Hindawi
Security and Communication Networks
Volume 2020, Article ID 8833693, 16 pages
https://doi.org/10.1155/2020/8833693

WILEY | Hindawi

Research Article

A Collusion-Resistant Identity-Based Proxy Reencryption Scheme with Ciphertext Evolution for Secure Cloud Sharing

Shimao Yao,[1,2] Ravi Sankar,[3] and In-Ho Ra [2]

[1]*School of Information Science and Technology, Jiujiang University, Jiujiang 322005, China*
[2]*Information and Communication Engineering, Kunsan National University, Gunsan 54150, Republic of Korea*
[3]*Department of Electrical Engineering, University of South Florida, Tampa 33620, USA*

Correspondence should be addressed to In-Ho Ra; ihra@kunsan.ac.kr

Received 7 April 2020; Revised 21 August 2020; Accepted 3 September 2020; Published 14 October 2020

Academic Editor: Clemente Galdi

In order to solve the challenges of user data security in the cloud computing (storage) environment, many encryption solutions with different features have been presented. Among them, proxy reencryption (PRE) based on public-key infrastructure (PKI) is a promising technology for secure cloud sharing. And identity-based proxy reencryption (IBPRE), which uses identity as the public key, eliminates burdensome certificate management and is, therefore, more preferable. However, most of the current IBPRE schemes only focus on the processing of data sharing while overlooking the functions of authorization revocation and ciphertext update, which are more closely related to the security of data itself. Moreover, the few existing schemes that involve ciphertext update turn out to be impractical because the length of ciphertext increases with the reencryption of ciphertext. In this paper, an improved IBPRE scheme, which provides improvements on the inadequacies of the scheme proposed by Ateniese et al. especially in terms of collusion safety and ciphertext evolution, is proposed. To the best of our knowledge, this is a practical IBPRE scheme integrating the functions of access authorization, delegation revocation, ciphertext update, reauthorization, and conditional reservation delegation. The proposed technique has high practicability in the scenario where a large number of ciphertexts need to be updated synchronously. Lastly, the comparative analysis and simulation results show that the two reencryption algorithms in the proposed scheme have the shortest computing time than other schemes.

1. Introduction

With the advancement and prevalence of cloud computing technology, more and more users opt to store their user data on cloud servers due to its convenience and ubiquity of access. However, different from local storage, the control and ownership of data in cloud storage are separated, and the cloud server is not completely trustworthy. Outsourcing data to the cloud will face many security issues and challenges in data integrity, confidentiality, access authorization, ciphertext search, and ciphertext evolution. To solve these security problems, many scholars have conducted numerous relevant researches and proposed various solutions. For example, there are cloud storage public auditing schemes for data integrity [1], proxy reencryption (PRE) schemes for encrypted data authorization [2], identity-based encryption (IBE) schemes with keyword search [3], IBE schemes supporting ciphertext update evolution [4], and lattice-based encryption (LBE) schemes to achieve postquantum security [5]. Some of these solutions addressed multiple aspects of data security, while others focused on only one or two aspects of data security. For example, the PRE scheme [6] assumes that the proxy is semitrusted and will follow the protocol and not tamper with the user's data to ensure data integrity. It only focuses on confidentiality and access authorization of the data.

It is too complex and unrealistic to consider all security issues in one cloud storage solution, but it is feasible and necessary to assemble some security requirements into one system. For example, in a personal data public cloud sharing application scenario, it urgently requires an encryption solution, which includes functions of access authorization,

key update, ciphertext update, authorization revocation, reauthorization, conditional reservation authorization, and avoids complex certificate management. The consideration is based on the reasons as follows:

(1) Access authorization is the primary function of secure cloud data sharing. If a user's encrypted private data stored in the cloud are accessed only by himself but not shared with others, it is not as convenient as carrying a large mobile storage device around. At present, to realize secure authorization of encrypted data, PRE technology is considered as the most promising solution. Also, identity-based proxy reencryption (IBPRE) uses identity as the public key, eliminates burdensome certificate management, and is, therefore, more preferable.

(2) Authorization revocation is also a function that must be considered in the personal data cloud sharing scheme. If a user's identity has expired or his key has been broken, his access permission should be revoked.

(3) The necessity of reauthorization. If the original authorization expires or becomes invalid after the ciphertext is updated or the key is updated, the authorization must be renewed for the legitimate user.

(4) The need for the key update. For cryptography systems, its security relies on the protection of the key. According to the Special Publication SP-80057 [7] issued by the National Institute of Technology (NIST), cryptograph-based ciphertext key has a strict life cycle; that is, the key needs to be updated after a certain period.

(5) The requirement of ciphertext update. There are two types of ciphertext update. The first one is passive update. If the authorized access permission is revoked, the encrypted ciphertext should be renewed to prevent the old authorization key and the revoked visitor from continuing to access the ciphertext. The second one is active update, also be termed ciphertext evolution. If the delegator updates his encryption key, the ciphertext needs to evolve synchronously.

(6) Special requirements for conditional reservation authorization. Sometimes, the user wants to specify the requestor's access right at the reserved time to implement access control. For instance, one user with a solar energy collection system wants to sell the excess energy to others in the microgrid. Similarly, other prosumers also wish to sell their excess energy. These suppliers will share their energy information to an energy consumer, who cannot access the report containing the energy price in advance. Otherwise, it may lead to malicious bidding or collusion.

As far as we know, the IBPRE scheme can basically achieve access authorization without complicated certificate management. There are also some proposals that have supported conditional authorization [8, 9], some systems that have achieved authorization revocation [10, 11], and some schemes that have even considered the function of updating ciphertext [10]. However, no satisfactory solution has been found to meet all of the above needs. Therefore, this motivates our work to find a practical IBPRE scheme with functions of access authorization, ciphertext update, and delegation revocation.

1.1. Related Work. PRE allows a proxy to convert a ciphertext under one user's (delegator's) public key into another ciphertext which can be decrypted with another user's (delegatee's) private key, without disclosing the underlying plaintext and secret key information. Since Blaze et al. [12] proposed the first PRE scheme, a large number of PRE schemes with different characteristics have been presented. In [13], Nuñez et al. reviewed, compared, and analyzed the main PRE researches. Their study included PKI-based proxy reencryption [14–16], identity-based proxy reencryption (IBPRE) [17, 18], attribute-based proxy reencryption (ABPRE) [19, 20], and lattice-based proxy reencryption (LBPRE) [21]. Among these studies, the IBPRE scheme is presented to be an important research direction. It combines identity-based encryption (IBE) [22, 23] and PRE [12], where the user's identity is used as the public key for encryption, avoiding the complex public-key certificate management which is beneficial in several scenarios.

Green and Ateniese [18] put forward the first IBPRE scheme, in which two noncollusion safe approaches were introduced, and some promising applications of IBPRE were mentioned. Subsequently, many improvements and applications were proposed to meet various needs and shortages of previous solutions. Wang et al. [24] proposed two IBPRE schemes that could resist collusion attacks. The first one has no ciphertext expansion, while the second one achieves CCA security. Wang et al. [25] gave an improved multiuse IBPRE scheme, which achieves CCA2 security in the random oracle model. Their solution gives a confirmable answer to the open problem mentioned in [18]. Shao and Cao [26] introduced the first CCA-secure and collusion-safe IBPRE scheme in the standard model. Xu et al. [8] presented a conditional identity-based broadcast PRE scheme and applied it to cloud e-mail. Zhou et al. [27] proposed an IBPRE scheme with version 2, which provides a ciphertext transformation from complicated identity-based broadcast encryption (IBBE) to simple IBE. Ge et al. [9] presented a secure fine-grained identity-based broadcast PRE scheme for encrypted microvideo sharing. The schemes mentioned above analyzed the efficiency, security, and access control of the algorithms in detail but did not consider the functions of permission revocation and ciphertext evolution.

Liang et al. [10] proposed an efficient cloud-based revocable IBPRE scheme for periodic key and ciphertext updates by updating time tokens, in which the length of ciphertext increases linearly with the number of times of reencryption. Sun et al. [4] proposed a CCA-secure revocable IBE with ciphertext evolution for data sharing in cloud

storage, which emphasizes that the size of the ciphertext in the cloud remains in constant size regardless of evolutions. However, their approach is not based on PRE. The ciphertext stored in the cloud is encrypted by the data owner using the identity of the requester instead of his own identity, which means that there are multiple ciphertexts stored on the cloud corresponding to different requesters, instead of only one ciphertext in the PRE system. Shafagh et al. [6] realized a project that includes functions of authorization, revocation, key update, and ciphertext update in PKI-based architecture, which needs complex certificate management.

Intuitively, IBPRE (such as in [10, 25, 26]) can be used directly for updating ciphertext, but usually ciphertext grows with the number of encryption times, which is not practical.

1.2. Our Contributions. Inspired by the work presented in [6], we propose an improved IBPRE scheme, which includes the functions above for a secure personal data cloud sharing application. This improved approach has the characteristics of noninteractivity, unidirectionality, collusion safety, ciphertext optimization, and multiuse and nontransferability in the random oracle model. Moreover, the ciphertext update (reencryption) operations can be executed multiple times, and the ciphertext length remains the same. The ciphertext reencrypted (delegated) to the delegate, however, cannot be reencrypted (reauthorized). The improvements are based on Green and Ateniese [18] and aim to realize secure user data sharing on cloud servers by combining essential characteristics of data sharing, ciphertext updating, and attribute-based access permission granting and revocation. Furthermore, the scheme also highlights the properties of multiuse and collusion-resistant and the optimization of reencryption performance (that shortens reencryption time; ensure efficiency when many users access data, or much ciphertext updated concurrently). The main contributions of this paper are as follows:

(1) Provide improvements in the work proposed in [18] that focuses on achieving collusion safety and multiuse without ciphertext expansion and minimizing reencryption time.

(2) Propose a practical IBPRE scheme that includes functions of access authorization, delegation revocation, ciphertext update, reauthorization, and conditional reservation delegation to implement secure cloud data sharing.

(3) Apply the improved IBPRE scheme to a practical secure user data sharing application and provided an analysis comprehensively.

The rest of the paper is organized as follows. Section 2 describes some of the cryptographic primitives, definitions, and some properties of the proxy reencryption. The system model and the assumptions of the proposed scheme are given in Section 3. The improved IBPRE algorithm is described in detail in Section 4, and the application of the improvements is deployed to a secure cloud data sharing scenario presented in Section 5. A comparison and analysis of the proposed scheme and existing schemes are provided in Section 6, and finally, the conclusion is given in Section 7.

2. Preliminaries

2.1. Bilinear Map and Decisional Bilinear Diffie–Hellman Problem [25]

Definition 1 (bilinear map). Let G_1 and G_2 are two cyclic groups of the same prime order q and g be a generator of G_1. We say that e: $G_1 \times G_1 \longrightarrow G_2$ is a bilinear map if it satisfies the following properties:

(1) Bilinear: for all $a, b \in \mathbb{Z}_q^*$ and $g \in G_1$, $e(g^a, g^b) = e(g, g)^{ab}$.

(2) Nondegenerate: $e(g, g) \neq 1_{G_2}$, where 1_{G_2} is the unit of G_2.

(3) Computability: e can be efficiently computed.

Definition 2 (decisional bilinear Diffie–Hellman (DBDH) problem). Let G_1 and G_2 are two cyclic groups of the same prime order q and g be a generator of G_1. Support that e: $G_1 \times G_1 \longrightarrow G_2$ is a bilinear map. The decisional bilinear Diffie–Hellman (DBDH) problem is to decide, given a tuple of values $(g, g^a, g^b, g^c, T) \in G_1^4 \times G_2$ (where $a, b, c \in_R \mathbb{Z}_q^*$), whether $T = e(g, g)^{abc}$ holds.

Let k be a security parameter of sufficient size. Formally, we say that the DBDH assumption holds in G_1, G_2, e, if for all probabilistic polynomial time (PPT) algorithm A, the following condition is true, where $\nu(\cdot)$ is defined as a negligible function:

$$\left|\Pr\left[a, b, c \longleftarrow_R \mathbb{Z}_q^*; 1 \longleftarrow \mathrm{A}\left(g, g^a, g^b, g^c, e(g, g)^{abc}\right)\right] - \Pr\left[a, b, c \longleftarrow_R \mathbb{Z}_q^*; T \longleftarrow_R G_2; 1 \longleftarrow A\left(g, g^a, g^b, g^c, T\right)\right]\right| \leq \nu(k). \tag{1}$$

2.2. Definition and Security Model of IBPRE-CE

Definition 3 (identity-based proxy reencryption with ciphertext evolution). An identity-based proxy reencryption scheme with ciphertext evolution (IBPRE-CE) is a tuple of algorithms (Setup, KeyGen, Encrypt, RKGen_1, RKGen_2, $\mathrm{Reencrypt}_1$, $\mathrm{Reencrypt}_2$, $\mathrm{Decrypt}_1$, and $\mathrm{Decrypt}_2$) as follows:

(1) Setup (1^k): taking a security parameter k as input, this algorithm generates both master public parameters

(mpp) and master secret key (msk). The mpp is distributed to all participants, while the msk is kept secretly.

(2) KeyGen (mpp, msk, id): taking mpp, msk, and an identity id as input, this algorithm outputs a corresponding private key $\mathrm{sk}_{\mathrm{id}}$.

(3) Encrypt (mpp, id, m): taking mpp, identity id, and message m as input, this algorithm computes an original ciphertext c_{id}.

(4) RKGen_1 (mpp, $\mathrm{sk}_{\mathrm{id}_1}$, id_1, id_2, C_1): taking mpp, $\mathrm{sk}_{\mathrm{id}_1}$, identities ($\mathrm{id}_1$, id_2), and an element of ciphertext C_1 as input, this algorithm calculates a delegation token $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2}$.

(5) RKGen_2 (mpp, $\mathrm{sk}_{\mathrm{id}_1}$, id_1', C_1): taking mpp, old private key $\mathrm{sk}_{\mathrm{id}_1}$, a new identity id_1', and an element of ciphertext C_1 as input, this algorithm produces an update key $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_1'}$.

(6) $\mathrm{Reencrypt}_1$ (mpp, $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2}$, c_{id_1}): taking mpp, delegation key $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2}$, and ciphertext c_{id_1} as input, this algorithm converts the ciphertext c_{id_1} into a delegation ciphertext c_{id_2}.

(7) $\mathrm{Reencrypt}_1$ (mpp, $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_1'}$, c_{id_1}): taking mpp, update key $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_1'}$, and old ciphertext c_{id_1} as input, this algorithm updates the ciphertext c_{id_1} into a new ciphertext $c_{\mathrm{id}_1'}$.

(8) $\mathrm{Decrypt}_1$ (mpp, $\mathrm{sk}_{\mathrm{id}_1}$, c_{id_1}): taking mpp, delegator's (latest) private key $\mathrm{sk}_{\mathrm{id}_1}$, and ciphertext c_{id_1} as input, this algorithm outputs corresponding plaintext or an error flag $\perp$.

(9) $\mathrm{Decrypt}_2$ (mpp, $\mathrm{sk}_{\mathrm{id}_2}$, c_{id_2}): taking mpp, delegatee's private key $\mathrm{sk}_{\mathrm{id}_2}$, and delegation ciphertext c_{id_2} as input, this algorithm outputs corresponding plaintext or an error flag $\perp$.

We say that an IBPRE-CE scheme is consistent if for any valid identities id_1, id_2, and id_1', their secret keys $\mathrm{sk}_{\mathrm{id}_1}$, $\mathrm{sk}_{\mathrm{id}_2}$, and $\mathrm{sk}_{\mathrm{id}_1'}$ (generated by KeyGen), the corresponding reencryption key $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_1'}$ and $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2}$ (generated by RKGen_2 and RKGen_1), and ciphertexts c_{id_1}, c_{id_2}, and $c_{\mathrm{id}_1'}$ (computed by Encrypt, $\mathrm{Reencrypt}_1$, and $\mathrm{Reencrypt}_2$), the following equations hold:

① $\mathrm{Decrypt}_1$ (mpp, $\mathrm{sk}_{\mathrm{id}_1}$, Encrypt (mpp, id_1, m)) = m.

② $\mathrm{Decrypt}_1$ (mpp, $\mathrm{sk}_{\mathrm{id}_1'}$, $\mathrm{Reencrypt}_2$ (mpp, $\mathrm{rk}_{\mathrm{id}_1} \longrightarrow \mathrm{id}_1'$, Encrypt (mpp, id_1, m))) = m.

③ $\mathrm{Decrypt}_2$ (mpp, $\mathrm{sk}_{\mathrm{id}_2}$, $\mathrm{Reencrypt}_1$ (mpp, $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2}$, Encrypt (mpp, id_1, m))) = m.

Definition 4 (IND-PrID-CPA security of IBPRE-CE). An IBPRE-CE scheme is indistinguishable against chosen plaintext and identity attack (IND-PrID-CPA) secure if no probabilistic polynomial time (PPT) adversary A wins the following game with a nonnegligible advantage:

(1) Setup. The challenger C runs Setup (1^k) to generate the master public parameters (mpp) and the master secret key (msk). The mpp is sent to A, while the msk is kept securely.

(2) Phase 1. A issues some private key queries $\mathcal{O}_{\mathrm{sk}}$, updates reencryption key queries $\mathcal{O}_{\mathrm{uk}}$, and delegation reencryption key queries $\mathcal{O}_{\mathrm{dk}}$ as follows:

(i) On receiving any query of the form ($\mathcal{O}_{\mathrm{sk}}$, id), C returns $\mathrm{sk}_{\mathrm{id}} \leftarrow$ KeyGen (mpp, msk, id) to A.

(ii) On receiving any query of the form ($\mathcal{O}_{\mathrm{dk}}$, id_1, id_2, C_1), C returns $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2} \leftarrow \mathrm{RKGen}_1$ (mpp, $\mathrm{sk}_{\mathrm{id}_1}$, id_1, id_2, C_1) to A, where $\mathrm{sk}_{\mathrm{id}_1} \leftarrow$ KeyGen (mpp, msk, id_1) and C_1 is an element of ciphertext.

(iii) On receiving any query of the form ($\mathcal{O}_{\mathrm{uk}}$, id_1, id_1', C_1), C returns $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_1} \leftarrow \mathrm{RKGen}_2$ (mpp, $\mathrm{sk}_{\mathrm{id}_1}$, id_1', C_1) to A, where $\mathrm{sk}_{\mathrm{id}_1} \leftarrow$ KeyGen (mpp, msk, id_1) and C_1 is an element of ciphertext.

(4) Challenge. At the end of phase 1, A outputs two equal-length messages m_0 and m_1, and a challenging identity id^* which does not exist in trivial decryption (e.g., A holds a private key of id^* or some transformation keys which are update reencryption keys or delegation reencryption keys from id^* to id and the decryption key of id). The challenger C returns a challenge ciphertext $c^* = \mathrm{Encrypt}(\mathrm{mpp}, \mathrm{id}^*, m_i)$, where $i \in_R \{0, 1\}$.

(5) Phase 2. A adaptively issues queries as phase 1 with some restrictions that the private key of id^* and any trivial decryption key query ($\mathcal{O}_{\mathrm{sk}}$, $\mathcal{O}_{\mathrm{uk}}$, $\mathcal{O}_{\mathrm{dk}}$) have never been queried. The restrictions are as follows:

(i) The private key corresponding to id^* has not been queried.

(ii) If a series of update key queries which originate from id^* to id_1, next from id_1 to id_2,..., from id_{n-1} to id_n, have been issued, the private key of any id in set ID = $\{\mathrm{id}_1, \mathrm{id}_2, \ldots \mathrm{id}_n\}$ has not been queried.

(iii) If the delegation key query from id^* to id_j has been issued, the private key of id_j has not been queried.

(iv) If a series of update key queries which originate from id^* to id_1, next from id_1 to id_2,..., from id_{n-1} to id_n have been issued, and delegation key from any id in set ID = $\{\mathrm{id}_1, \mathrm{id}_2, \ldots \mathrm{id}_n\}$ to id_j has been queried, the private key of id_j has not been queried.

(6) Guess. A gives a guess $i' \in \{0, 1\}$.

The advantage of A in the above game is defined by $\varepsilon = |\Pr[i' = i] - (1/2)|$.

2.3. Properties of Proxy Reencryption. In literature [13], Nuñez et al. summarized several security concepts and properties of PRE and described that the existing schemes

are either unidirectional or bidirectional, single use or multiuse, collusion safe or not, transferable or nontransferable, transitive or nontransitive, identity-based or not, and interactive or noninteractive. In addition, based on the achieved notion of security, PRE schemes can achieve chosen plaintext attacks (CPAs) security or chosen ciphertext attacks (CCAs) security. For clarity, this subsection briefly reiterates some of these properties, in particular, giving focus to those that are provided by the proposed scheme and are relevant for practical instantiations of IBPRE:

(1) Noninteractivity: this property means that the user does not need to interact with the requester or trusted third party because there is no need for their secret information (e.g., private key or master system key) to generate the reencryption key. With this property, the user can delegate access permission to the requester without interaction and can even assign a requester's decryption condition associated with its identity.

(2) Multiuse capability: the ciphertext can be converted multiple times, e.g., from the ciphertext c_{id_1} to ciphertext c_{id_2} and from ciphertext c_{id_2} to ciphertext c_{id_3}. Therefore, the data owner can update the ciphertext multiple times as needed.

(3) Collusion safety: in this characteristic, even if the delegatee colludes with the proxy, the delegator's private key does not be compromised. And we can denote it as $rk_{id_1 \longrightarrow id_2} + sk_{id_2} \nrightarrow sk_{id_1}$, where $rk_{id_1 \longrightarrow id_2}$ is the reencryption key known by the proxy, and sk_{id_2} is id_2's (delegatee's) private key, and sk_{id_1} is id_1's (delegator's) private key.

(4) Nontransitivity: in this characteristic, the proxy (or two colluded proxies) cannot derive a reencryption key from two continuous reencryption keys. For example, $rk_{id_1 \longrightarrow id_2}$ and $rk_{id_2 \longrightarrow id_3}$ can get $rk_{id_1 \longrightarrow id_3}$.

(5) Nontransferability: this property means that a delegatee (even if colludes with the proxy) cannot transfer the decryption ability to a third party or entity. For example, user id_1 generates the reencryption key $rk_{id_1 \longrightarrow id_2}$ to delegate the decryption capacity to the requestor id_2 who can decrypt the reencrypted ciphertext but cannot generate a new delegation key which originates from id_1 (i.e., $rk_{id_1 \longrightarrow id_3}$) or reencrypt the delegation ciphertext to generate a new ciphertext which can be decrypted by a third entity with the private key sk_{id_3}.

(6) Unidirectionality: with the reencryption key $rk_{id_1 \longrightarrow id_2}$, the proxy can convert user id_1's ciphertext to another one that can be decrypted by the user id_2, but not vice versa.

(7) Ciphertext optimality: the ciphertext does not expand upon reencryption. This property means that the size of the ciphertext will be constant when transformed.

(8) Key optimality: the requester needs only one private key to decrypt the ciphertext encrypted by himself or the reencrypted ciphertext comes from different proxies.

Besides the characteristics described above, it is also worthy to note that there are other properties identified to be essential for some applications of PRE introduced in [13, 14, 18, 24].

3. System Model and Assumptions

3.1. System Model. As shown in Figure 1, the system consists of the user (data owner and delegator), the requester (delegatee), the private key generator (PKG), and the cloud server (proxy).

The PKG is responsible for generating the master public parameters and master secret key of the system, so it should be a trusted third party. All the participants in the system can get the master public parameters, but only PKG knows the master secret key. In addition, the PKG will generate a corresponding private key when it receives a legal key generation request for a user's identity. Then, the generated private key (and the master public parameters) will be sent to the user securely.

The user is assumed to subscribe to a cloud service from a cloud service provider for user data storage and/or data processing. After getting the public parameters from the PKG, the user can encrypt the user data with the identity before uploading it to the cloud server for storage. The user can download and decrypt the ciphertext with his own secret key. If the data needs to be shared with some requesters (e.g., friends or cooperators), the user needs to generate a corresponding reencryption key (a.k.a. delegation key). This reencryption key also needs to be sent to the proxy secretly. Moreover, the user generates an update key, which is also a reencryption key for updating the ciphertext on the cloud server through the proxy.

The requester may be a friend or a cooperative partner who wants to access some of the data encrypted by the data owner and stored on a cloud server. The requester, who is a valid delegatee, can decrypt the delegation ciphertext converted by the proxy to get the delegator's data. To reduce the complexity, the authentication of the requester is not considered in this work.

The proxy stores the ciphertext encrypted by the user and transforms the ciphertext with the reencryption key. If the reencryption key is for updating ciphertext, the proxy transforms the ciphertext to a new ciphertext directly. If the reencryption key is for delegation, the proxy will keep it secret and use it to generate a new delegation ciphertext when the legitimate delegatee requests the user data. Finally, the proposed scheme assumes that the cloud server only keeps one copy of the data.

3.2. System Security Assumptions. The proposed work assumes that the user's private key can be transported secretly from the PKG to the user. The user's delegation key can be securely sent to the proxy (cloud server) too. The cloud service is robust but not wholly trusted; that is, the user data and delegation key stored on it will not be lost due to some

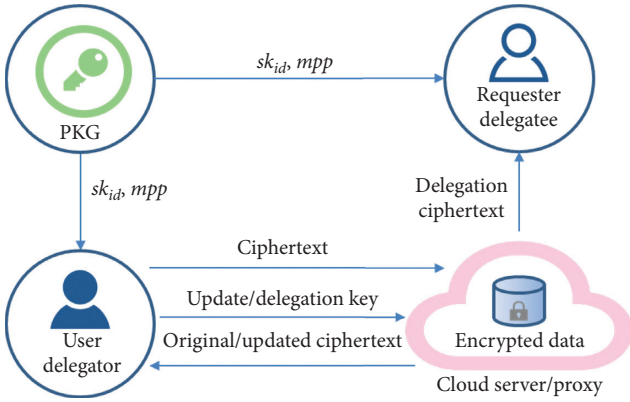

Figure 1: System model.

physical disasters or not be modified, but the cloud server or its administrators because of several reasons (curiosity or specific purpose) are interested in obtaining user data as well as some of the user's key. Moreover, the cloud servers might collude with some malicious users, or a former authorized user whose permission has been revoked is also considered in the improvements. All entities in the system will follow the communication protocol that will respond to the request correctly.

4. Proposed Scheme

The differences between the proposed scheme and the original scheme in [18] exist in three aspects: reencryption key generation algorithm, reencryption algorithm, and decryption algorithm:

(1) There are two reencryption key generation algorithms in our proposal. The first one (RKGen_1) improves the key generation algorithm of the original scheme by replacing the multiplication operation in the function with a bilinear mapping operation. In this way, if the proxy colludes with the delegatee, they can only get the bilinear pair, but not the private key. Thus, this achieves collusion resistance. The second one (RKGen_2) is a new reencryption key generation algorithm, wherein the generated key is used for updating ciphertext and is discussed in detail in Section 5.

(2) Our approach has two reencryption algorithms with different functions, while the original scheme only has one reencryption algorithm used for delegation. The first reencryption algorithm $\mathrm{REncrypt}_1$ used to calculate the delegation ciphertext is single use. The newly added one $\mathrm{REncrypt}_2$ used for ciphertext update is multiuse. We utilize these two algorithms to perfectly solve the contradiction between requiring multiple reencryption and not allowing reencryption.

(3) Two decryption algorithms are used in the proposed system. The first one $\mathrm{Decrypt}_1$ is used by the data owner to decrypt ciphertext or updated ciphertext which have been reencrypted for several times. The second one $\mathrm{Decrypt}_2$ is utilized by the delegatee to decrypt the delegated ciphertext. In the original scheme, however, there is only one decryption algorithm used to decrypt original (level-1) ciphertext and reencrypted (level-n, $n > 1$) ciphertext. And for the delegation ciphertext, the decryption needs to be performed recursively. Firstly, the latter two terms of the ciphertext are decrypted with the private key of the delegatee to get the random number selected in the calculation of the reencryption key. Then, the random number is used to calculate the plaintext. However, in the proposed scheme, the decryption object is an optimized ciphertext (ciphertext is not extended when reencrypting), so there is no need for recursive decryption.

For the sake of simplicity and comparison of the Green and Ateniese [18] (shown in Figure 2) and the proposed scheme (shown in Figure 3), a description and a detailed discussion are given below.

4.1. A Brief Description of Original Scheme. The original scheme has six algorithms as follows:

(1) Setup: let $e: G_1 \times G_1 \longrightarrow G_2$ be a bilinear map, where G_1 and G_2 are two cyclic groups of the same prime order q and g is a generator of G_1. Select two hash functions $\mathscr{H}_1: \{0,1\}^* \longrightarrow G_1$ and $\mathscr{H}_2: G_2 \longrightarrow G_1$. Randomly select $s \in \mathbb{Z}_q^*$, set the master secret key $\mathrm{msk} = s$, and output the master public parameters $\mathrm{mpp} = (G_1, G_2, q, g, g^s, \mathscr{H}_1, \mathscr{H}_2)$.

(2) $\mathrm{KeyGen}(\mathrm{mpp}, \mathrm{msk}, \mathrm{id})$: providing mpp, msk, and identity $\mathrm{id} \in \{0,1\}^*$ as input, the algorithm outputs a corresponding private key $\mathrm{sk}_{\mathrm{id}} = \mathscr{H}_1(\mathrm{id})^s$.

(3) $\mathrm{Encrypt}(\mathrm{mpp}, \mathrm{id}, m)$: providing mpp, identity id, and data m as input, randomly select $r \in \mathbb{Z}_q^*$, and the algorithm outputs a corresponding ciphertext $c_{\mathrm{id}} = (g^r, m \cdot e(g^s, \mathscr{H}_1(\mathrm{id}))^r)$.

(4) $\mathrm{RKGen}(\mathrm{mpp}, \mathrm{sk}_{id_1}, \mathrm{id}_2)$: providing mpp, private key $\mathrm{sk}_{\mathrm{id}_1}$, and identity id_2 as input, randomly select element X from group G_2, and the algorithm computes $R_1, R_2 = \mathrm{Encrypt}(\mathrm{mpp}, \mathrm{id}_2, X)$ and outputs a reencryption key $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2} = (R_1, R_2, R_3)$, where $R_3 = (\mathscr{H}_2(X)/\mathrm{sk}_{\mathrm{id}_1})$.

(5) $\mathrm{REncrypt}(\mathrm{mpp}, \mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2}, c_{\mathrm{id}_1})$: providing mpp, the reencryption key $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2}$, and ciphertext c_{id_1} as input:

① If $c_{\mathrm{id}_1} = (C_1, C_2)$ is a level-1 ciphertext (original ciphertext), the algorithm computes $C_2' = C_2 \cdot e(C_1, R_3)$, $C_3 = R_1$, and $C_4 = R_2$ and outputs ciphertext $c_{\mathrm{id}_2} = (C_1, C_2', C_3, C_4)$.

② If $c_{\mathrm{id}_1} = (C_1, C_2, \ldots, C_{2n-1}, C_{2n})$ is a level-n ($n > 1$) ciphertext, the algorithm computes $C_{2n}' = C_{2n} \cdot e(C_{2n-1}, R_3)$, $C_{2n+1} = R_1$, and $C_{2n+2} = R_2$ and outputs ciphertext $c_{\mathrm{id}_2} = (C_1, C_2, \ldots, C_{2n-1}, C_{2n}', C_{2n+1}, C_{2n+2})$.

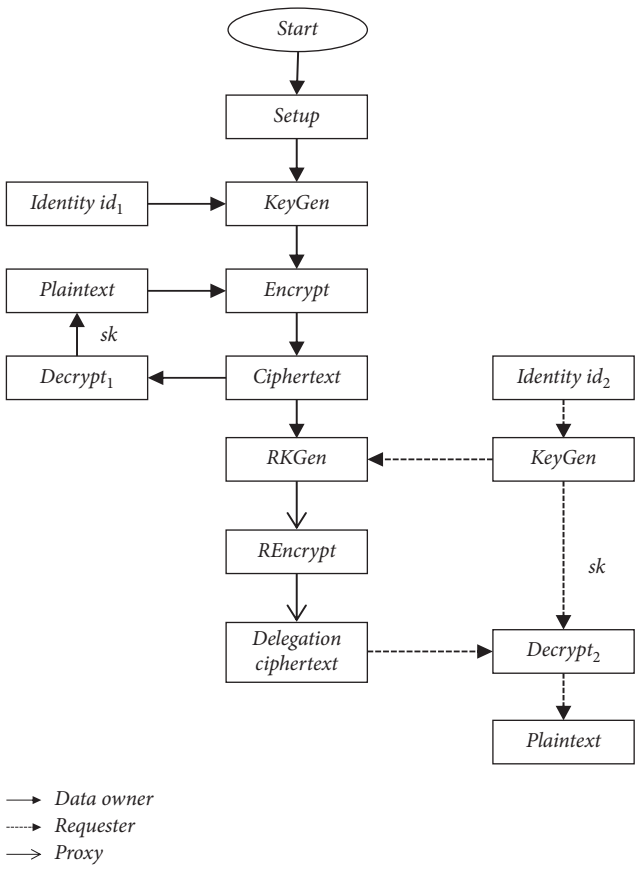

Figure 2: Process flow of the original scheme.

(6) Decrypt(mpp, $\mathrm{sk}_{\mathrm{id}}$, c_{id}): providing mpp, private key $\mathrm{sk}_{\mathrm{id}}$, and ciphertext c_{id} as input:

① If $c_{\mathrm{id}} = (C_1, C_2)$ is a level-1 ciphertext (original ciphertext), the algorithm computes $m = (C_2/e(C_1, \mathrm{sk}_{\mathrm{id}}))$.

② If $c_{\mathrm{id}} = (C_1, C_2, \ldots, C_{2n-1}, C_{2n})$ is a level-n ciphertext, consider the combination (C_{2n-1}, C_{2n}) as a level-1 ciphertext, the algorithm computes $X_n = (C_{2n}/e(C_{2n-1}, \mathrm{sk}_{\mathrm{id}}))$ and $X_i = (C_{2i}/e(C_{2i-1}, \mathscr{H}_2(X_{i+1})))$, where i is evaluated from $n-1$ to 2. Finally, the algorithm computes $m = (C_2/e(C_1, \mathscr{H}_2(X_2)))$.

The original ciphertext contains two elements, the length of which increases by two elements for each reencryption. Theoretically, the ciphertext can be reencrypted any number of times.

4.2. Detailed Design of the Proposed Scheme. There are nine algorithms in the proposed scheme, and the process flow is depicted in Figure 3. The description of the processes is as follows:

(1) Setup: let $e: G_1 \times G_1 \longrightarrow G_2$ be a bilinear map, where G_1 and G_2 are two cyclic groups of the same prime order q and g is a generator of G_1. Select a random $s \in \mathbb{Z}_q^*$ as the master secret key msk and two hash functions $\mathscr{H}_1: \{0,1\}^* \longrightarrow G_1$ and $\mathscr{H}_2: \{0,1\}^* \longrightarrow G_1$. Output the master public parameters mpp = $(\mathscr{H}_1, \mathscr{H}_2, g, g^s, q)$ while keeping the master secret key msk = s private.

(2) KeyGen(msk, id): when PKG receives a key generation request from a legitimate user with identity id, this algorithm takes as input mpp (this parameter is included implicitly here and in following algorithms for simplicity), msk, and identity id and generates the corresponding private key $\mathrm{sk}_{\mathrm{id}} = \mathscr{H}_1(\mathrm{id})^s$.

(3) Encrypt(id, m): when a user wants to encrypt data $m \in G_2$ with identity id, the encryption algorithm picks $r \in_R \mathbb{Z}_q^*$ and computes $C_1 = g^r$ and $C_2 = m \cdot e(g^s, \mathscr{H}_1(\mathrm{id})^r)$ to generate the ciphertext $c_{\mathrm{id}} = (C_1, C_2)$, which will be sent to a cloud server for storage. This resulting ciphertext is also called original ciphertext (level-1 ciphertext).

(4) $\mathrm{RKGen}_1(sk_{\mathrm{id}_1}, \mathrm{id}_1, \mathrm{id}_2, C_1)$: providing as input user's private key $\mathrm{sk}_{\mathrm{id}_1}$, identities $(\mathrm{id}_1, \mathrm{id}_2)$, and an element of ciphertext C_1, this algorithm generates a reencryption key (a.k.a. access token) $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2}$, which will be sent to proxy for storage and converting the ciphertext upon receiving the request from the delegatee. The calculation processing is as follows:

① $K_{\mathrm{id}_1,\mathrm{id}_2} = e(sk_{\mathrm{id}_1}, \mathscr{H}_1(\mathrm{id}_2))$.

② $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2} = e(C_1, \mathscr{H}_2(K_{\mathrm{id}_1,\mathrm{id}_2} \| \mathrm{id}_1 \| \mathrm{id}_2) \cdot \mathrm{sk}_{\mathrm{id}_1})$.

(5) $\mathrm{RKGen}_2(\mathrm{sk}_{\mathrm{id}_1}, \mathrm{id}_1', C_1)$: inputting user's private key $\mathrm{sk}_{\mathrm{id}_1}$, a new identity id_1' and part of the ciphertext C_1 choose a random $r' \in \mathbb{Z}_q^*$, the algorithm computes $R_1 = g^{r'}$ and $R_2 = (e(g^s, \mathscr{H}_1(\mathrm{id}_1')^{r'})/e(C_1, \mathrm{sk}_{\mathrm{id}_1}))$ and generates an update key $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_1'} = (R_1, R_2)$, which is also a reencryption key.

(6) $\mathrm{REncrypt}_1(\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2}, c_{\mathrm{id}_1})$: providing reencryption key $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2}$ and ciphertext $c_{\mathrm{id}_1} = (C_{1,1}, C_{1,2})$ under identity id_1 as input, the algorithm computes $C_{2,1} = C_{1,1}$ and $C_{2,2} = (C_{1,2}/\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2})$ and outputs a delegation ciphertext $c_{\mathrm{id}_2} = (C_{2,1}, C_{2,2})$, which can be decrypted by delegatee with $\mathrm{sk}_{\mathrm{id}_2}$. This ciphertext will be sent to the requester (delegatee) but not stored on the cloud server.

(7) $\mathrm{REncrypt}_2(\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_1'}, c_{\mathrm{id}_1})$: with update key $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_1'} = (R_1, R_2)$ and ciphertext $c_{\mathrm{id}_1} = (C_{1,1}, C_{1,2})$ as input, the algorithm computes $C_{1,1}' = R_1$ and $C_{1,2}' = C_{1,2} \cdot R_2$ and outputs a new ciphertext $c_{\mathrm{id}_1'} = (C_{1,1}', C_{1,2}')$ which will replace the old ciphertext c_{id_1} to store on the cloud server.

(8) $\mathrm{Decrypt}_1(\mathrm{sk}_{\mathrm{id}}, c_{\mathrm{id}})$: after downloading the ciphertext $c_{\mathrm{id}} = (C_{1,1}, C_{1,2})$, original ciphertext or updated ciphertext, from the cloud server, the user (data owner and delegator) runs the algorithm to compute $m = (C_{1,2}/e(C_{1,1}, \mathrm{sk}_{\mathrm{id}}))$ with the private key (usually the latest private key corresponding to the newest identity) to output the plaintext or an error flag $\perp$.

(9) $\mathrm{Decrypt}_2(\mathrm{sk}_{\mathrm{id}_2}, c_{\mathrm{id}_2})$: the requester (delegatee) decrypts the delegation ciphertext $c_{\mathrm{id}_2} = (C_{2,1}, C_{2,2})$ with the private key $\mathrm{sk}_{\mathrm{id}_2} = \mathscr{H}_1(\mathrm{id}_2)^s$ to output plaintext or an error flag $\perp$. The process is as follows:

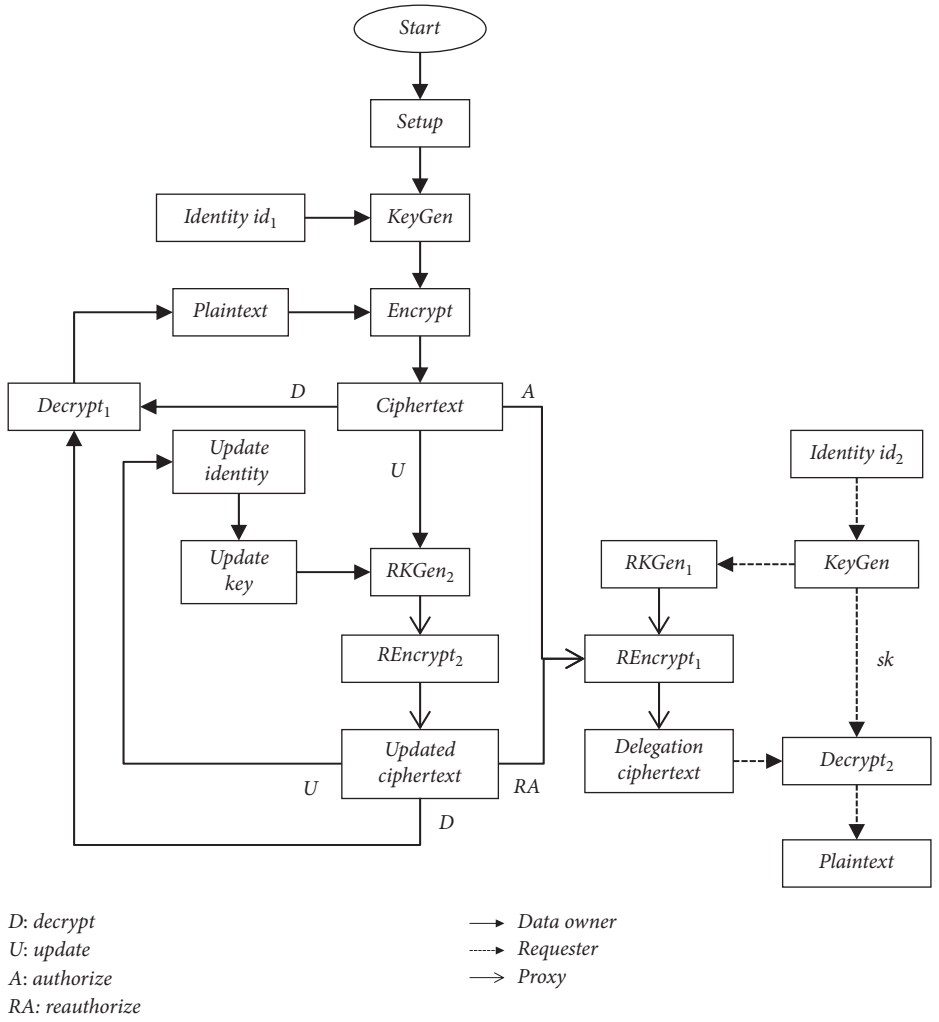

Figure 3: Process flow with additional improvement algorithms.

① $K_{\mathrm{id}_2,\mathrm{id}_1} = e(\mathrm{sk}_{\mathrm{id}_2}, \mathscr{H}_1(\mathrm{id}_1))$.
② $m = C_{2,2} \cdot e(C_{2,1}, \mathscr{H}_2(K_{\mathrm{id}_2,\mathrm{id}_1} \| \mathrm{id}_1 \| \mathrm{id}_2))$.

4.3. Correctness of the Proposed Scheme

(1) The correctness of decrypting the original ciphertext $c_{\mathrm{id}} = (C_{1,1}, C_{1,2})$ with the original private key $\mathrm{sk}_{\mathrm{id}_1} = \mathscr{H}_1(\mathrm{id}_1)^s$ is verified as follows:

$$\frac{C_{1,2}}{e\left(C_{1,1}, \mathrm{sk}_{\mathrm{id}_1}\right)} = m \cdot \left(\frac{e\left(g^s, \mathscr{H}_1(\mathrm{id}_1)^r\right)}{e\left(g^r, \mathscr{H}_1(\mathrm{id}_1)^s\right)}\right) = m. \tag{2}$$

(2) The correctness of processing updated ciphertext $c_{\mathrm{id}_1'} = (C_{1,1}', C_{1,2}') = (R_1, C_{1,2} \cdot R_2)$ with the new private key $\mathrm{sk}_{\mathrm{id}_1'} = \mathscr{H}_1(\mathrm{id}_1')^s$ corresponding to the new identity id_1' is verified as follows:

$$\begin{aligned} \frac{C_{1,2}'}{e\left(C_{1,1}', \mathrm{sk}_{\mathrm{id}_1'}\right)} &= C_{1,2} \cdot \frac{R_2}{e\left(R_1, \mathrm{sk}_{\mathrm{id}_1'}\right)} = \frac{m \cdot e\left(g^s, \mathscr{H}_1(\mathrm{id}_1)^r\right) \cdot \left(e\left(g^s, \mathscr{H}_1(\mathrm{id}_1')^{r'}\right) / e\left(C_1, \mathrm{sk}_{\mathrm{id}_1}\right)\right)}{e\left(g^{r'}, \mathscr{H}_1(\mathrm{id}_1')^s\right)} \\ &= m \cdot e\left(g^s, \mathscr{H}_1(\mathrm{id}_1)^r\right) \cdot \left(\frac{e\left(g^s, \mathscr{H}_1(\mathrm{id}_1')^{r'}\right)}{\left(e\left(g^r, \mathrm{sk}_{\mathrm{id}_1}\right) \cdot e\left(g^{r'}, \mathscr{H}_1(\mathrm{id}_1')^s\right)\right)}\right) = m. \end{aligned} \tag{3}$$

(3) The correctness of processing delegation ciphertext $c_{\mathrm{id}_2} = (C_{1,1}, C_{1,2}/\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2}) = (C_{2,1}, C_{2,2})$ with the delegatee's private key $\mathrm{sk}_{\mathrm{id}_2} = \mathscr{H}_1(\mathrm{id}_2)^s$ is verified as follows:

① $C_{2,2} = (C_{1,2}/(\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2})) = m \cdot (e(g^s, \mathscr{H}_1(\mathrm{id}_1)^r)/e(C_1, \mathscr{H}_2(K_{\mathrm{id}_1,\mathrm{id}_2} \| \mathrm{id}_1 \| \mathrm{id}_2) \cdot \mathrm{sk}_{\mathrm{id}_1})) = m \cdot (e(g^s, \mathscr{H}_1(\mathrm{id}_1)^r)/(e(g^r, \mathscr{H}_2(K_{\mathrm{id}_1,\mathrm{id}_2} \| \mathrm{id}_1 \| \mathrm{id}_2)) \cdot e(g^r, \mathscr{H}_1(\mathrm{id}_1)^s))) = (m/e(g^r, \mathscr{H}_2(K_{\mathrm{id}_1,\mathrm{id}_2} | \mathrm{id}_1 | \mathrm{id}_2)))$

② $K_{\mathrm{id}_2,\mathrm{id}_1} = e(\mathrm{sk}_{\mathrm{id}_2}, \mathscr{H}_1(\mathrm{id}_1)) = e(\mathscr{H}_1(\mathrm{id}_2)^s, \mathscr{H}_1(\mathrm{id}_1)) = e(\mathscr{H}_1(\mathrm{id}_1)^s, \mathscr{H}_1(\mathrm{id}_2)) = e(\mathrm{sk}_{\mathrm{id}_1}, \mathscr{H}_1(\mathrm{id}_2)) = K_{\mathrm{id}_1,\mathrm{id}_2}$

③ $C_{2,2} \cdot e(C_{2,1}, \mathscr{H}_2(K_{\mathrm{id}_2,\mathrm{id}_1} \| \mathrm{id}_1 \| \mathrm{id}_2)) = (m/e(g^r, \mathscr{H}_2(K_{\mathrm{id}_1,\mathrm{id}_2} \| \mathrm{id}_1 \| \mathrm{id}_2))) \cdot e(g^r, \mathscr{H}_2(K_{\mathrm{id}_2,\mathrm{id}_1} \| \mathrm{id}_1 \| \mathrm{id}_2))) = m$

4.4. Security Analysis of the Proposed Scheme. In this study, two reencryption key generation algorithms are employed, the reencryption key generation algorithm for ciphertext update is a new one, and the other for the delegation is an improved one. In an event where the delegatee colludes with the proxy, it can obtain the value $e(g^r, \mathrm{sk}_{\mathrm{id}_1})$ rather than $\mathrm{sk}_{\mathrm{id}_1}$ which means achieving collusion safety. The ciphertext obtained by the former one is in the same format as the ciphertext is generated directly with the new identity id_1'. So the security analysis is the same as the scheme [18].

In addition, we analyze the possible security problems caused by the collaboration of two reencryption key generation algorithms. The delegatee and the proxy conspire to compute a value $e(g^r, \mathrm{sk}_{\mathrm{id}_1})$ from the delegation key. If an attacker constructs another pair such as $(g^{r'}, e(g^s, \mathscr{H}_1(\mathrm{id}_1')^{r'}))$, a new update key $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_1'} = (g^{r'}, e(g^s, \mathscr{H}_1(\mathrm{id}_1')^{r'})/e(C_1, \mathrm{sk}_{\mathrm{id}_1}))$ can be calculated. If the proxy updates the ciphertext with this forged key, a spoofing attack will happen. However, we have assumed that the user can securely send the reencryption key to the proxy and the user needs to be authenticated, and the proxy is semitrusted, so the unauthenticated attacker cannot update the ciphertext even if he has built a new update key.

Next, we present the formal security proofs for the proposed scheme in the random oracle model as follows.

Theorem 1. *Suppose the DBDH assumption holds, our IBPRE-CE scheme is IND-PrID-CPA secure in the random oracle model.*

Proof. Let A be a PPT algorithm that has a nonnegligible advantage ϵ in attacking the proposed scheme. We use A to construct a second algorithm $\mathscr{B}$, which has a nonnegligible advantage at solving the DBDH problem in G_1 and G_2. Algorithm $\mathscr{B}$ accepts as input a properly distributed tuple $(G_1 = g, g^a, g^b, g^c, T) \in G_1^4 \times G_2$ and outputs 1 if $T = e(g, g)^{abc}$. A interacts with algorithm $\mathscr{B}$ as follows.

First, algorithm $\mathscr{B}$ treats the hash functions $\mathscr{H}_1$ and $\mathscr{H}_2$ as random oracles and performs the following simulation:

(1) Simulation of $\mathscr{H}_1$: $\{0,1\}^* \longrightarrow G_1$. When a query for identity id is received, simulator searches the hash table L_1 which includes some tuples like $(\mathrm{id}, h, z, \alpha)$. If the id exists in the L_1, return h as the query result. Otherwise, flip a weighted coin that the probability of heads is γ (represented as $\Pr[\alpha = 1] = \gamma$) to get a random $\alpha \in \{0,1\}$. Select $z \in Z_q^*$ randomly and compute h. If $\alpha = 1$, set $h = g^z$; otherwise, set $h = (g^c)^z$. No matter $\alpha = 1$ or 0, h is correctly distributed.

(2) Simulation of $\mathscr{H}_2$: $\{0,1\}^* \longrightarrow G_1$. When a query for string X like $K \| \mathrm{id}_1 \| \mathrm{id}_2$ where $K = e(\mathrm{sk}_{\mathrm{id}_1}, \mathscr{H}_1(\mathrm{id}_2))$ is received, simulator simply returns $x \in G_1$ and records tuple (X, x) into table L_2.

Then, the interaction simulation proceeds as follows:

(1) Select. A selects i from set $\{0,1\}$.

(2) Setup. $\mathscr{B}$ generates the system's master public parameters $\mathrm{mpp} = (\mathscr{H}_1, \mathscr{H}_2, g, g^a, q)$ and sends it to A.

(3) Phase 1. A adaptively issues a series of queries.

(1) Private key query. When A sends an identity id to $\mathscr{B}$ to query its corresponding private key, $\mathscr{B}$ evaluates $\mathscr{H}_1(\mathrm{id})$ as above to get $(\mathrm{id}, h, z, \alpha)$ and returns $\mathrm{sk}_{\mathrm{id}} = (g^a)^z$ to A.

(2) Update key query. When A sends an update key query from identity id to id', $\mathscr{B}$ selects $r, r' \in Z_q^*$ randomly and evaluates $\mathscr{H}_1(\mathrm{id})$ and $\mathscr{H}_1(\mathrm{id}')$ to obtain (z, α) and (z', α').

If $\alpha = 0$, $\mathscr{B}$ computes $\mathrm{rk}_{\mathrm{id} \longrightarrow \mathrm{id}'} = (e(g^a, \mathscr{H}_1(\mathrm{id}')^{r'})/T^z)$ and returns it to A. Note that this update key is incorrectly formed.

If $\alpha = 1$, $\mathscr{B}$ computes $\mathrm{rk}_{\mathrm{id} \longrightarrow \mathrm{id}'} = (e(g^a, \mathscr{H}_1(\mathrm{id}')^{r'})/e(g^r, g^a)^z)$ and returns it to A.

(3) Delegation key query. When A sends a delegation key query from identity id_1 to identity id_2, $\mathscr{B}$ selects $r1 \in Z_q^*$ and evaluates $\mathscr{H}_1(\mathrm{id}_1)$ and $\mathscr{H}_1(\mathrm{id}_2)$ to obtain $(z1, \alpha 1)$ and $(z2, \alpha 2)$, computes $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2} = e(g^{r1}, (g^a)^{z1} \cdot \mathscr{H}_2((g^a)^{z1}, \mathscr{H}_1(\mathrm{id}_2)) \| \mathrm{id}_1 \| \mathrm{id}_2))$, and returns it to A. Note that when $\alpha = 1$, this delegation key is correctly formed; when $\alpha = 0$, this delegation key is incorrectly formed.

(4) Challenge phase. At the end of phase 1, A submits challenge identity id^* and two same length message $m_0, m_1 \in G_2$ to $\mathscr{B}$ who will evaluate $\mathscr{H}_1(\mathrm{id}^*)$ and look for tuple $(\mathrm{id}^*, h, z, \alpha)$ from table L_1 and return challenge ciphertext $c^* = (g^b, m_i \cdot T^z)$ to A. Note that there are following restrictions on id^* to avoid trivial decryption:

(1) The private key corresponding to id^* has not been queried.

(2) If a series of update key queries which originate from id^* to id_1, next from id_1 to id_2,..., from

id_{n-1} to id_n have been issued, the private key of any id in set $\mathrm{ID} = \{\mathrm{id}_1, \mathrm{id}_2, \ldots \mathrm{id}_n\}$ has not been queried.

(3) If the delegation key query from id^* to id_j has been issued, the private key of id_j has not been queried.

(4) If a series of update key queries which originate from id^* to id_1, next from id_1 to id_2,..., from id_{n-1} to id_n have been issued and delegation key from any id in set $\mathrm{ID} = \{\mathrm{id}_1, \mathrm{id}_2, \ldots \mathrm{id}_n\}$ to id_j has been queried, the private key of id_j has not been queried.

(5) Phase 2. A issues query as in phase 1 except restrictions as above.

(6) Guess. A outputs its guess $i\prime \in \{0, 1\}$. If any of the following conditions are false, $\mathscr{B}$ aborts the simulation. Otherwise, if $i' = i$, $\mathscr{B}$ outputs 1 or 0 otherwise. □

4.4.1. Conditions for Abort

(1) When the private key query for id^* is issued, α is chosen to be 1.

(2) When the private key queries for other id except id^* are issued, α is chosen to be 0.

(3) When the update key queries from id_j to id_j ($\mathrm{rk}_{\mathrm{id}_i \longrightarrow \mathrm{id}_j}$) are issued, if $\mathrm{id}_i \longrightarrow \mathrm{id}_j$ lies along a path leading from id^*, the value α for id_j is chosen to be 1. Or $\mathrm{id}_i \longrightarrow \mathrm{id}_j$ does not lie along a path leading from id^*, the value α for id_i is chosen to be 0.

(4) When the delegation key queries from id_i to id_j ($\mathrm{rk}_{\mathrm{id}_i \longrightarrow \mathrm{id}_j}$) are issued, if $\mathrm{id}_i \longrightarrow \mathrm{id}_j$ lies along a path leading from id^*, the value α for id_j is chosen to be 1. Or $\mathrm{id}_i \longrightarrow \mathrm{id}_j$ does not lie along a path leading from id^*, the value α for id_i is chosen to be 0.

Claim. If $\mathscr{B}$ does not abort during the game, then A's view is identical to the real attack, except reencryption keys of the form $\mathrm{rk}_{\mathrm{id}^* \longrightarrow \ldots}$. The discussion of this case can refer to [18]. If the input to $\mathscr{B}$ is a DBDH tuple, then the challenge ciphertext c^* is correct encryption of m_i under id^* and hence (subject to the definition of A and the argument above) $|\Pr[i' = i] - (1/2)| \geq \varepsilon$. When the input to $\mathscr{B}$ is random, c^* represents the encryption of a random element. Since A is unable to distinguish between the simulation and a real attack, it must hold that $\Pr[i\prime = i] \geq \epsilon + (1/2)$ for a nonnegligible ϵ. Hence, $\mathscr{B}$ succeeds with nonnegligible advantage.

5. Application Scenario Implementation of the Proposed Scheme

In this section, we introduce a secure cloud data management scenario and apply the improved algorithms mentioned above to implement three secure data management functions:

(1) Update ciphertext and update the key to achieve access permission revocation

(2) Reauthorization after the ciphertext is updated

(3) Conditional reservation authorization

5.1. Secure Cloud Data Management Scenario. In a cloud data sharing environment, the user encrypts data with the identity and uploads it to the cloud server. User data may be encrypted and uploaded to the cloud server at different times. Usually, the same random number r is selected to encrypt several files at the same time, while different random number r is chosen to encrypt files at different times. In order to share the encrypted data to others, the user needs to calculate the corresponding reencryption key for the ciphertext. Table 1 shows the relations of plaintext, ciphertext, random number, and time (can be utilized to compute the lifetime of the key) during encryption and reencryption keys list (REK list) for delegation. In Table 1, the random value and the time can be the same which means that these data were encrypted at the same time, where $C_i = (g^{r_i}, m_i \cdot e(g^s, \mathscr{H}_1(\mathrm{id}_1)^{r_i}))$, and $\mathrm{REK}_i \subseteq \left\{\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2}, \mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_3}, \ldots\right\}$ is a reencryption key list that contains some reencryption keys corresponding to the delegation. If this set is empty, this means the data are not allowed to be shared with others.

For the sake of analysis, let us assume this scenario: user id_1 has three types of files (m_1, m_2, and m_3), which are encrypted with the identity and stored on the cloud server. The first type of file m_1 is some study notes shared with a requester (a common learner with identity id_2). The second type of file m_2 is personal photos that involve some private information and should not be allowed to be accessed by others. The content of the third file m_3 is several relevant information that a user (assumed as a prosumer) shares with the utility organization or other prosumers in a microgrid.

5.2. Delegation Revocation Management. From the construction of the reencryption key, we can find that the delegation reencryption key has nothing to do with the second part of the ciphertext but only with the random number chosen for the ciphertext, the delegator, and the delegatee. Assume that the files m_1, m_2, and m_3 are encrypted with the same random number under the identity id_1. In this situation, as long as one of the user's files is authorized to the requester, the proxy can use the delegation reencryption key to delegate all other data to the requester. So it is not security. To keep the user's other files safe, the user needs to revoke the delegation of the requester. Here, it can revoke access rights by updating the ciphertext. The processing of revoking the requestor id_2's access permission to files m_2 and m_3 is shown as follows:

Step 1: generate a new identity. The user combines well-known information with an attribute (manufacturer/smart meter id: $\mathrm{id}_{\mathrm{sm}}$) of microgrid devices (i.e., smart meter) as a new identity$\mathrm{id}_1' = \mathrm{id}_1 \| \mathrm{id}_{\mathrm{sm}}$ for the third type of file m_3. In the same manner, the user will also combine another attribute as an encryption identity for

Table 1: Plaintext, ciphertext, random, encryption time, and reencryption key list information.

Data	Random	Ciphertext	Time	REK list
m_1	r_1	C_1	T_1	REK_1
m_2	r_2	C_2	T_2	REK_2
...	...	...	...	...
m_n	r_n	C_n	T_n	REK_n

the second type of file m_2. Both operations are the same, and file m_3 is considered for the following analysis.

Step 2: generate a new private key. The user sends the new identity to the PKG to generate a new private key $\mathrm{sk}_{\mathrm{id}_1'} = \mathscr{H}_1(\mathrm{id}_1')^s$.

Step 3: compute a reencryption key for updating ciphertext. The user chooses a random $r' \in \mathbb{Z}_q^*$ and runs algorithm $\mathrm{RKGen}_2(\mathrm{sk}_{\mathrm{id}_1}, \mathrm{id}_1', C_{1,1})$ to generate $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_1'} = (R_1, R_2) = (g^{r'}, (e(g^s, \mathscr{H}_1(\mathrm{id}_1')^{r'})/e(C_{1,1}, \mathrm{sk}_{\mathrm{id}_1})))$ and then send it to the cloud server.

Step 4: update the ciphertext. After receiving the update reencryption key $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_1'} = (R_1, R_2)$, the proxy uses it to convert the ciphertext c_{id_1} to a new ciphertext $c_{\mathrm{id}_1'} = (R_1, C_{1,2} \cdot R_2) = (C_{1,1}', C_{1,2}')$, which will replace the old ciphertext c_{id_1} to store in the cloud server. Note that the proxy does not save this reencryption key.

Step 5: the user (data owner) decrypts the updated ciphertext with the new private key $\mathrm{sk}_{\mathrm{id}_1'}$. The computation steps are as follows:

(1) $C_{1,1}' = R_1 = g^{r'}$.

(2) $C_{1,2}' = C_{1,2} \cdot R_2 = m_3 \cdot (e(g^s, \mathscr{H}_1(\mathrm{id}_1)^r) \cdot e(g^s, \mathscr{H}_1(\mathrm{id}_1')^{r'})/e(g^r, \mathrm{sk}_{\mathrm{id}_1})) = m_3 \cdot e(g^s, \mathscr{H}_1(\mathrm{id}_1')^{r'})$.

(3) $(C_{1,2}'/e(C_{1,1}', sk_{id_1'})) = m_3 \cdot (e(g^s, \mathscr{H}_1(id_1')^{r'})/e(C_{1,1}', sk_{id_1'})) = m_3 \cdot (e(g^s, \mathscr{H}_1(id_1')^{r'})/e(g^{r'}, \mathscr{H}_1(id_1')^s)) = m_3$.

Obviously, when the requester requests the data from the cloud server again, the new ciphertext transformed by proxy from the updated ciphertext with old reencryption key $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2}$ cannot be decrypted with the requester's private key $\mathrm{sk}_{\mathrm{id}_2} = \mathscr{H}_1(\mathrm{id}_2)^s$ because

$$C_{2,2}' = m_3 \cdot \frac{\left(g^s, \mathscr{H}_1(\mathrm{id}_1')^{r'}\right)}{\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_2}} = \frac{m_3 \cdot e\left(g^s, \mathscr{H}_1(\mathrm{id}_1')^{r'}\right)}{e\left(g^r, \mathscr{H}_2\left(K_{\mathrm{id}_1,\mathrm{id}_2} \| \mathrm{id}_1 \| \mathrm{id}_2\right) \cdot \mathscr{H}_1(\mathrm{id}_1)^s\right)}. \tag{4}$$

So, when the user revokes the delegation for the requester id_2 by updating the ciphertext, the proxy will delete the corresponding old reencryption key. And when the requester id_2 requests the data, the user will refuse to generate a delegation key.

5.3. Reauthorization Management. After the above ciphertext update operation, given that these operations were also applied to m_2. It can now only be accessible to the data owner. However, file m_3 is required to be shared with other entities (other prosumers or the utility organization and assume identity is id_3) in the microgrid. For example, the user needs to reauthorize to energy consumers id_3, and this process is shown as follows:

Step 1: generate the reauthorization key. The user (delegator) runs the algorithm $\mathrm{RKGen}_1(\mathrm{sk}_{\mathrm{id}_1'}, \mathrm{id}_1', \mathrm{id}_3, C_{1,1}')$ to compute a new reencryption key $\mathrm{rk}_{\mathrm{id}_1' \longrightarrow \mathrm{id}_3}$. The calculation process is shown as follows:

(1) $K_{\mathrm{id}_1',\mathrm{id}_3} = e(sk_{\mathrm{id}_1'}, \mathscr{H}_1(\mathrm{id}_3))$.

(2) $\mathrm{rk}_{\mathrm{id}_1' \longrightarrow \mathrm{id}_3} = e(C_{1,1}', \mathscr{H}_2(K_{\mathrm{id}_1',\mathrm{id}_3} \| \mathrm{id}_1' \| \mathrm{id}_3) \cdot \mathrm{sk}_{\mathrm{id}_1'})$.

Then, this reencryption key is sent to the proxy to update the reencryption key list.

Step 2: generate reauthorization ciphertext. When the requester (delegatee) wants to access the data, the proxy runs the algorithm $\mathrm{REncrypt}_1(\mathrm{rk}_{\mathrm{id}_1' \longrightarrow \mathrm{id}_3}, c_{\mathrm{id}_1'})$ to convert the updated ciphertext $c_{\mathrm{id}_1'}$ to a new ciphertext $c_{\mathrm{id}_3} = (C_{1,1}', C_{1,2}'/\mathrm{rk}_{\mathrm{id}_1' \longrightarrow \mathrm{id}_3}) = (C_{3,1}, C_{3,2})$. Then, the ciphertext c_{id_3} is sent to the requester (delegatee).

Step 3: decrypt the reauthorization ciphertext. After receiving the ciphertext from the proxy, the requester decrypts the ciphertext c_{id_3} with the private key $\mathrm{sk}_{\mathrm{id}_3} = \mathscr{H}_1(\mathrm{id}_3)^s$ to get the plaintext. The decryption process is shown as follows:

(1) $K_{\mathrm{id}_3,\mathrm{id}_1'} = e(\mathrm{sk}_{\mathrm{id}_3}, \mathscr{H}_1(\mathrm{id}_1'))$.

(2) $m = C_{3,2} \cdot e(C_{3,1}, \mathscr{H}_2(K_{\mathrm{id}_3,\mathrm{id}_1'} \| \mathrm{id}_1' \| \mathrm{id}_3))$.

5.4. Conditional Reservation Delegation Management. This function is added to handle the time period access control in scenarios where a user wants to grant access to a requester at a future time. For example, one user with a solar energy collection system wants to sell the excess energy to others in the microgrid. Similarly, other prosumers also wish to sell their excess energy. These suppliers will share their energy information to an energy consumer, who cannot access the data containing the energy price in advance. Otherwise, it may lead to malicious bidding or collusion. This situation may require conditional reservation delegation. In this work, this process is shown as follows:

Step 1: encrypt the bidding file and upload it to the cloud. The user (seller, identity is id_1) generates an energy information file and encrypts it with the identity. The ciphertext $c_{\mathrm{id}_1} = (C_{1,1}, C_{1,2})$ is uploaded to the cloud server and can be shared with other users.

Step 2: generate conditional reservation delegation key:

(1) The user assigns the combination of the requester's identity, another information attribute (i.e., smart meter id), and a date constraint (i.e., trade schedule) as the identity $\mathrm{id}_3 = \mathrm{identity}||\mathrm{id}_{\mathrm{sm}}||\mathrm{duration}$.

(2) The user generates a delegation reencryption key $\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_3} = e(C_{1,1}, \mathscr{H}_2(K_{\mathrm{id}_1,\mathrm{id}_3} \| \mathrm{id}_1 \| \mathrm{id}_3) \cdot \mathrm{sk}_{\mathrm{id}_1})$ and uploads to the proxy and indicates the composition of the delegatee's identity.

Step 3: generate a private key. At the appointed time, the requester applies for the private key $\mathrm{sk}_{\mathrm{id}_3} = \mathscr{H}_1(\mathrm{id}_3)^s$ corresponding to the identity $\mathrm{id}_3 = \mathrm{identity}||\mathrm{id}_{\mathrm{sm}}||\mathrm{duration}$ from PKG.

Step 4: generate the delegation ciphertext. The proxy runs the algorithm $\mathrm{REncrypt}_1(\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_3}, c_{\mathrm{id}_1})$ to transform the ciphertext c_{id_1} to the delegation ciphertext $c_{\mathrm{id}_3} = (C_{1,1}, C_{1,2}/\mathrm{rk}_{\mathrm{id}_1 \longrightarrow \mathrm{id}_3}) = (C_{3,1}, C_{3,2})$. And then the ciphertext c_{id_3} is sent to the requester.

Step 5: decrypt the delegation ciphertext. The requester decrypts the ciphertext c_{id_3} with the private key $\mathrm{sk}_{\mathrm{id}_3} = \mathscr{H}_1(\mathrm{id}_3)^s$ to get the plaintext. The process is shown as follows:

(1) $K_{\mathrm{id}_3,\mathrm{id}_1} = e(sk_{\mathrm{id}_3}, \mathscr{H}_1(\mathrm{id}_1))$.

(2) $m = C_{3,2} \cdot e(C_{3,1}, \mathscr{H}_2(K_{\mathrm{id}_3,\mathrm{id}_1}\|\mathrm{id}_1\|\mathrm{id}_3))$.

6. Comparison and Analysis

In this section, we compare the proposed scheme with three PRE schemes in terms of characteristics, functionality, computation cost, and efficiency. In literatures, several works that implement secure data sharing with PRE can be divided into two categories. One is based on PKI, and the other is identity-based. Identity-based architecture can be further divided into two categories: BF (Boneh–Franklin) [22] architecture in the random oracle model and waters [23] architecture in the standard model. For convenience and simplicity of comparison and analysis, GA (Green–Ateniese) [18] is chosen as the representative of BF architecture, LLWS (Liang–Liu–Wong–Susilo) [10] for waters architecture, and SHB (Shafagh–Hithnawi–Burkhalter) [6] for PKI architecture to compare with the proposed scheme. The GA_1 and GA_2 correspond to basic IBP1 in the GA scheme, and GA_2 is an optimization of GA_1 in terms of the absence of ciphertext expansion during reencryption but is not multiusable and nontransferable.

Table 2 shows some of the characteristics and achieved security notion of the schemes for two functions: update and delegation. Since the reencrypted ciphertext in scheme GA_2 is non-reencryptable [25], this method cannot be used to update the ciphertext and not included in the comparison set supporting updates. Similarly, the delegation function is not discussed in the scheme LLWS, so it is not included in the function comparison set of delegation. For the update function, the decryptor is still the data owner himself regardless of the original ciphertext or the updated ciphertext. There is no change in the access right. It is of no need to consider transferability, which is not compared in the table and filled with "-." The TM and EM represent true multiuse and expansive multiuse, respectively [13]. Note that although the reencryption key generation algorithm needs an element of the ciphertext, the proposed scheme is still noninteractive because the ciphertext component can be downloaded from the cloud server without the involvement of proxy.

The characteristic comparison in Table 2 is important for secure data sharing application, and the definitions are described in Section 2. In most cases, the public-key certificate has an expiration date. In addition, a user's key (both public and private key) needs to be changed for security consideration. These changes will cause the ciphertext to be updated. Thus, in the process of user data management, ciphertext update is a very common operation, and the proxy reencryption algorithm must support multiple encryptions (multiuse). Furthermore, reencryption that leads to the extension of ciphertext length is not allowed. Thus, the scheme should achieve ciphertext optimization. Moreover, in cloud servers, the delegation ciphertext is not permitted to be shared again, which is a typical security policy for data sharing. Thus, this requires that the scheme should be nontransferable, but the schemes GA_1 and GA_2 are transferable. In the PRE system, the CPA security is a minimum requirement, and the private key of the delegator should be safe even if the delegatee colludes with the proxy; this means that the scheme should be collusion safe as achieved by the proposed one. If the user data need to be shared with multiple requesters in the PKI-based system, the user has to maintain a lot of public-key certificates. It will induce a higher overhead to the user. However, since the proposed proposal is identity-based, this situation is avoided.

The essential feature of PRE is delegation. Thus, it is used as the basis of underlying encryption in many secure data sharing systems. However, many proposals in literature only considered the delegation function but did not include the function of revoking delegation. Some systems employ conditional authorization, but very few schemes have taken the conditional reservation delegation into account. Furthermore, fewer schemes in the literature involve the ciphertext/key update and reauthorization, and even if there are, the analysis is not detailed enough. Table 3 compares the functions of some existing representative schemes and the proposed approach. It is observed that the proposed scheme is a comprehensive data sharing solution that supports all the essential features of delegation, delegation revocation, conditional reservation delegation, ciphertext update, and reauthorization required for secure cloud data sharing.

In Table 4, the computational cost of each phase is shown. For the sake of illustration, *Setup, KeyGen, and Encrypt* denote the setup algorithm, key generation algorithm, and encryption algorithm, respectively. $Decrypt_1$ denotes the decryption algorithm used by the data owner to decrypt the ciphertext (original or updated). $Decrypt_2$ denotes the decryption algorithm used by the delegatee to decrypt the reencrypted delegation ciphertext. $RKGen_1$ denotes the delegation key generation algorithm. $RKGen_2$ denotes an update key generation algorithm. $REncrypt_1$ denotes the reencryption algorithm for generating a delegation ciphertext. $REncrypt_2$ denotes the reencryption algorithm for updating ciphertext. t_P denotes the pairing computation time. t_R denotes the time of the random computation. t_M denotes the calculation time of group element multiplication or division operations (the inverse computation is seen as division operation). t_E denotes the computation time of a group element power operation. In these calculations, the selection of random values, multiplication, division, and inverse takes tens of microseconds; exponentiation takes several milliseconds and pairing takes between 10 and 20 milliseconds.

Table 2: Comparison of the representative scheme and the proposed scheme based on essential characteristics.

Characteristics	Update				Delegation			
	SHB	LLWS	GA_1	Proposed	SHB	GA_1	GA_2	Proposed
Interactive	Yes	No	No	No	No	No	No	No
Multiuse	TM	EM	EM	TM	No	EM	No	No
Transitive	Yes	No	No	Yes	No	No	No	No
Transferable	—	—	—	—	No	Yes	Yes	No
Collusion safe	No	Yes	No	Yes	Yes	No	No	Yes
Ciphertext optimal	Yes	No	No	Yes	Yes	No	Yes	Yes
Achieved security	CPA	CPA	CPA	CPA	CPA	CPA	CPA	CPA
Standard model	Yes	Yes	No	No	Yes	No	No	No
Identity-based	No	Yes	Yes	Yes	No	Yes	Yes	Yes

Table 3: Functionality comparison of the different PRE schemes.

Functionalities	SHB	LLWS	GA	Proposed
Delegation	Yes	Yes	Yes	Yes
Delegation revocation	Yes	Yes	No	Yes
Conditional reservation delegation	No	No	Yes	Yes
Ciphertext update	Yes	Yes	No	Yes
Reauthorization	No	Yes	No	Yes
Detail analysis	No	No	No	Yes

Table 4: Computation cost comparison of each phase in the different schemes.

Algorithms	SHB	LLWS	GA_1	GA_2	Proposed
Setup	t_R	$3t_E, (n+7)t_R$	$t_E, 2t_R$	$t_E, 2t_R$	$t_E, 2t_R$
KeyGen	t_E, t_R	$11t_E, 7t_R, (n+11)t_M$	t_E	t_E	t_E
Encrypt	$2t_E, t_R, t_P, t_M$	$4t_E, t_R, t_P, t_M$	$2t_E, t_R, t_P, t_M$	$2t_E, t_R, t_P, t_M$	$2t_E, t_R, t_P, t_M$
$Decrypt_1$	$t_E, t_P, 2t_M$	$3t_P, 3t_M$	t_P, t_M	t_P, t_M	t_P, t_M
$RKGen_1$	t_E, t_M	—	$2t_E, 2t_R, t_P, t_M$	t_P, t_M	$2t_P, t_M$
$REncrypt_1$	t_P	—	t_P, t_M	t_P	t_M
$Decrypt_2$	$t_E, 2t_M$	$4t_P, t_E, 5t_M$	$2t_P, 2t_M$	$2t_P, t_M$	$2t_P, t_M$
$RKGen_2$	t_M	$8t_E, 3t_R, t_P, 5t_M$	—	—	$2t_E, t_R, 2t_P, t_M$
$REncrypt_2$	t_P	$2t_P, t_M$	—	—	t_M

Table 5: Comparison of the computation time (in ms) of each algorithm in different schemes.

Algorithms	SHB	LLWS	GA_1	GA_2	Proposed
Setup	6.474	376.159	12.801	13.113	13.025
KeyGen	6.467	71.027	21.669	22.01	22.146
Encrypt	21.056	33.701	21.019	21.327	21.033
$Decrypt_1$	19.552	39.174	13.121	13.255	13.009
$RKGen_1$	6.472	—	21.103	13.278	26.068
$REncrypt_1$	13.065	—	13.109	13.214	0.02
$Decrypt_2$	1.557	58.607	26.215	26.52	26.076
$RKGen_2$	0.005	59.323	—	—	34.592
$REncrypt_2$	6.453	26.089	—	—	0.011

Table 5 and Figure 4 show the computation time of each phase of the five schemes. For the convenience of calculation, the hash calculation in all algorithms is not considered. The SHB scheme is the ElGamal algorithm based on the elliptic curve. The LLWS scheme is mainly used for ciphertext updating. Thus, the run time of algorithm $RKGen_1$ and $REncrypt_1$ for authorization is not provided. Note that the run time of the algorithm $Decrypt_1$ is for decrypting the original ciphertext, while the run time of the algorithm $Decrypt_2$ is for decrypting the reencrypted ciphertext, which is updated only once. The GA work does not include ciphertext update or is not suitable for updating ciphertext (e.g., ciphertext expands with reencryption), and the time of update key generation and ciphertext update was also not

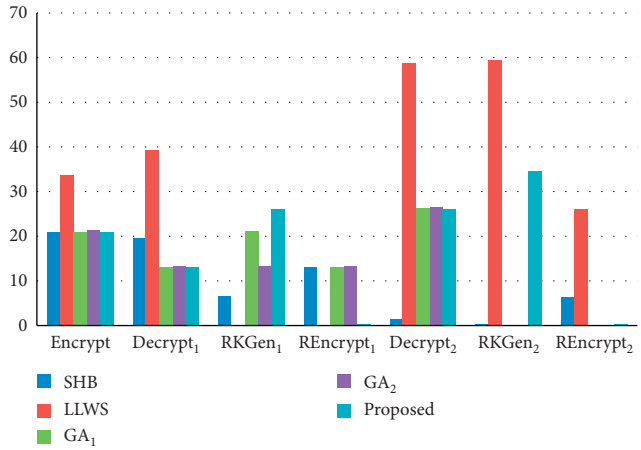

Figure 4: Comparison of the computation time (in ms) of each algorithm in different schemes.

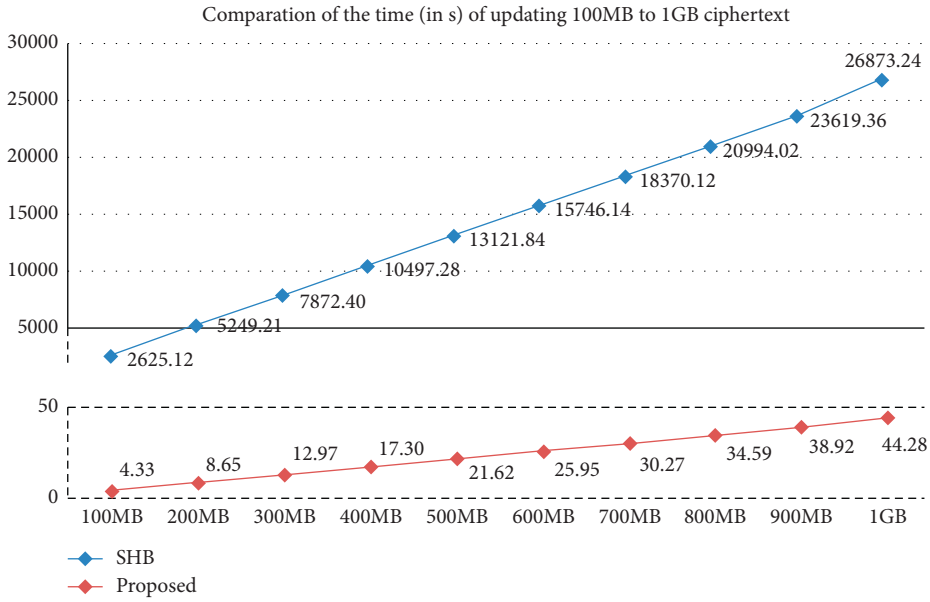

Figure 5: Computation time of updating 100 MB to 1 GB ciphertext in SHB and proposed scheme.

calculated. These performance evaluations are performed on a computer with 3.4 GHz Intel Core i5-3570 processor, 8 GB RAM, and the operating system is Windows 7 Professional with Service Pack 1. All the programs are based on the PBC library [28], where the parameter is the type A curve.

In Table 5, the setup time (the length of the user's identity is 50 bits) of the LLWS scheme is larger than other systems because it is a waters-based architecture and often needs to generate more system parameters. Since the time of hash computation in *KeyGen* algorithm is not included, the computation time of this phase in the LLWS scheme is less than the BF architecture. For the purposes of simplifying comparison, the calculation times of the *Setup* phase and *KeyGen* phase are omitted. And it is shown that the computation time of LLWS is the longest, while SHB is the shortest. The computation time of *Setup*, *KeyGen*, *Encrypt*, and *Decrypt*$_1$ are the same in GA$_1$ and GA$_2$ and the proposed proposal because these approaches are based on BF architecture. And the computation time for *Decrypt*$_2$ of the three schemes is approximately the same. During the *RKGen*$_1$ phase, the proposed scheme needs two pairing computations making the computation time larger than GA$_1$ and GA$_2$. Although the computation time of *RKGen*$_2$ is very long, the impact on the overall system will be modest because the number of keys needed to be calculated is few and the ciphertext update frequency is not too high. Furthermore, the computation time of *REncrypt*$_1$ and *REncrypt*$_2$ in the proposed scheme is very short. Therefore, the proposed algorithm has the quickest processing time when the proxy responds to the request for delegation or update, which is especially important when there are many access requests, or there are many ciphertexts that need to be updated. Since in practice, once the user refreshes the key, the ciphertext needs to be updated synchronously, and the reencryption (update) algorithm in our scheme only needs tens of microseconds, so it is very efficient to update a large number of ciphertext synchronously in practical applications.

Figure 5 shows the time required to update the ciphertext from 100 MB to 1 GB, respectively. The SHB scheme needs 26,873.24 seconds to update 1 GB ciphertext, while the proposed work only needs 44 seconds. This value plus the time of reencryption key generation (34.592 ms) is also much less than the time required by SHB. This means that the proposed scheme has high efficiency and practicability in the scenario that a lot of ciphertext needs to be updated synchronously after the key update.

7. Conclusions

In this paper, an improved IBPRE scheme is proposed to realize secure cloud data sharing. The approach fully considers the ciphertext update while considering the access authorization and supports file security management operations such as access authorization, authorization revocation, ciphertext update, reauthorization, and conditional reservation authorization. Moreover, the scheme included the characteristics of unidirectionality, noninteractivity, collusion safety, multiuse, nontransferability, ciphertext optimization, and key optimization, which are essential and practical for general applications. From the performed simulation and analysis, the two reencryption algorithms in the proposed proposal require only one multiplication of group elements, in which the processing time of simultaneously responding to multiple delegation requests or updating numerous ciphertexts is shorter compared with other schemes. However, as of this writing, the design only achieves CPA security in the random oracle model but aims to achieve CCA security in the future iteration.

Data Availability

The simulation codes based on PBC used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

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

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China (Project no. 61662039), and it was supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (Project no. 2018-0-00508) and in part by KETEP, Korean Government, Ministry of Trade, Industry, and Energy (MOTIE) (Project no. 20194010201800).

References

[1] X. Zhang, H. Wang, and C. Xu, "Identity-based Key-Exposure Resilient Cloud Storage Public Auditing Scheme from Lattices," *Information Sciences*, pp. 223–234, 2019.

[2] C. Ge, Z. Liu, J. Xia, and L. Fang, "Revocable identity-based broadcast proxy re-encryption for data sharing in Clouds," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[3] X. Zhang, Y. Tang, H. Wang, C. Xu, Y. Miao, and H. Cheng, "Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage," *Information Sciences*, vol. 494, pp. 193–207, 2019.

[4] 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.

[5] X. Zhang, C. Xu, H. Wang, Y. Zhang, and S. Wang, "FS-PEKS: lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial internet of things," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[6] H. Shafagh, A. Hithnawi, L. Burkhalter, P. Fischli, and S. Duquennoy, "Secure sharing of partially homomorphic encrypted IoT data," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, ACM, Delft, Netherlands, pp. 1–14, November 2017.

[7] National Institute of Standards and Technology, "Recommendation for Key Management," 2020, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf.

[8] P. Xu, T. Jiao, Q. Wu, W. Wang, and H. Jin, "Conditional identity-based broadcast proxy re-encryption and its application to cloud email," *IEEE Transactions on Computers*, vol. 65, no. 1, pp. 66–79, 2016.

[9] C. Ge, W. Susilo, J. Wang, and L. Fang, "Identity-based conditional proxy re-encryption with fine grain policy," *Computer Standards & Interfaces*, vol. 52, pp. 1–9, 2017.

[10] 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 *Proceedings of the European Symposium On Research In Computer Security*, pp. 257–272, Wroclaw, Poland, September 2014.

[11] C. Wang, J. Fang, and Y. Li, "An improved cloud-based revocable identity-based proxy re-encryption scheme," in *Proceedings of the International Conference on Applications and Techniques in Information Security*, pp. 14–26, Beijing, China, November 2015.

[12] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proceedings of the International Conference on the Theory And Applications of Cryptographic Techniques*, pp. 127–144, Espoo, Finland, May 1998.

[13] D. Nuñez, I. Agudo, and J. Lopez, "Proxy Re-Encryption: analysis of constructions and its application to secure access delegation," *Journal of Network and Computer Applications*, vol. 87, pp. 193–209, 2017.

[14] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.

[15] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ACM, Alexandria, VA, USA, pp. 185–194, October 2007.

[16] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1786–1802, 2011.

[17] C.-K. Chu and W.-G. Tzeng, "Identity-based proxy re-encryption without random oracles," in *Proceedings of the*

International Conference On Information Security, pp. 189–202, Valparaíso, Chile, October 2007.

[18] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, pp. 288–306, Zhuhai, China, June 2007.

[19] X. Liang, Z. Cao, H. Lin, and J. Shao, "Attribute based proxy re-encryption with delegating capabilities," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ACM, Sydney Australia, pp. 276–286, March 2009.

[20] K. Liang, L. Fang, W. Susilo, and D. S. Wong, "A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security," in *Proceedings of the 2013 5th International Conference On Intelligent Networking And Collaborative System*, IEEE, Xi'an, China, pp. 552–559, September 2013.

[21] Y. Aono, X. Boyen, L. T. Phong, and L. Wang, "Key-private proxy re-encryption under LWE,"vol. 8250, pp. 1–18, in *Proceedings of the International Conference on Cryptology in India*, vol. 8250, pp. 1–18, Lecture Notes in Computer Science, Mumbai, India, December 2013.

[22] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proceeedings of the Annual International Cryptology Conference*, vol. 32, no. 3, pp. 586–615, Santa Barbara, California, USA, August 2001.

[23] B. Waters, "Efficient identity-based encryption without random oracles," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 114–127, Aarhus, Denmark, May 2005.

[24] L. Wang, L. Wang, M. Mambo, and E. Okamoto, "New identity-based proxy re-encryption schemes to prevent collusion attacks," in *Proceedings of the International Conference on Pairing-Based Cryptography*, Yamanaka Hot Spring, Kaga, Japan, pp. 327–346, December 2010.

[25] H. Wang, Z. Cao, and L. Wang, "Multi-use and unidirectional identity-based proxy re-encryption schemes," *Information Sciences*, vol. 180, no. 20, pp. 4042–4059, 2010.

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

[27] Y. Zhou, H. Deng, Q. Wu, B. Qin, J. Liu, and Y. Ding, "Identity-based proxy re-encryption version 2: making mobile access easy in cloud," *Future Generation Computer Systems*, vol. 62, pp. 128–139, 2016.

[28] B. Lynn, "PBC: pairing-based cryptography," https://crypto.stanford.edu/pbc/.