Proposed scheme
Ciphertext update	Yes	No	Yes	Yes	Yes	Yes	Yes
Ciphertext delegation	No	Yes	No	No	Yes	Yes	Yes
Conditional delegation	No	Yes	No	No	No	No	Yes
Delegation revocation	Yes	No	Yes	Yes	Yes	Yes	Yes
Key update	No	No	No	No	Yes	Yes	Yes
Ciphertext size constant	No	Yes	Yes	Yes	Yes	Yes	Yes
Collusion resistant	No	Yes	-	-	No	Yes	Yes
CCA-secure	No	Yes	No	Yes	No	No	Yes
Without random model	Yes	No	Yes	No	Yes	No	Yes

needs to update the identity to the next expiration date and request the PKG for the private key corresponding to the new identity. The new key generation request can also be made ahead of time to alleviate congestion. But the new identity is only valid for the next period. For example, identity $ID = Alice|20210301$ will not be valid until Marth 1, 2021, although on February 16, 2021, Alice has requested the private key for new identity $ID = Alice|20210301$.

For the situation of private key leakage, the user updates the identity to the current time immediately. For example, on February 18, 2021, Alice found that her private key corresponding to the identity $ID = Alice|20210201$ was leaked, and she set her new identity to be $ID' = Alice|20210218$ at once. Then she announced it to PKG and requested the new private key. PKG returns the new private key and refreshes the user identity information.

B. CIPHERTEXT UPDATE

After the data owner updates the identity and private key, the ciphertext corresponding to the old identity ID_i and the private key sk_{ID_i} needs to be updated to adapt to the new private key $sk_{ID_i'}$. The process is shown as follows:

Step1: The data owner runs the algorithm *URKeyGen* with input (sk_{ID_i}, ID_i') to generate a update token $urk_{ID_i \to ID_i'}$ and sends it to the proxy for ciphertext evolution.

Step2: After receiving the ciphertext evolution request from the data owner, the proxy will verify it. If the request is legitimate, the proxy runs the algorithm $UpReEnc(urk_{ID_i \to ID_i'}, C_{ID_i|*})$ to generate the new ciphertext $C_{ID_i'|*}$, which will replace the old ciphertext to store in the cloud storage.

C. AUTHORIZATION REVOCATION AND RE-AUTHORIZATION

After the ciphertext has been updated, the decryption ability of the data owner's old key has been revoked. And the old delegation token has also become invalid, which means that the previous delegation has been canceled. If the data owner needs to continue to share the data with the requester, a new delegation token needs to be generated for re-authorization or authorization renewal. The process is the same as a regular delegation.

For the leakage of the private key of the data requester, another delegation revocation is required. The process is shown as follows:

Step1: The data requester (Bob) updates his identity immediately and notices PKG and the data owner (Alice). The details of how to send the notifications and verify them are not covered in this paper.

Step2: The data owner informs the proxy to delete the delegation token previously used to authorize Bob, which means the authorization to the data requester's compromised identity has been revoked.

Step3 (optional): Enhanced authorization revocation. The data owner updates her identity and the ciphertext as in the previous subsection. In this way, the updated ciphertext cannot be accessed even if the adversary (who has broken Bob's old key) colludes with the proxy (who secretly keeps the old authorization key). This processing can enhance the system's security but increases the burden on the data owner and proxy server, so it needs to be a trade-off based on the actual situation.

Step4: Re-authorization. The data owner generates a new delegation token for the data requester.

VI. COMPARISON AND ANALYSIS

In this section, the proposed scheme is compared with several related works with the function of ciphertext evolution (update) in terms of characteristics, as shown in Table 2. It can be seen from the comparison that the proposed solution is the only one that simultaneously has many attractive functions (including access delegation, conditional delegation, delegation revocation, key update, and ciphertext evolution) and features of constant ciphertext length, collusion security, CCA security, and without random oracle. Note that the key update here refers to the underlying private key, not the decryption key derived from the time token. Suppose the underlying private key is not updated. In that case, there may be a situation where the adversary colludes with other delegatee to obtain the update token and derive a new decryption key. For details, please refer to *He et al.*'s attack [21] on Liang *et al.*'s scheme [17]. Besides, the schemes [22], [23] do not need to generate the re-encryption key, and there are not collusion attacks in their schemes, so the symbol "-" is used to indicate it.

In addition, Table 3 shows the calculation complexity of calculating update tokens, updating ciphertext, and generating re-authorization tokens. The time required for the data owner to compute the ciphertext for n users (data owner and $n-1$ sharers), the space needed to save ciphertexts on the

TABLE 3. Comparison of time and space resources required to authorize n users at different phases of several schemes.

Phases	[17]	[19]	[22]	[23]	[24]	[25]	Proposed scheme
Encryption	$n \cdot t_E$	t_E	$n \cdot t_E$	$n \cdot t_E$	t_E	t_E	t_E
Ciphertext upload	$n \cdot t_C$	t_C	$n \cdot t_C$	$n \cdot t_C$	t_C	t_C	t_C
Ciphertext storage	$n \cdot s_C$	s_C	$n \cdot s_C$	$n \cdot s_C$	s_C	s_C	s_C
Update token generation	$n \cdot t_{UK}$	-	0	0	t_{UK}	t_{UK}	t_{UK}
Ciphertext evolution	$n \cdot t_{CE}$	-	$n \cdot t_{CE}$	$n \cdot t_{CE}$	t_{CE}	t_{CE}	t_{CE}
Delegation token generation	-	$(n-1) \cdot t_{DK}$	-	-	$(n-1) \cdot t_{DK}$	$(n-1) \cdot t_{DK}$	$(n-1) \cdot t_{DK}$

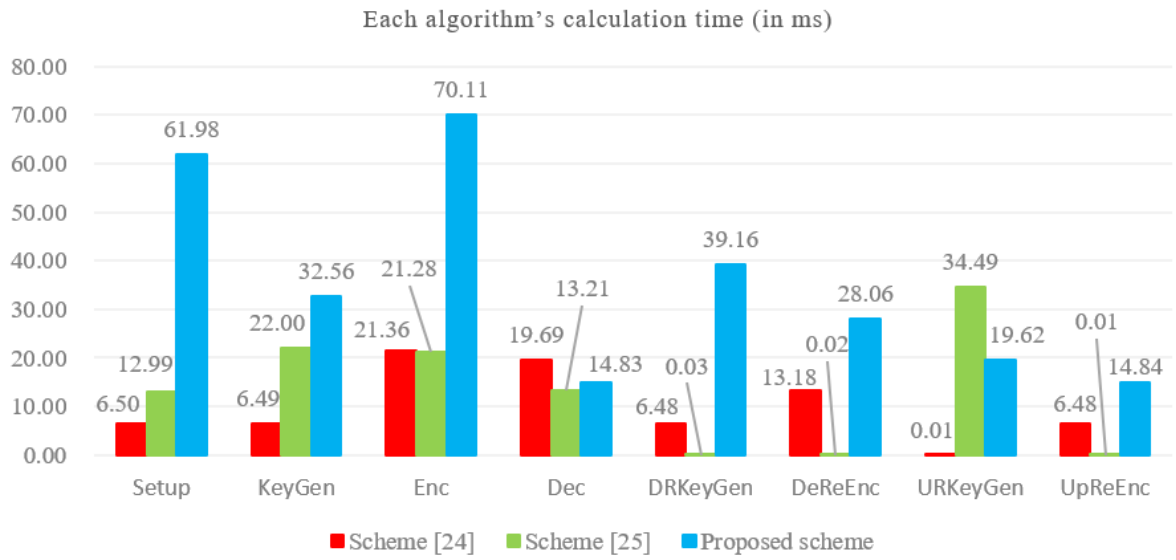

FIGURE 5. The calculation time of each algorithm in several secure data sharing schemes.

cloud server, and the time needed for users to upload shared data are also given. The t_E represents the time of encrypting data to generate the original ciphertext, t_{UK} represents update token generation time, t_{DK} represents delegation token generation time, t_{CE} represents the ciphertext evolution time, s_C represents the storage space for one ciphertext copy of data m, and t_C represents the time required to upload one copy of ciphertext to the cloud server. In schemes [17], [22], [23], the data owner encrypts the data with different requesters' identities. So the data owner needs to encrypt the data n times and generates n ciphertext copies of the data if there are n visitors (data owner and $n-1$ sharers). While in schemes [19], [24], [25] and the proposed scheme, the data owner encrypts the data with her/his own identity directly. The data only needs to be encrypted one time, and only one ciphertext copy is generated and uploaded to the cloud for storage. This ciphertext can be decrypted by the data owner, and delegated to other requesters by proxy. In Liang *et al.*' scheme [17], n update tokens to be used for updating n ciphertext copies of the same data for different requesters are computed, and n ciphertext copies are transformed. In schemes [22], [23], no update token needs to be calculated, but the n ciphertext copies need to be updated by the proxy. In scheme [24], [25] and proposed scheme, only one update token needs to be generated, and the ciphertext evolution transformation only needs to be conducted one time. In Xiong *et al.*'s scheme [19], the calculation of update token generation and ciphertext evolution are not included. In schemes [17], [22], [23], the authorization token is not required. In schemes [19], [24], [25] and the proposed scheme, $n-1$ new authorization re-encryption key need to be generated by the data owner for $n-1$ legitimate sharers after the data owner updating the ciphertext stored on the cloud.

Next, Table 4 shows the computational complexity comparison of each algorithm in proposed scheme and schemes [17], [19], [22]–[25], where t_p, t_e, t_s, t_v and t_m denote the computation time of pairing, modular exponentiation, signature, ciphertext verification and multiplication operation, respectively. And the symbol "-" represents that the scheme does not support the algorithm. The multiplication time is ignored for simplicity when multiple calculations are included simultaneously because the multiplication time (in microseconds) is far shorter than other operations (in milliseconds). Except for the newly added update key generation and ciphertext evolution algorithm, other algorithms in the proposed scheme have the same computational complexity as the scheme [19]. The computational complexity of update key generation and ciphertext evolution algorithm is less than that of delegation key generation and delegation ciphertext generation algorithm. Also, in terms of ciphertext evolution algorithm, if the cost of ciphertext verification is not taken into account, the proposed scheme has one more modular

TABLE 4. Computation complexity comparison.

Algorithms	[17]	[19]	[22]	[23]	[24]	[25]	Proposed scheme
Setup	$3t_e$	$2t_p + 3t_e$	$t_p + 2t_e$	t_e	t_m	t_e	$2t_p + 3t_e$
KeyGen	$11t_e$	$5t_e$	$10t_e$	t_e	t_e	t_e	$5t_e$
Enc	$t_p + 4t_e$	$t_p + 8t_e + t_s$	$4t_e$	$2t_p + 3t_e$	$t_p + 2t_e$	$t_p + 2t_e$	$t_p + 8t_e + t_s$
Dec	$3t_p + n(t_p + 2t_e)$	$t_p + t_e + t_v$	$3t_p$ or $6t_p$	$2t_p + t_e$	$t_p + t_e$	t_p	$t_p + t_e + t_v$
DRKeyGen	-	$6t_e$	-	-	t_e	$2t_p$	$6t_e$
DeReEnc	-	$2t_p + t_e + t_v$	-	-	t_p	t_m	$2t_p + t_e + t_v$
URKeyGen	$t_p + 10t_e$	-	-	t_e	t_m	$2t_p + 2t_e$	$3t_e$
UpReEnc	$2t_p$	-	$4t_e$	$2t_p$	t_p	t_m	$t_p + t_e + t_v$

exponentiation calculation than scheme [24], one less pairing operation than scheme [23], which has certain advantages in the term of efficiency. Compared with the scheme [25], the re-encryption key generation's complexity in the proposed scheme is lower. Therefore, it can be said that the proposed scheme not only improves the security and practicability of the system but also ensures efficiency.

Fig. 5 shows each algorithm's calculation time in several secure data sharing schemes [24], [25] with ciphertext evolution. Note that the verification time (about 92ms) is not included in the proposed scheme because schemes [24], [25] do not need to verify the ciphertext before decryption and re-encryption. These evaluations are performed on a computer with a 3.4GHz Intel Core i5-3570 processor, 8GB RAM, and the operating system is Windows 10 Professional. All the programs are based on the PBC library, where the parameter is the type A curve. Although each algorithm calculation time in the proposed scheme is more than the other two schemes, its' security (CCA-secure, collusion resistant, and without random oracle) is the highest.

VII. CONCLUSION

This paper improved Xiong *et al.*'s identity-based conditional proxy re-encryption scheme with features such as multi-purpose, constant ciphertext length, collusion security, CCA security by adding a ciphertext update function to support key update and authorization revocation. Although the function of ciphertext evolution has been added to the scheme, the efficiency and security have not been reduced. For ciphertext authorization, the efficiency of the proposed scheme is the same as the original one. Moreover, the computational complexity of the increased algorithms used to generate update re-encryption keys and ciphertext evolution is lower than that used for authorization. In addition to being applicable to secure cloud data sharing environment, the proposed scheme is also suitable to other systems that need to consider key change, which has certain practicability and security.

ACKNOWLEDGMENT

The authors are grateful to Prof. Caixue Zhou, Guangyong Gao, Ph.D. Jiaoli Shi, and Tao Li, Hyuntae Kim for many useful comments and suggestions. They also thank anonymous reviewers for helpful comments and discussions.

REFERENCES

[1] Z. Qin, H. Xiong, S. Wu, and J. Batamuliza, "A survey of proxy re-encryption for secure data sharing in cloud computing," *IEEE Trans. Services Comput.*, early access, Apr. 6, 2016, doi: 10.1109/TSC.2016.2551238.

[2] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, Espoo, Finland, 1998, pp. 127–144.

[3] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Inf. Syst. Secur.*, vol. 9, no. 1, pp. 1–30, Feb. 2006.

[4] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, Alexandria, VA, USA, 2007, pp. 185–194.

[5] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," in *Proc. Int. Workshop Public Key Cryptogr.*, Barcelona, Spain, 2008, pp. 360–379.

[6] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, Zhuhai, China, 2007, pp. 288–306.

[7] C. K. Chu and W. G. Tzeng, "Identity-based proxy re-encryption without random oracles," in *Proc. Int. Conf. Inf. Secur.*, Valparaíso, Chile, 2007, pp. 189–202.

[8] J. Shao and Z. Cao, "Multi-use unidirectional identity-based proxy re-encryption from hierarchical identity-based encryption," *Inf. Sci.*, vol. 206, pp. 83–95, Nov. 2012.

[9] J. Weng, R. H. Deng, X. Ding, C.-K. Chu, and J. Lai, "Conditional proxy re-encryption secure against chosen-ciphertext attack," in *Proc. 4th Int. Symp. Inf., Comput., Commun. Secur. (ASIACCS)*, Sydney, NSW, Australia, 2009, pp. 322–332.

[10] J. Weng and Y. Zhao, "Direct constructions of bidirectional proxy re-encryption with alleviated trust in proxy," IACR Cryptol. ePrint Arch., Tech. Rep. 2011/208, May 2011.

[11] S. Luo, Q. Shen, and Z. Chen, "Fully secure unidirectional identity-based proxy re-encryption," in *Proc. Int. Conf. Inf. Secur. Crypt.*, Seoul, South Korea, 2011, pp. 109–126.

[12] W. Feng, Z. Zhang, J. Wang, and L. Han, "A novel authorization delegation scheme for multimedia social networks by using proxy re-encryption," *Multimedia Tools Appl.*, vol. 75, no. 21, pp. 13995–14014, Nov. 2016.

[13] G. Chunpeng, Z. Liu, J. Xia, and F. Liming, "Revocable identity-based broadcast proxy re-encryption for data sharing in clouds," *IEEE Trans. Dependable Secure Comput.*, early access, Feb. 14, 2019, doi: 10.1109/TDSC.2019.2899300.

[14] C. Ge, J. Xia, A. Wu, H. Li, and Y. Wang, "A source hiding identity-based proxy reencryption scheme for wireless sensor network," *Secur. Commun. Netw.*, vol. 2018, pp. 1–8, Oct. 2018, doi: 10.1155/2018/6395362.

[15] V. Vijayakumar, M. K. Priyan, G. Ushadevi, R. Varatharajan, G. Manogaran, and P. V. Tarare, "E-health cloud security using timing enabled proxy re-encryption," *Mobile Netw. Appl.*, vol. 24, no. 3, pp. 1034–1045, Jun. 2019.

[16] *Recommendation for Key Management: Part 1—General*, Standard NIST SP 800-57 Part 1 Rev. 5, May 2020.

[17] K. Liang, J. K. Liu, D. S. Wong, and W. Susilo, "An efficient cloud-based revocable identity-based proxy re-encryption scheme for public clouds data sharing," in *Proc. 19th Eur. Symp. Res. Comput. Secur.*, Wroclaw, Poland, Sep. 2014, pp. 257–272.

[18] J. Lai, Z. Huang, M. H. Au, and X. Mao, "Constant-size CCA-secure multi-hop unidirectional proxy re-encryption from indistinguishability obfuscation," *Theor. Comput. Sci.*, vol. 847, pp. 1–16, Dec. 2020, doi: 10.1016/j.tcs.2020.09.031.

[19] H. Xiong, Y. Wang, W. Li, and C.-M. Chen, "Flexible, efficient, and secure access delegation in cloud computing," *ACM Trans. Manage. Inf. Syst.*, vol. 10, no. 1, pp. 1–20, May 2019, doi: 10.1145/3318212.

[20] K. Liang, C.-K. Chu, X. Tan, D. S. Wong, C. Tang, and J. Zhou, "Chosen-ciphertext secure multi-hop identity-based conditional proxy re-encryption with constant-size ciphertexts," *Theor. Comput. Sci.*, vol. 539, pp. 87–105, Jun. 2014.

[21] K. He, J. Weng, R. H. Deng, and J. K. Liu, "On the security of two identity-based conditional proxy re-encryption schemes," *Theor. Comput. Sci.*, vol. 652, pp. 18–27, Nov. 2016, doi: 10.1016/j.tcs.2016.08.023.
[22] C. Wang, Y. Li, J. Fang, and J. Xie, "Cloud-aided scalable revocable identity-based encryption scheme with ciphertext update," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 20, p. e4035, Oct. 2017.
[23] Y. Sun, W. Susilo, F. Zhang, and A. Fu, "CCA-secure revocable identity-based encryption with ciphertext evolution in the cloud," *IEEE Access*, vol. 6, pp. 56977–56983, 2018, doi: 10.1109/ACCESS.2018.2873019.
[24] H. Shafagh, A. Hithnawi, L. Burkhalter, P. Fischli, and S. Duquennoy, "Secure sharing of partially homomorphic encrypted IoT data," in *Proc. 15th ACM Conf. Embedded Netw. Sensor Syst.*, Delft, The Netherlands, Nov. 2017, pp. 1–14.
[25] S. Yao, R. Sankar, and I.-H. Ra, "A collusion-resistant identity-based proxy reencryption scheme with ciphertext evolution for secure cloud sharing," *Secur. Commun. Netw.*, vol. 2020, pp. 1–16, Oct. 2020, doi: 10.1155/2020/8833693.
[26] M. Bellare and S. Shoup, "Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles," in *Proc. Int. Workshop Public Key Cryptogr.*, Beijing, China, 2007, pp. 201–216.

SHIMAO YAO received the B.E. degree in information and computing science from the Nanjing University of Aeronautics and Astronautics (NUAA), China, in 2003, and the M.S. degree in electronic and communications engineering from the Dalian University of Technology, China, in 2010. He is currently pursuing the Ph.D. degree with the Kunsan National University, South Korea. He is also employed as a Lecturer with Jiujiang University, China. His research interests include information security and computer networks.

RALPH VOLTAIRE J. DAYOT received the B.Sc. degree in information technology and the master's degree in information technology from West Visayas State University, Philippines, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree with Kunsan National University, South Korea. He is also employed as a Lecturer with West Visayas State University. His research interests include machine learning, artificial intelligence, transactive energy, and edge computing.

HYUNG-JIN KIM received the M.S. and Ph.D. degrees in information and telecommunication engineering from Kunsan National University, Kunsan, South Korea, in 1999 and 2004, respectively. He is currently a Professor with the Department of IT Applied System Engineering, Chonbuk National University, Jeonju, South Korea. His research interests include multimedia systems, sensor networks, image processing, cross-layer design, robot vision, and human–robot interaction.

IN-HO RA (Member, IEEE) received the B.S. degree in computer engineering from Seoul National University, in 1986, and the M.S. and Ph.D. degrees in computer engineering from Chung-Ang University, Seoul, South Korea, in 1991 and 1995, respectively. He is currently a Professor with the Department of Electronic and Information Engineering, Kunsan National University, Gunsan, South Korea. From 2007 to 2008, he was a Visiting Scholar with the University of South Florida, Tampa, FL, USA. His research interests include mobile wireless communication networks, sensor networks, middleware design, cross-layer design, quality-of-service management integration of sensor networks, and social networks.