Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 1567019, 16 pages
https://doi.org/10.1155/2021/1567019

Research Article

Broadcast Proxy Reencryption Based on Certificateless Public Key Cryptography for Secure Data Sharing

Won-Bin Kim,[1] Su-Hyun Kim,[2] Daehee Seo,[3] and Im-Yeong Lee[1]

[1]*Department of Software Convergence, Soonchunhyang University, Asan 31538, Republic of Korea*
[2]*National IT Industry Promotion Agency, Jincheon 27872, Republic of Korea*
[3]*Faculty of Artificial Intelligence and Data Engineering, Sangmyung University, Seoul 03016, Republic of Korea*

Correspondence should be addressed to Im-Yeong Lee; imylee@sch.ac.kr

Received 30 August 2021; Accepted 9 November 2021; Published 16 December 2021

Academic Editor: Yingjie Wang

Broadcast proxy reencryption (BPRE), which combines broadcast encryption (BE) and proxy reencryption (PRE), is a technology used for the redistribution of data uploaded on the cloud to multiple users. BPRE reencrypts data encrypted by the distributor and then uploads it to the cloud into a ciphertext that at a later stage targets multiple recipients. As a result of this, flexible data sharing is possible for multiple recipients. However, various inefficiencies and vulnerabilities of the BE, such as the recipient anonymity problem and the key escrow problem, also creep into BPRE. Our aim in this study was to address this problem of the existing BPRE technology. The partial key verification problem that appeared in the process of solving the key escrow problem was solved, and the computational efficiency was improved by not using bilinear pairing, which requires a lot of computation time.

1. Introduction

The cloud technology allows users to access a range of services anytime, anywhere. Depending on the requirement of users, the cloud provides a range of services, including software-as-a-service (SaaS), platform-as-a-service (PaaS), and infrastructure-as-a-service (IaaS). Recently, it has become easier to use the cloud owing to the rapid development of communication and computing technology, and consequently, the cloud has been introduced and used in various domains and environments.

In general, the cloud technology is recognized as a remote storage environment or as a software that can be used without the need for installing it on a local device. Microsoft's Office 365 products or Adobe's CC product line can be seen as cloud-based software, and Google Drive and Microsoft's One Drive are representative examples of cloud-based remote storage. These products, running on the Internet, provide functions that local storage fails to provide and can be easily used anytime, anywhere. However, the cloud is not limited to the range of services described above. Its scope has expanded into more diverse and extensive areas and domains in recent times.

Cloud technology requires an Internet connection to work. In other words, the cloud is an online technology. As a result of this, the working environment of the cloud can be shared online at any given time. For example, the cloud in an enterprise environment is not just for each employee to store their own data. It can also be used by employees as an efficient tool to share their work with each other. In light of this, there is an increasingly urgent need for technologies that can store and share data through such a cloud [1, 2].

The cloud essentially is a proxy server, that is, it is a remote server that can be accessed and used via a network. However, the proxy server responds to the request but it is always considered a semitrusted server because it always wants to know its contents. Therefore, for data to be stored safely in the cloud, data encryption is essential. In addition, to further share the data stored in the cloud, the recipient must be able to easily decrypt the encrypted data. Moreover, for the data sender to decrypt the encrypted data, a decryption key is required. However, the two most popular methods for this, symmetric key encryption and asymmetric key encryption, suffer from the key distribution problem.

Therefore, proxy reencryption (PRE) has been proposed to securely share data without exposing the data contents and decryption keys to risks during the data sharing process.

PRE reencrypts data encrypted using the sender's public key in the proxy so that the receiver can decrypt it by using their own private key. As a result, the private keys of the sender and receiver are not exposed to risks during data sharing and the cloud, too, has no access to the contents of the encrypted data. However, because this PRE is based on one-to-one transmission, it is not suitable for environments where the same data are distributed to multiple recipients (for example, environments such as update servers or secure e-mail). In such a scenario, if the existing proxy reencryption is used, reencryption key generation and reencryption must be performed as many times as the number of recipients.

Broadcast proxy reencryption (BPRE) combines broadcast encryption (BE), which enables sending of the same data to multiple recipients by using a single encryption, and PRE. Therefore, BPR can reencrypt encrypted data stored in the cloud and distribute it to multiple recipients, enabling flexible data sharing. However, because BPRE is an encryption method based on BE, it also suffers from some security vulnerabilities that are typical to BE. For example, the lack of receiver anonymity in BE, wherein the identity of a specific receiver in a communication channel gets exposed, leading to serious privacy issues, is also present in BPRE. In addition, the security threat caused by BE's public key generation method also appears in BPRE.

Typically, there is a key escrow problem that appears in ID-based cryptography (IBC) and the certificateless (CL) cryptography can be applied to solve this problem. In addition, the partial key verification problem of the CL cryptography must also be considered. Finally, existing BPRE schemes were designed using a bilinear pairing operation. However, bilinear pairing operation is a time-consuming process, which increases the computing cost in BPRE. Therefore, the goal of this study is to provide a relatively safe environment for data sharing by solving the security threats presented above, as well as to develop a more efficient BPRE technology by leaving out the bilinear pairing operation.

2. Related Works

This section describes related studies and theoretical constructs for understanding the concepts discussed in the present study.

2.1. Data Sharing. Data sharing refers to the sharing of data owned by the data owner (sender) with other users. Encryption becomes essential when sharing data safely over a network [3–5]. In the absence of encryption, data may be exposed or tampered with by eavesdropping events during the communication process. In addition, to share encrypted data, the receiver must be able to decrypt the data. If the sender encrypts data using the symmetric key method, the symmetric key must be safely delivered to the receiver. This, however, is difficult to achieve. Conversely, when using the asymmetric key (public key) method, the sender and receiver can share data by exchanging only the public key with each other. However, in both of the above two methods, the sender and the receiver must both remain online until the data transmission is completed, which is not possible at times. To solve this problem, a data sharing method using a cloud (proxy) has been proposed: after uploading data to the cloud, the sender can use the method of allowing access to the data stored in the cloud according to the request of the receiver. Encryption is indispensable even when using this method to ensure the contents of the data is not exposed. However, to decrypt data encrypted with the sender's public key using the receiver's private key, it is necessary to encrypt the data with the receiver's public key after decryption in the cloud; however, during this process, the data are inevitably exposed to the cloud. Therefore, PRE has been proposed to ensure that the private key or the contents of the data are not exposed while sharing data through the cloud.

2.2. Proxy Reencryption. PRE delegates decryption authority to other users by reencrypting data through a proxy represented by the cloud. As shown in Figure 1, the sender encrypts data with his/her public key, uploads it to the cloud, and generates a reencryption key at the request of the receiver and sends it to the cloud. Upon receiving the ciphertext and the reencryption key, the cloud reencrypts the ciphertext to generate a reencrypted ciphertext and transmits it to the receiver. PRE was first introduced in 1998 by Blaze et al. [6]. Since then, a number of traditional public key cryptography (PKC) PREs have been proposed [7–17, 18]. Such a PRE requires a public key certificate to prove the validity of the public key. However, the generation and storage of public key certificates involve considerable overhead. To address this, several ID-based PRE (IBPRE) methods [19] have been proposed [14, 20, 21]. However, the key generation center (KGC) generates the full private key of each user in IBPRE, which gives rise to the key escrow problem. To solve the key escrow problem, certificateless PRE (CL-PRE) has been proposed [22–24, 25]. In CL-PRE, the KGC does not generate the user's full private key but generates a partial key and delivers it to the user. As a result, the KGC cannot know the user's private key.

2.3. Broadcast Proxy Reencryption. Broadcast encryption (BE) is a technique first introduced by Berkovits [26]. BE is an access control technology designed to securely transmit data such as digital media, notifications or messages, and distance education to multiple recipients. BE can be divided into a symmetric broadcast encryption method and an asymmetric broadcast encryption method according to an encryption method. The symmetric broadcast encryption method is a method of delivering data to multiple receivers using a symmetric key method. Representative examples include Berkovits' scheme [26], Naor et al.'s scheme [27], and Halevy and Shamir's scheme [28]. This symmetric broadcast encryption method makes it difficult to distribute and manage keys.

On the other hand, asymmetric broadcast encryption is a method of delivering data to multiple recipients using a public key method. Therefore, the roles of encryption and decryption can be distinguished by utilizing the easy key

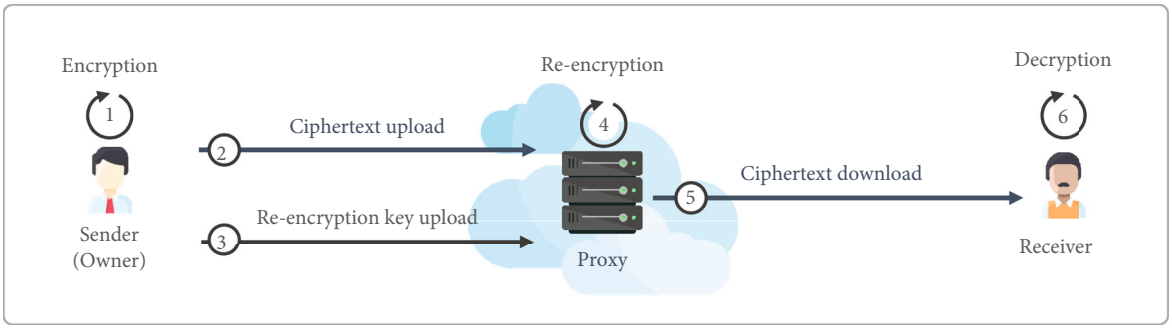

Figure 1: Basic form of proxy reencryption.

distribution and management, which are the advantages of the existing public key encryption method. Asymmetric broadcast encryption was first proposed by Dodis and Fazio [29] in 2002. However, this scheme has the disadvantage that the size of the encryption key is too large. In [30], Delerablée et al. proposed a scheme to perform BE using a dynamic method to target unpredictable users. Since then, BE schemes such as [31–34, 35] have been proposed.

Meanwhile, with the development of communication and storage technologies, the movement to utilize cloud storage has gradually increased. Also, a method for sharing data stored in cloud storage to other users was required. However, in order to make the encrypted data stored in the cloud available to other users, it is difficult to reencrypt after decryption or to deliver the decryption key. These problems increase the network load and reduce the efficiency. Therefore, PRE was proposed to solve this problem and research was conducted to deliver data to multiple recipients using cloud storage by combining this PRE with BE.

Chu et al. first proposed CPBRE by combining conditional proxy reencryption with BE [36]. Since then, various broadcast proxy reencryption (BPRE) has been proposed as shown in Figure 2. BPRE, proposed by Wang et al. in 2009, provides recipient anonymity. Also, data distribution is controlled by the KGC or broadcast center (BC). However, it requires a high computational overhead by using bilinear pairing and a key escrow problem also appears. Various methods of research were also conducted in the relatively recently proposed studies of Maiti and Misra [37], Sun et al. [38], Yin et al. [39], and Chunpeng et al. [40]. However, both of these methods use bilinear pairing and incur high computational overhead. In addition, there is a problem that the key escrow problem occurs.

3. Preliminaries

This section describes the basic environment and settings used to understand the scheme proposed in this study. To this end, the system model, security requirements, and algorithm used in the proposed scheme are explained.

3.1. System Model. The system model used in this study, shown in Figure 3, comprises a *sender*, *receiver*, *cloud* (proxy), and the *KGC*.

(i) *Sender*: the sender is the owner of the data and the user with whom the data are shared. The sender encrypts the plaintext with his/her public key and uploads it to the cloud. Then, to distribute the data, a reencryption key is generated and transmitted to the cloud. The sender can also download the data that he/she uploaded to the cloud and decrypt it with his/her private key to obtain the plaintext

(ii) *Receiver*: the receiver receives sender's data. The receiver may receive the reencrypted ciphertext from the cloud and decrypt the data with his/her private key to obtain the plaintext

(iii) *Cloud* (proxy): the cloud is assumed to be the same object as the remote proxy server. Because the cloud is a semitrusted server, it comes with a danger of data leakage. Therefore, the sender must apply data encryption to safely store data in the cloud. In addition, during data sharing, the plaintext or the user's private key should not be exposed to the cloud. Finally, users with legitimate rights should be able to access and use data in the cloud at any time

(iv) *Key generation center* (KGC): the KGC is an object that generates and issues a user key. Although the KGC is involved in generating each user's private key, in this study, to solve the key escrow problem, the KGC does not generate the user's full private key but generates a partial key and delivers it to each user. In addition, all users must use the public parameters created by the KGC to perform data encryption, decryption, and reencryption.

3.2. Security Requirements. This study consists of seven security requirements. The details are as follows:

(i) *Confidentiality*: the data that are kept in the proxy and the data delivered through the proxy shall not be unknown other than the authorized user. To do this, the data must be encrypted using the encryption key and the user who does not have a legitimacy decryption key should not be able to decrypt the contents

(ii) *Integrity*: data uploaded and shared by the sender must not be changed without permission in the

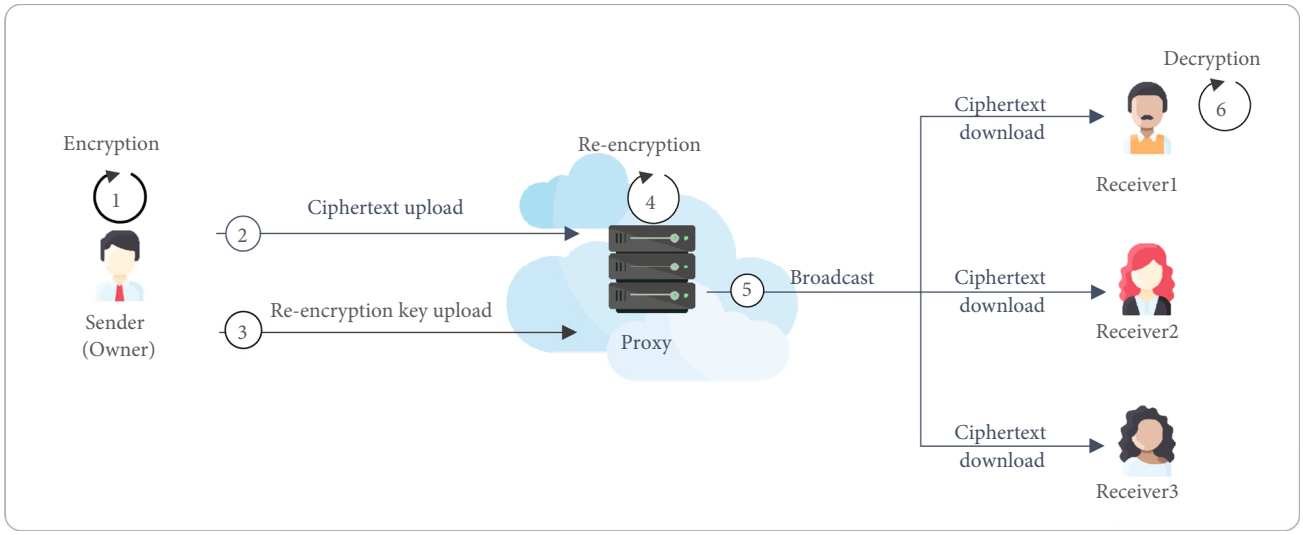

Figure 2: Broadcast proxy reencryption.

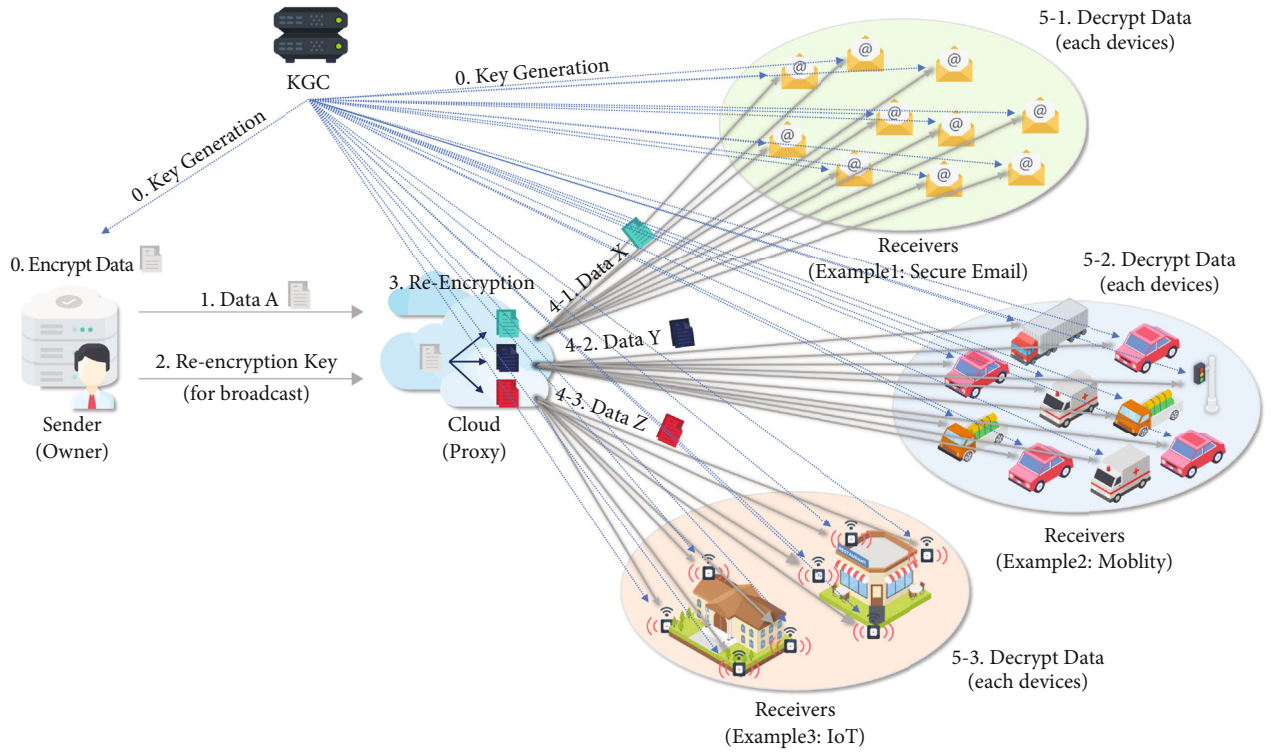

Figure 3: Form of the proposed system model.

process of being delivered to the cloud and the receiver and stored in the cloud. If at all the contents are changed, the sender or receiver who shares the data must be made aware of the change

(iii) *Key escrow problem*: all users who want to use the cloud must communicate with the KGC to generate a private key and public key pair. In this process, the KGC generates a user's full private key and the KGC may arise the user's authority. This problem is called a key escrow problem, and a method for solving this problem is required

(iv) *Partial key verifiability*: to solve the previously described key escrow problem, a key generation method in the form of a partial key can be used. In this case, each user must be able to verify whether the partial key generated and issued by the KGC to each user is legitimately generated by the correct KGC

(v) *Receiver anonymity*: the reencrypted ciphertext in cloud storage can be decrypted by a number of designated receivers. For this purpose, the reencryption key and reencrypted ciphertext include information generated by the public key of each receiver. However, privacy issues arise when such information allows a particular recipient or third party to identify another receiver

(vi) *Decryption fairness*: each legitimate receiver designated by the sender can decrypt the reencrypted ciphertext. However, in this process, a specific receiver should not be discriminated against or disadvantaged in the decryption process by a specific receiver or a third party

3.3. Algorithms. A total of 10 algorithms were used in the scheme proposed in this study. The purpose and details of each algorithm are as follows.

(i) Setup $(\lambda) \longrightarrow (\mathrm{msk}, \mathrm{mpk})$: this algorithm is performed by the KGC, which generates KGC's master secret key msk and master public key mpk for each user to use the cloud and publishes the mpk

(ii) Set-secret-value $(\mathrm{mpk}) \longrightarrow (T_i, \mathrm{ID}_i)$: this algorithm is performed by the user, wherein user i generates T_i using randomly selected t_i and mpk and sends it to the KGC along with ID_i

(iii) Partial-key-extract $(T_i, \mathrm{ID}_i, \mathrm{msk}, \mathrm{mpk}) \longrightarrow (R_i, k_i)$: this algorithm is performed by the KGC, which generates partial keys (R_i, k_i) of user i using T_i and ID_i transmitted by user i and its own msk and mpk and delivers it to user i

(iv) Set-private-key $(t_i, R_i, k_i, \mathrm{mpk}) \longrightarrow \mathrm{sk_i}$: this algorithm is performed by the user, wherein user i generates his/her own private key sk_i using the partial keys (R_i, k_i) received from the KGC. The generated private key sk_i was kept secure

(v) Set-public-key $(t_i, R_i, \mathrm{mpk}) \longrightarrow \mathrm{pk}_i$: this algorithm is performed by the user, wherein user i generates his/her public key pk_i using the partial key $(R_i, k_i$) received from the KGC and the secret value t_i generated by the user. The generated public key p k_i is made public so that anyone can use it

(vi) Enc $(\mathrm{pk}_{\mathbb{S}}, \mathrm{ID}_{\mathbb{S}}, m, \mathrm{mpk}) \longrightarrow \mathrm{CT}$: this algorithm is performed by the sender, wherein sender $\mathbb{S}$ encrypts his/her data $m \in M$ using public key $\mathrm{pk}_{\mathbb{S}}$ to obtain ciphertext CT and uploads it to the cloud

(vii) Re-key-gen $(\mathrm{sk}_{\mathbb{S}}, \mathrm{pk}_{\mathbb{R}}, \mathrm{ID}_{\mathbb{S}}, \mathrm{ID}_{\mathbb{R}}, \mathrm{mpk}) \longrightarrow \mathrm{rk}_{\mathbb{S}\longrightarrow\mathbb{R}}$: this algorithm is performed by the sender, wherein sender $\mathbb{S}$ specifies a receiver set $\mathbb{R} = (\curlywedge_1, \curlywedge_2, \cdots, \curlywedge_n)$ of receivers $\curlywedge_j (1 \le j \le n)$ to share their data with, generates a reencryption key for $\mathbb{R}$, and delivers it to the cloud

(viii) Re-enc $(\mathrm{CT}, \mathrm{rk}_{\mathbb{S}\longrightarrow\mathbb{R}}, \mathrm{mpk}) \longrightarrow \mathrm{CT}_R$: this algorithm is performed by the cloud, wherein the cloud reencrypts ciphertext CT of sender $\mathbb{S}$ using reencryption key $\mathrm{rk}_{\mathbb{S}\longrightarrow\mathbb{R}}$ of sender $\mathbb{S}$ to obtain reencrypted ciphertext CT_R

(ix) Dec-1 $(\mathrm{CT}, \mathrm{sk}_{\mathbb{S}}, \mathrm{ID}_{\mathbb{S}}, \mathrm{mpk}) \longrightarrow m$: this algorithm is performed by the sender, wherein sender $\mathbb{S}$ downloads his/her ciphertext CT stored in the cloud and then uses his/her private key $\mathrm{sk}_{\mathbb{S}}$ to decrypt it to obtain plaintext m

(x) Dec-2 $(\mathrm{CT}_R, \mathrm{sk}_j, \mathrm{ID}_j, \mathrm{mpk}) \longrightarrow m$: this algorithm is performed by the receiver, wherein receiver $\curlywedge_j$ downloads ciphertext CT_R stored in the cloud and then uses his/her private key sk_j to decrypt it to obtain plaintext m

4. Proposed CL Broadcast Proxy Reencryption

This section describes the scheme proposed in this study. For this, a technical overview, system parameters, and algorithm construction are described.

4.1. Technical Overview. The basic model of BRE, shown in Figure 4, can be broadly divided into four phases: *a setup phase*, *key generation phase*, *data storage phase*, and *data broadcast phase*. More details about these phases are presented in Sections 4.2 and 4.3.

4.2. System Parameters. The following are the system parameters used in this proposed scheme.

(i) $*$: participants (KGC, sender $\mathbb{S}$, receiver set $\mathbb{R}$, receiver $\curlywedge_j$, and user i)

(ii) p, q: λ-bit prime integer

(iii) E: elliptic curve

(iv) F_q: finite field for q

(v) λ: security parameter

(vi) l_1, l_2: length of message space (determined by the λ)

(vii) P: random generator in G_q $(P \in G_q)$

(viii) G: additive group on elliptic curve E

(ix) G_q: subgroup of G with prime order q

(x) ID_*: identity of participant $*$ $(\mathrm{ID}_* \in \{0,1\}^*)$

(xi) msk: KGC's system master secret key

(xii) mpk: KGC's system master public key

(xiii) sk_i: user i's full private key

(xiv) pk_i: user i's full public key

(xv) $\mathrm{rk}_{\mathbb{S}\longrightarrow\mathbb{R}}$: reencryption key (sender $\mathbb{S}$ delegates to receiver set $\mathbb{R}$)

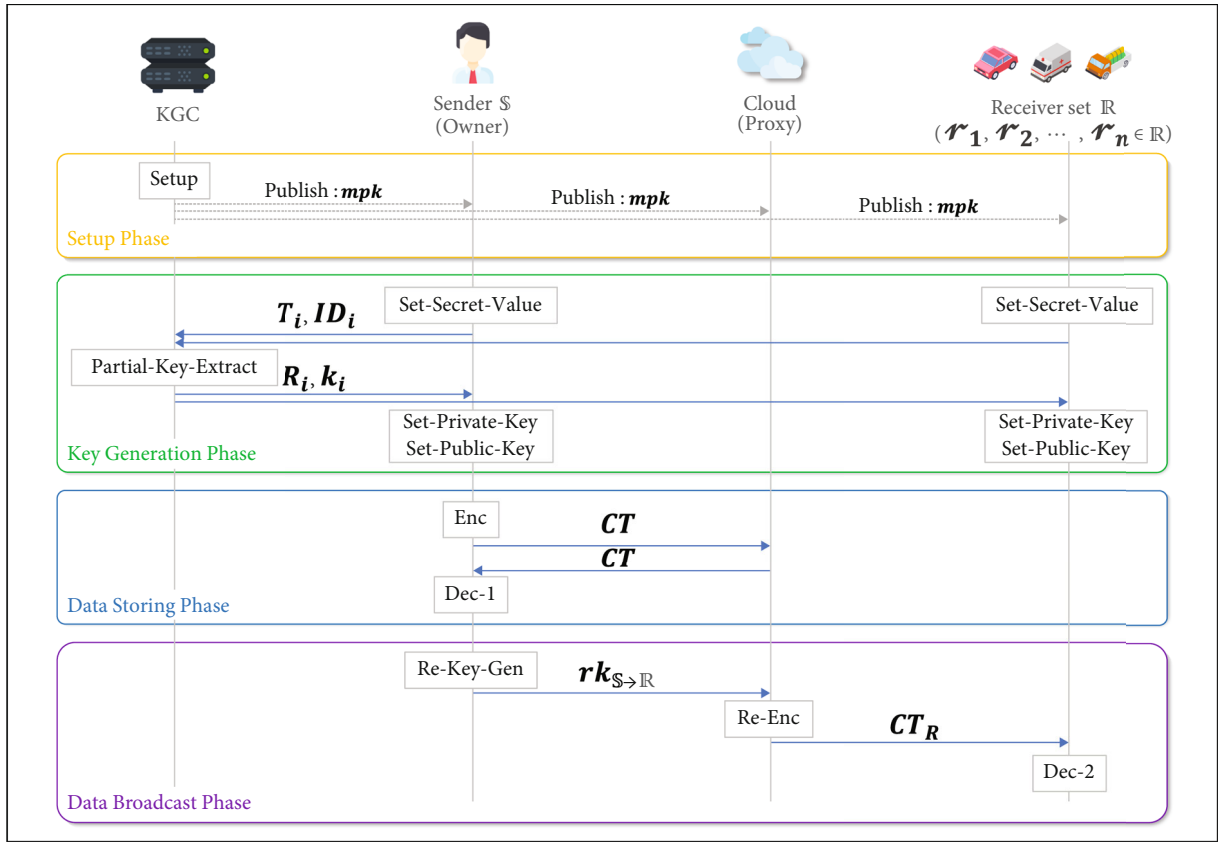

Figure 4: Overview of the proposed scheme.

(xvi) M: message space

(xvii) m: plaintext (message) $(m \in M)$

(xviii) CT: ciphertext

(xix) CT_R: reencrypted ciphertext

(xx) H_1 : one-way hash function $Z_q^* \longrightarrow Z_q^*$

(xxi) H_2 : one-way hash function $\{0,1\}^{l_1+l_2} \longrightarrow Z_q^*$

(xxii) H_3 : one-way hash function $G_q \times \{0,1\}^* \longrightarrow Z_q^*$

(xxiii) H_4 : one-way hash function $Z_q^* \times Z_q^* \longrightarrow \{0,1\}^{l_1+l_2}$

(xxiv) H_5 : one-way hash function $Z_q^* \times \{0,1\}^* \times \{0,1\}^* \longrightarrow Z_q^*$

(xxv) H_6 : one-way hash function $Z_q^* \longrightarrow \{0,1\}^{l_2}$

(xxvi) H_7 : one-way hash function $G_q \times G_q \longrightarrow Z_q^*$

4.3. Construction. The overall structure of this proposed scheme is shown in Figure 4. This scheme is mainly composed of four phases, each of which is composed of the *setup phase*, *key generation phase*, *data storage phase*, and *data broadcast phase*. A detailed description of each phase proceeds in each phase.

4.3.1. Setup Phase. This phase includes the *setup* algorithm. This phase is performed by the KGC in advance so that each user can use the cloud. Here, a master public key that can be commonly used by each user and a master secret key known only to the KGC are generated.

(i) Setup $(\lambda) \longrightarrow$ (msk, mpk): this algorithm is an algorithm performed by the KGC. With the security parameter λ as input, the KGC performs the following process

(1) Choose two λ-bit prime integers p, q and an elliptic curve E defined on F_p. Let G be the additive group on elliptic curve E and G_q be the subgroup of G with prime order q

(2) Select randomly a generator $P \in G_q$

(3) Randomly choose $d \in Z_q^*$ as the msk and calculate $P_{\mathrm{pub}} = d \bullet P$ which is part of mpk

(4) Select five secure one-way hash functions are follows:

$$
\begin{aligned}
&H_1 : Z_q^* \longrightarrow Z_q^*,\\
&H_2 : \{0,1\}^{l_1+l_2} \longrightarrow Z_q^*,\\
&H_3 : G_q \times \{0,1\}^* \longrightarrow Z_q^*,\\
&H_4 : Z_q^* \times Z_q^* \longrightarrow \{0,1\}^{l_1+l_2},\\
&H_5 : Z_q^* \times \{0,1\}^* \times \{0,1\}^* \longrightarrow Z_q^*,\\
&H_6 : Z_q^* \longrightarrow \{0,1\}^{l_2},\\
&H_7 : G_q \times G_q \times \{0,1\}^* \longrightarrow Z_q^*,
\end{aligned}
\tag{1}
$$

where l_1 and l_2 mean the length of the bit string and is determined by the security parameter λ

(5) Publish the system's maser public key $\text{mpk} = \{p, q, l_1, l_2, E, G, G_q, P, P_{\text{pub}}, H_1, H_2, H_3, H_4, H_5, H_6, H_7\}$ and message space $M = \{0,1\}^{l_1}$

4.3.2. Key Generation Phase. This phase includes *set-secret-value*, *partial-key-extract*, *set-private-key*, and *set-public-key* algorithms. In this phase, each user generates their own private key and public key pair so that they can use the cloud. In this phase, each user communicates with the KGC to receive a partial key and uses the partial key to generate their own public key and private key pair as shown in Figure 5.

(i) *Set-secret-value*: this algorithm is an algorithm performed by user i. A user i randomly selects $t_i \in Z_q^*$ and keeps it secure. User i computes $T_i = t_i \bullet P$ as the public key, and user i sends (T_i, ID_i) to the KGC

(ii) *Partial-key-extract*: this algorithm is an algorithm performed by the KGC. According to the identity ID_i of user i, the KGC performs the following steps:

(1) Randomly select $r_i \in Z_q^*$ and compute $R_i = r_i \bullet P$

(2) Calculate a part of the partial private key k_i as follows:

$$
k_i \longleftarrow r_i + dH_3(R_i, T_i, \text{ID}_i) + H_3(dT_i, \text{ID}_i) \,(\text{mod } q) \tag{2}
$$

(3) After that, partial key (R_i, k_i) is delivered to user i through the public channel

(iii) *Set-private-key*: this algorithm is an algorithm performed by user i. After receiving partial key (R_i, k_i) from the KGC, user i verifies these like equations (3) and (4). If verification passes, user i is compute private key $\text{sk}_i = (s_i, t_i)$ as the following steps:

(1) Verify whether the following equation holds:

$$
k_i \bullet P \stackrel{?}{=} R_i + H_7(R_i, T_i, \text{ID}_i) P_{\text{Pub}} + H_3(t_i P_{\text{Pub}}, \text{ID}_i) P \tag{3}
$$

(2) If not, return $\perp$; otherwise, user i computes s_i as follows:

$$
s_i \longleftarrow k_i - H_3(t_i P_{\text{Pub}}, \text{ID}_i) \tag{4}
$$

(3) After that, user i keeps secret $\text{sk}_i = (s_i, t_i)$ as his/her the full private key

(iv) *Set-public-key*: this algorithm is an algorithm performed by user i. User i keeps $\text{pk}_i = (R_i, T_i)$ as the full public key

4.3.3. Data Storing Phase. This phase includes the Enc and Dec-1 algorithms. This phase represents the process of the sender encrypting his/her data with his/her public key and storing it in the cloud. In addition, the sender downloads his/her own data stored in the cloud and a decryption process is also included using the private key to obtain the data source again as shown in Figure 6.

(i) *Enc*: this algorithm is an algorithm performed by the sender $\mathbb{S}$. Sender $\mathbb{S}$ encrypts message m with ciphertext CT by entering his/her public key $\text{pk}_{\mathbb{S}} = (R_{\mathbb{S}}, T_{\mathbb{S}})$ and message $m \in M$. Then, upload the ciphertext CT to the cloud

(1) Compute w, z, and Z using the given message $m \in M$ and $\text{pk}_{\mathbb{S}} = (R_{\mathbb{S}}, T_{\mathbb{S}})$

$$
\begin{aligned}
w &\longleftarrow H_7(R_{\mathbb{S}}, T_{\mathbb{S}}, \text{ID}_{\mathbb{S}}),\\
z &\longleftarrow H_2(m \| w),\\
Z &\longleftarrow zP
\end{aligned}
\tag{5}
$$

(2) Then, sender $\mathbb{S}$ calculates $U_{\mathbb{S}}$ using z and $\text{pk}_{\mathbb{S}} = (R_{\mathbb{S}}, T_{\mathbb{S}})$

$$
U_{\mathbb{S}} \longleftarrow z \bullet \left(R_{\mathbb{S}} + H_7(R_{\mathbb{S}}, T_{\mathbb{S}}, \text{ID}_{\mathbb{S}}) P_{\text{pub}} + T_{\mathbb{S}}\right) \tag{6}
$$

(3) Sender $\mathbb{S}$ calculates α, θ, and C as follows:

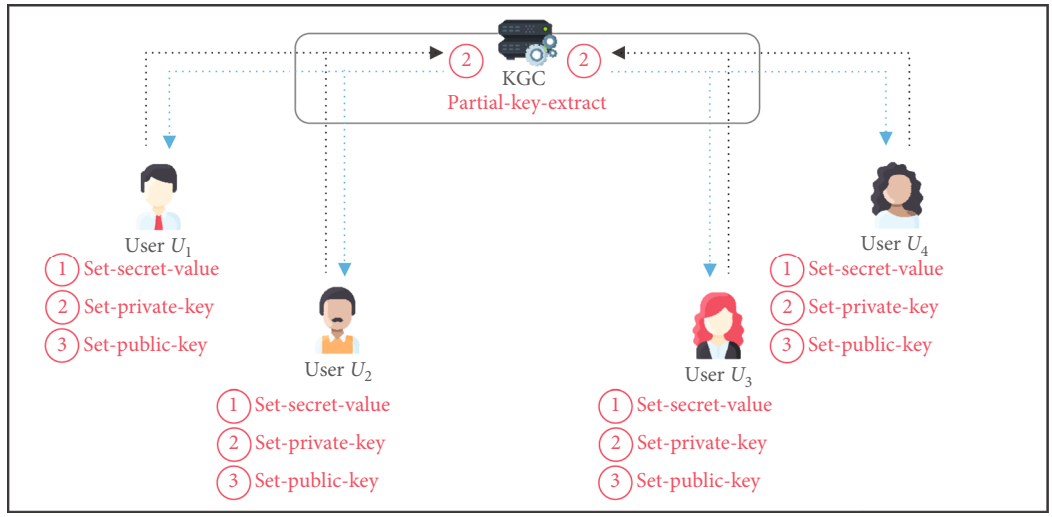

Figure 5: Key generation phase.

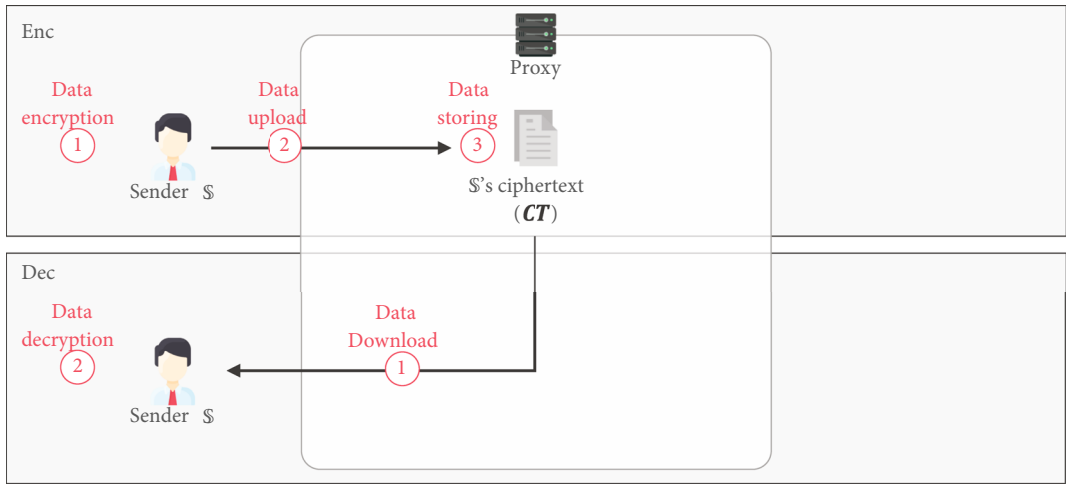

Figure 6: Data storing phase.

$$\begin{aligned} \alpha &\longleftarrow H_1(s_{\mathbb{S}} \cdot t_{\mathbb{S}}), \\ \theta &\longleftarrow H_1(U_{\mathbb{S}} \cdot \alpha), \\ C &\longleftarrow H_4(Z, \theta) \oplus (m \| w) \end{aligned} \tag{7}$$

(4) Generate ciphertext $\mathrm{CT} \longleftarrow (C_1, C_2) = (Z, C)$. Then, the generated CT is uploaded and stored to the cloud

(ii) *Dec-1*: this algorithm is an algorithm performed by the sender $\mathbb{S}$. The sender $\mathbb{S}$ can download the ciphertext $\mathrm{CT} = (C_1, C_2) = (Z, C)$ from cloud. The sender $\mathbb{S}$ who has downloaded the ciphertext CT can obtain the plaintext m by decrypting the ciphertext CT with his/her private key $\mathrm{sk}_{\mathbb{S}} = (s_{\mathbb{S}}, t_{\mathbb{S}})$

(4) Sender $\mathbb{S}$ calculates $U'_{\mathbb{S}}$ using its private key $\mathrm{sk}_{\mathbb{S}} = (s_{\mathbb{S}}, t_{\mathbb{S}})$ and the given ciphertext $\mathrm{CT} = (C_1, C_2, C_3)$

$$U'_{\mathbb{S}} \longleftarrow (s_{\mathbb{S}} + t_{\mathbb{S}}) \bullet C_1 \tag{8}$$

(5) Calculate α and θ' by inputting $\mathrm{sk}_{\mathbb{S}}$ and $U'_{\mathbb{S}}$

$$\alpha \longleftarrow H_1(s_{\mathbb{S}} \cdot t_{\mathbb{S}}), \tag{9}$$

$$\theta' \longleftarrow H_1\left(U'_{\mathbb{S}} \cdot \alpha\right) \tag{10}$$

(6) Calculate m by inputting C_1, C_2, θ'

$$(m\|w) \longleftarrow C_2 \oplus H_4\left(C_1, \theta'\right), \tag{11}$$

$$\begin{aligned} \because C_2 \oplus H_4\left(C_1, \theta'\right) &= H_4(Z, \theta) \oplus (m\|w) \oplus H_4\left(C_1, \theta'\right) \\ &= H_4(Z, \theta) \oplus (m\|w) \oplus H_4\left(Z, \theta'\right) = (m\|w), \end{aligned} \tag{12}$$

where $C_1 = Z$

(7) Verify whether the following equation holds. If not, return $\perp$; otherwise, sender $\mathbb{S}$ keeps plaintext m

$$C_1 \stackrel{?}{=} H_2(m\|H_7(R_{\mathbb{S}}, T_{\mathbb{S}}, ID_{\mathbb{S}}))P, \tag{13}$$

$$\because C_1 = H_2(m\|H_7(R_{\mathbb{S}}, T_{\mathbb{S}}, \mathrm{ID}_{\mathbb{S}}))P = H_2(m\|w)P = zP = Z, \tag{14}$$

where $Z = zP$ and $z = H_2(m\|w)$

4.3.4. Data Broadcast Phase. This phase includes *re-key-gen*, *re-enc*, and *dec-2* algorithms. In this phase, the sender generates a reencryption key for a set of recipients and passes it to the proxy. After receiving the reencryption key, the proxy reencrypts the encrypted data and broadcasts it to the recipients. The receiver who has received the broadcast ciphertext can obtain the message by decrypting the ciphertext with their private key as shown in Figure 7.

(i) *Re-key-gen*: this algorithm is executed by sender $\mathbb{S}$ to delegate a ciphertext to set of recipients $\mathbb{R} = (\curlyvee_1, \curlyvee_2, \cdots, \curlyvee_n)$ of selected receiver $\curlyvee_j$ with identity ID_j $(1 \le j \le n)$. The following steps will be performed in this algorithm

(1) Compute U_j, where $j = 1, 2, \cdots, n$.

$$U_j \longleftarrow z \bullet \left(R_j + H_7\left(R_j, T_j, \mathrm{ID}_j\right) P_{\mathrm{pub}} + T_j\right) \tag{15}$$

(2) Compute a polynomial $f(x)$ with degree n using $\beta \in Z_q^*$ as follows:

$$f(x) = \prod_{i=0}^{n} \left(x - U_j\right) + \beta \,(\mathrm{mod}\ q) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0, \tag{16}$$

where, $a_i \in Z_p^*$ $(i = 0, 1, \cdots, n-1)$

(3) Compute x using $\mathrm{sk}_{\mathbb{S}}$, α, and β^{-1} as follows:

$$x \longleftarrow (s_{\mathbb{S}} + t_{\mathbb{S}}) \cdot \alpha \cdot \beta^{-1} \tag{17}$$

(4) Sender $\mathbb{S}$ generates reencryption key $\mathrm{rk}_{\mathbb{S} \longrightarrow \mathbb{R}} = (rk_1, rk_2) = (x, \{a_0, a_1, \cdots, a_{n-1}\})$ and sends $\mathrm{rk}_{\mathbb{S} \longrightarrow \mathbb{R}}$ to cloud

(ii) *Re-Enc*: this algorithm is executed by cloud. This algorithm reencrypts ciphertext CT to ciphertext C T_R using reencryption key $rk_{\mathbb{S} \longrightarrow \mathbb{R}}$. The following steps will be performed in this algorithm

(1) Compute CT_R using ciphertext CT and reencryption key $\mathrm{rk}_{\mathbb{S} \longrightarrow \mathbb{R}}$

$$C_1' \longleftarrow C_1, \tag{18}$$

$$C_2' \longleftarrow C_2, \tag{19}$$

$$C_3' \longleftarrow \mathrm{rk}_1 \cdot C_1, \tag{20}$$

$$C_4' \longleftarrow \mathrm{rk}_2 \tag{21}$$

(2) Output $\mathrm{CT}_R = (C_1', C_2', C_3', C_4')$ and send CT_R to receivers $\mathbb{R}$

(iii) *Dec-2*: this algorithm is executed by the selected receiver $\curlyvee_j$ to extract the plaintext from the received ciphertext $\mathrm{CT}_R = (C_1', C_2', C_3', C_4')$. Receiver $\curlyvee_j$ performs following steps:

(1) Compute U_j

$$U_j' \longleftarrow \left(s_j + t_j\right) \bullet C_1' \tag{22}$$

(2) Generate polynomial $f(x)$ and compute β'

$$f(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0, \tag{23}$$

$$\beta' = f\left(U_j'\right) \tag{24}$$

(3) Compute θ' as input C_3' and β'

$$\theta' = H_1\left(C_3' \cdot \beta'\right), \tag{25}$$

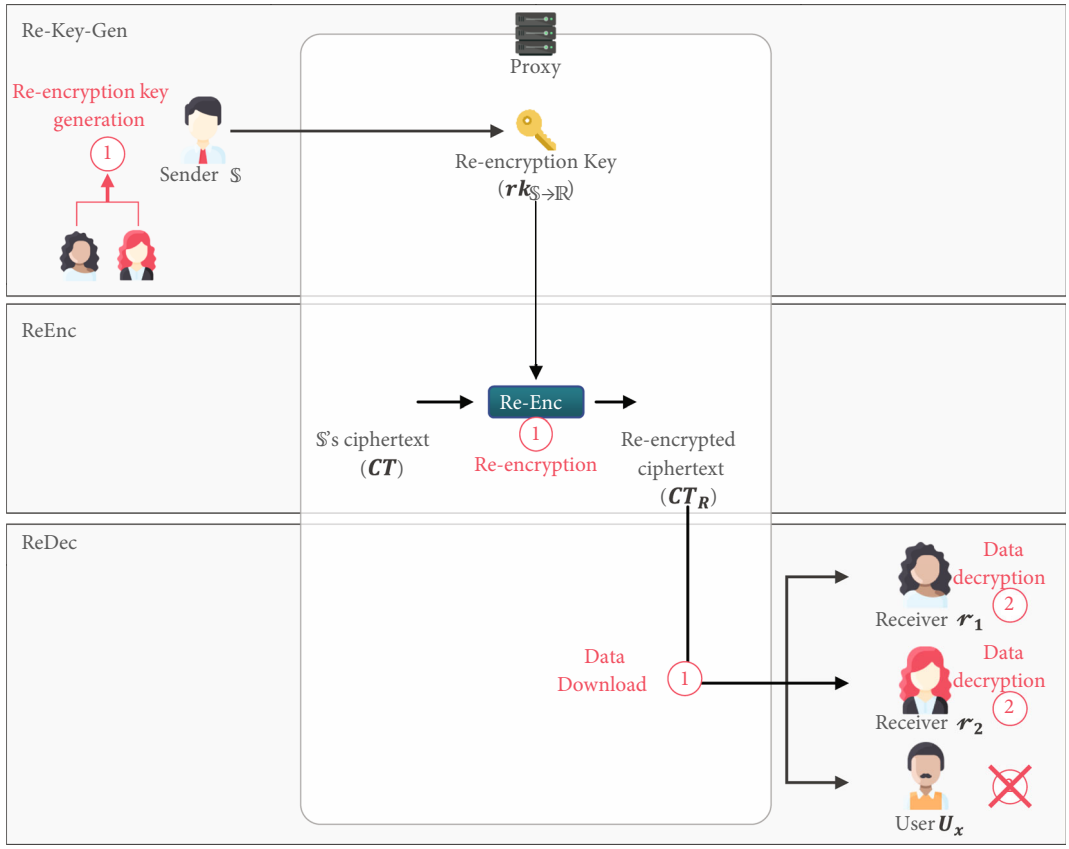

Figure 7: Data broadcast phase.

$$\because C_3' \cdot \beta' = \mathrm{rk}_1 \cdot C_1 \cdot \beta' = x \cdot C_1 \cdot \beta' = (s_{\mathbb{S}} + t_{\mathbb{S}}) \cdot \alpha \cdot \beta^{-1} \cdot C_1 \cdot \beta' = (U_{\mathbb{S}}) \cdot \alpha \tag{26}$$

(4) Compute m as input C_1', C_2', θ'

$$m \longleftarrow C_2' \oplus H_4\left(C_1', \theta'\right), \tag{27}$$

$$\because C_2' \oplus H_4\left(C_1', \theta'\right) = H_4(Z, \theta) \oplus m \oplus H_4\left(C_1', \theta'\right) = H_4(Z, \theta) \oplus m \oplus H_4\left(Z, \theta'\right) = m, \tag{28}$$

where $C_1' = C_1 = Z$

(5) Verify message m. If not, return $\perp$; otherwise, receiver i outputs the plaintext m

$$C_1' \stackrel{?}{=} H_2(m)P, \tag{29}$$

$$\because C_1' = H_2(m)P = zP = Z, \tag{30}$$

where $Z = zP$ and $z = H_2(m)$

4.4. Correctness. In this section, we will prove the correctness of the scheme proposed in Section 4. First, Theorem 1 describes in detail the execution process of the *set-private-key* algorithm, which is a process in which the user verifies whether the partial key received from the KGC is a correct value. Second, Theorem 2 describes in detail the execution process of the *Dec-1* algorithm, which is an algorithm for the sender to decrypt his/her data. Finally, Theorem 2 describes in detail the execution process of the *Dec-2* algorithm, which is an algorithm for the receiver to decrypt the reencrypted data.

Theorem 1. *User i can verify whether the partial key (R_i, k_i) received from the KGC is a value generated from the (T_i, ID_i) created by him/her and the mpk of the correct KGC. This process corresponds to equations (2)–(4).*

Proof. Assuming that one of the users is $\mathcal{U}_1$, $\mathcal{U}_1$ can perform the following process using (R_1, k_1) received from the KGC and its own value (T_1, ID_1) and KGC's master public key mpk.□ □

$\mathcal{U}_1$ can verify whether the received partial key (R_i, k_i) is correct by using the (T_i, ID_i) and mpk. This process corresponds to equation (3).

$$
\begin{aligned}
k_i \bullet P &\stackrel{?}{=} R_1 + H_7(R_1, T_1, \mathrm{ID}_i)P_{\mathrm{Pub}} + H_1(t_1 P_{\mathrm{Pub}}, \mathrm{ID}_1)P, \\
\because k_1 \bullet P &= r_1 \bullet P + H_7(R_1, T_1, \mathrm{ID}_1) \cdot d \cdot P + H_3(t_1 P_{\mathrm{Pub}}, \mathrm{ID}_1)P \\
&= (r_1 + H_7(R_1, T_1, \mathrm{ID}_1) \cdot d + H_3(t_1 \cdot d \cdot P, \mathrm{ID}_1))P \\
&= (r_1 + d \cdot H_7(R_1, T_1, \mathrm{ID}_1) + H_3(T_1 \cdot d_1, \mathrm{ID}_1))P = (k_1)P,
\end{aligned} \tag{31}
$$

where $k_i = r_i + dH_7(R_i, T_i, \mathrm{ID}_i) + H_3(dT_i, \mathrm{ID}_i)$.

Theorem 2. *The sender $\mathbb{S}$ can perform decryption using the ciphertext CT received from the cloud and his/her private key and obtain the plaintext m. This process corresponds to equations (8)–(13).*

Proof. Assuming that one of the senders is $\mathbb{S}$, $\mathbb{S}$ can perform the following process using $\mathrm{CT} = (C_1, C_2, C_3, C_4)$ received from the sender and its own private key $\mathrm{sk}_{\mathbb{S}} = (s_{\mathbb{S}}, t_{\mathbb{S}})$.□ □

$\mathbb{S}$ creates $U'_{\mathbb{S}}$ as follows using his/her private key $\mathrm{sk}_{\mathbb{S}} = (s_{\mathbb{S}}, t_{\mathbb{S}})$. This process corresponds to equations (8) and (9).

$$
\begin{aligned}
U'_{\mathbb{S}} &\longleftarrow (s_{\mathbb{S}} + t_{\mathbb{S}}) \bullet Z, \\
U_{\mathbb{S}} &= z \bullet \left(R_{\mathbb{S}} + H_7(R_{\mathbb{S}}, T_{\mathbb{S}}, \mathrm{ID}_{\mathbb{S}})P_{\mathrm{pub}} + T_{\mathbb{S}}\right) \\
&= z \bullet (r_{\mathbb{S}} P + H_7(R_{\mathbb{S}}, T_{\mathbb{S}}, \mathrm{ID}_{\mathbb{S}})dP + t_{\mathbb{S}} P) \\
&= z \bullet (r_{\mathbb{S}} + H_7(R_{\mathbb{S}}, T_{\mathbb{S}}, \mathrm{ID}_{\mathbb{S}})d + t_{\mathbb{S}})P \\
&= (r_{\mathbb{S}} + dH_7(R_{\mathbb{S}}, T_{\mathbb{S}}, \mathrm{ID}_{\mathbb{S}}) + t_{\mathbb{S}})Z \\
&= (r_{\mathbb{S}} + dH_7(R_{\mathbb{S}}, T_{\mathbb{S}}, \mathrm{ID}_{\mathbb{S}}) + H_3(dT_{\mathbb{S}}, \mathrm{ID}_{\mathbb{S}}) \\
&\quad - H_3(dT_{\mathbb{S}}, \mathrm{ID}_{\mathbb{S}}) + t_{\mathbb{S}})Z = (k_{\mathbb{S}} - H_7(t_{\mathbb{S}} P_{\mathrm{Pub}}, \mathrm{ID}_{\mathbb{S}}) \\
&\quad + t_{\mathbb{S}})Z = (s_{\mathbb{S}} + t_{\mathbb{S}})Z = (s_{\mathbb{S}} + t_{\mathbb{S}})C_1 = U'_{\mathbb{S}}.
\end{aligned} \tag{32}
$$

$\mathbb{S}$ obtains θ' using the generated equations (9) and (10) as follows:

$$
\begin{aligned}
\alpha &\longleftarrow H_1(s_{\mathbb{S}} \cdot t_{\mathbb{S}}), \\
\theta' &\longleftarrow H_1\left(U'_{\mathbb{S}} \cdot \alpha\right).
\end{aligned} \tag{33}
$$

$\mathbb{S}$ can obtain m as follows using C_1, C_2 and the acquired θ' in equation (11).

$$
\begin{aligned}
(m \| w) &\longleftarrow C_2 \oplus H_4\left(C_1, \theta'\right), \\
\because C_2 \oplus H_4\left(C_1, \theta'\right) &= C \oplus H_4\left(Z, \theta'\right) \\
&= H_4(Z, \theta) \oplus (m \| w) \oplus H_4\left(Z, \theta'\right) \\
&= (m \| w),
\end{aligned} \tag{34}
$$

where $\mathrm{CT} = (C_1, C_2) = (Z, C)$ and $C = H_4(Z, \theta) \oplus (m \| w)$.

Theorem 3. *The receivers $\mathbb{R}$ can perform decryption using the reencrypted ciphertext CT_R received from the cloud and his/her private key and obtain the plaintext m. This process corresponds to equations (18)–(29).*

Proof. Assuming that one of the receivers is $\mathscr{r}_1$, $\mathscr{r}_1$ can perform the following process using $\mathrm{CT}_R = (C'_1, C'_2, C'_3, C'_4)$ received from the sender and its own private key $\mathrm{sk}_{\mathscr{r}_1} = (s_{\mathscr{r}_1}, t_{\mathscr{r}_1})$.□ □

$\mathscr{r}_1$ creates $U_{\mathscr{r}_1}$ as follows using his/her private key $\mathrm{sk}_{\mathscr{R}} = (s_{\mathscr{R}}, t_{\mathscr{R}})$. This process corresponds to equations (18) and (22).

$$
\begin{aligned}
U'_{\mathscr{r}_1} &\longleftarrow \left(s_{\mathscr{r}_1} + t_{\mathscr{r}_1}\right) \bullet Z, \\
U_{\mathscr{r}_1} &= z \bullet \left(R_{\mathscr{r}_1} + H_7\left(R_{\mathscr{r}_1}, T_{\mathscr{r}_1}, \mathrm{ID}_{\mathscr{r}_1}\right)P_{\mathrm{pub}} + T_{\mathscr{r}_1}\right) \\
&= z \bullet \left(r_{\mathscr{r}_1} P + H_7\left(R_{\mathscr{r}_1}, T_{\mathscr{r}_1}, \mathrm{ID}_{\mathscr{r}_1}\right)dP + t_{\mathscr{r}_1} P\right) \\
&= z \bullet \left(r_{\mathscr{r}_1} + H_7\left(R_{\mathscr{r}_1}, T_{\mathscr{r}_1}, \mathrm{ID}_{\mathscr{r}_1}\right)d + t_{\mathscr{r}_1}\right)P \\
&= \left(r_{\mathscr{r}_1} + dH_7\left(R_{\mathscr{r}_1}, T_{\mathscr{r}_1}, \mathrm{ID}_{\mathscr{r}_1}\right) + t_{\mathscr{r}_1}\right)Z \\
&= \left(r_{\mathscr{r}_1} + dH_7\left(R_{\mathscr{r}_1}, T_{\mathscr{r}_1}, I\mathscr{r}_1\right)\right. \\
&\quad \left. + H_3\left(dT_{\mathscr{r}_1}, \mathrm{ID}_{\mathscr{r}_1}\right) - H_3\left(dT_{\mathscr{r}_1}, \mathrm{ID}_{\mathscr{r}_1}\right) + t_{\mathscr{r}_1}\right)Z \\
&= \left(k_{\mathscr{r}_1} - H_7\left(t_{\mathscr{r}_1} P_{\mathrm{Pub}}, \mathrm{ID}_{\mathscr{r}_1}\right) + t_{\mathscr{r}_1}\right)Z = \left(s_{\mathscr{r}_1} + t_{\mathscr{r}_1}\right)Z \\
&= \left(s_{\mathscr{r}_1} + t_{\mathscr{r}_1}\right)C_1 = U'_{\mathscr{r}_1}.
\end{aligned} \tag{35}
$$

$\mathscr{r}_1$ obtains θ' using the generated equations (23)–(25) as follows:

$$
\begin{aligned}
\beta' \longleftarrow f\left(U_{\mathscr{r}_1}\right) &= \prod_{i=0}^{n}\left(U_{\mathscr{r}_1} - U_i\right) + \beta \,(\mathrm{mod}\ q) \\
&= \left(U_{\mathscr{r}_1} - U_{\mathscr{r}_1}\right) \bullet \left(U_{\mathscr{r}_1} - U_{\mathscr{r}_2}\right) \cdots \left(U_{\mathscr{r}_1} - U_{\mathscr{r}_i}\right) \\
&\quad + \beta \,(\mathrm{mod}\ q) = 0 \bullet \left(U_{\mathscr{r}_1} - U_{\mathscr{r}_2}\right) \cdots \left(U_{\mathscr{r}_1} - U_{\mathscr{r}_i}\right) \\
&\quad + \beta \,(\mathrm{mod}\ q).
\end{aligned} \tag{36}
$$

$\mathscr{r}_1$ can obtain m using C'_1, C'_2 and the acquired θ' in equation (27).

$$
\begin{aligned}
(m \| w) &\longleftarrow C'_2 \oplus H_4\left(C'_1, \theta'\right), \\
\because C'_2 \oplus H_4\left(C'_1, \theta'\right) &= C \oplus H_4\left(Z, \theta'\right) \\
&= H_4(Z, \theta) \oplus (m \| w) \oplus H_4\left(Z, \theta'\right) \\
&= (m \| w),
\end{aligned} \tag{37}
$$

where $\mathrm{CT}_R = (C'_1, C'_2, C'_3, C'_4) = (Z, C, x \cdot Z, \mathrm{rk}_2)$ and $C = H_4(Z, \theta) \oplus (m \| w)$.

Table 1: Comparison of the security requirements.

	Bilinear pairing	Key escrow problem	Receiver anonymity	Re-key-generation
Wang and Yang [41]	Used	Insecure	Offer	KGC/BC
Maiti and Misra [37]	Used	Insecure	Offer	Sender
Sun et al. [38]	Used	Insecure	Offer	Sender
Yin et al. [39]	Used	Insecure	Offer	Sender
Chunpeng et al. [40]	Used	Insecure	Offer	Sender
Proposed scheme	Not used	Secure	Offer	Sender

5. Analysis of the Proposed CL-BPRE Scheme

In this section, we analyze the efficiency of the proposed scheme and explain how to achieve data security. In addition, the advantages of the proposed scheme are explained through a comparison with previous studies.

5.1. Analysis of Security Requirements. We analyze whether the proposed scheme is successful in achieving the security requirements presented in Section 3.2. There are a total of 7 security requirements, each of which is *confidentiality*, *integrity*, *key escrow problem*, *partial key verifiability*, *receiver anonymity*, and *decryption fairness* as shown in Table 1.

(i) *Confidentiality*: in the proposed scheme, an ECC-based encryption operation is performed to provide data confidentiality. In this process, the message itself is not encrypted with the public key of each recipient but a session key is created to encrypt the message. Therefore, to decrypt a message, a session key must be obtained, and to obtain a session key, only a legitimate recipient must carry out the computation

$$f(x) = \prod_{i=0}^{n}(x-\mu_i) + \beta \,(\mathrm{mod}\ q) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0. \tag{38}$$

Here, μ_i is $\mu_i \longleftarrow H_3(U_i, \mathrm{ID}_i, w)$ and U_i is $U_i \longleftarrow z\bullet(R_i + H_7(R_i, T_i, \mathrm{ID}_i)P_{\mathrm{Pub}} + T_i)$. Therefore, each recipient must have its own private key to generate μ_i and the user who generates μ_i can obtain the θ through the following process:

$$\begin{aligned}\beta' \longleftarrow f(\mu_i) &= \prod_{i=0}^{n}(\mu_i - \mu_i) + \beta \,(\mathrm{mod}\ q)\\ &= (\mu_1 - \mu_1)\bullet(\mu_1 - \mu_2)\cdots(\mu_1 - \mu_n) + \beta \,(\mathrm{mod}\ q)\\ &= 0\bullet(\mu_1 - \mu_2)\cdots(\mu_1 - \mu_n) + \beta \,(\mathrm{mod}\ q).\end{aligned} \tag{39}$$

μ_i can obtain the θ through the following process:

$$m \longleftarrow C \oplus H_4(Z, \theta). \tag{40}$$

If an attacker attempts to create U_i' with only the public key of the recipient i, U_i' cannot be generated normally, as follows:

$$U_i' \longleftarrow \overset{?}{z}\bullet(R_i + H_7(R_i, T_i, \mathrm{ID}_i)P_{\mathrm{Pub}} + T_i). \tag{41}$$

According to the above formula, because the attacker does not know z, it is impossible to forge U_i'.

(ii) *Integrity*: recipients who have decrypted the data can verify the integrity of the data using the values contained in the integrity ciphertext and the parameters of the public KGC, as in Theorem 3 of Section 4.4. The proof of this is as follows:

$$\begin{aligned}C_1' &\overset{?}{=} H_2(m\|w)P,\\ \because C_1' &= H_2(m\|w)P = zP = Z,\end{aligned} \tag{42}$$

where $Z = zP$ and $z = H_2(m\|w)$.

The receiver that decrypts ciphertext CT_R can obtain message m and verification value w. Here, $H_2(m\|w)$ is equal to z, and thus, the integrity of the message can be verified by comparing whether $H_2(m)P$ is equal to $C_1 = Z$.

(iii) *Key escrow problem*: the proposed scheme uses a certificateless PKC method that has been demonstrated to be successful in solving the key escrow problem. To solve the key escrow problem, the KGC must not know the user's complete private key. In the existing IBC, in the private-key-gen algorithm, the KGC generates a user's complete private key and delivers it to each user. The key escrow problem is by dividing the key generation process into four algorithms: *set-secret value*, *partial-key-extract*, *set-private-key*, and *set-public-key*

First, the *set-secret-value* algorithm generates T_i using secret value t_i, randomly selected by the user, and the master public key value P of the KGC. At this time, t_i is safely stored only by the user and T_i and ID_i are delivered to the KGC through an open channel.

In the *partial-key-extract* algorithm, using the T_i and ID_i received by the KGC from the user, partial keys R_i and k_i are generated through the following process and delivered to the user.

$$\begin{aligned}R_i &= r_i\bullet P,\\ k_i &\longleftarrow r_i + dH_7(R_i, T_i, \mathrm{ID}_i) + H_3(dT_i, \mathrm{ID}_i)(\mathrm{mod}\ P).\end{aligned} \tag{43}$$

In the *set-private-key* algorithm, the private key is calculated using the R_i and k_i received by the user from the KGC.

At this time, the user does not use (R_i, k_i) as the private key, but uses t_i, which only the user knows, and s_i, which is generated based on the k_i received from the KGC, as the private key.

$$s_i \longleftarrow k_i - H_3(t_i P_{\text{Pub}}, \text{ID}_i). \tag{44}$$

Finally, in the *set-public-key* algorithm, T_i generated by the user and R_i generated by the KGC are used as public keys.

As a result, the KGC must know t_i to obtain the user's private key $\text{sk}_i = (s_i, t_i)$. However, because t_i is a secret value stored safely by the user, the key escrow problem by the KGC does not occur.

The KGC only knows $\text{pk}_i = (T_i, R_i)$ and k_i, and the KGC cannot know $\text{sk}_i = (s_i, t_i)$.

(iv) *Partial key verifiability*: the proposed scheme is designed to satisfy several security requirements. In this process, there was an increase in the amount of computation

$$\begin{aligned} k_i \bullet P &\stackrel{?}{=} R_i + H_7(R_i, T_i, \text{ID}_i) P_{\text{Pub}} + H_3(t_i P_{\text{Pub}}, \text{ID}_i) P, \\ \because k_i \bullet P &= r_i \bullet P + H_7(R_i, T_i, \text{ID}_i) \cdot d \cdot P + H_3(t_i P_{\text{Pub}}, \text{ID}_i) P \\ &= (r_i + H_7(R_i, T_i, \text{ID}_i) \cdot d + H_3(t_i \cdot d \cdot P, \text{ID}_i)) P \\ &= (r_i + d \cdot H_7(R_i, T_i, \text{ID}_i) + H_3(T_i \cdot d_i, \text{ID}_i)) P = (k_i) P, \end{aligned} \tag{45}$$

where, $k_i = r_i + dH_7(R_i, T_i, \text{ID}_i) + H_3(dT_i, \text{ID}_i)$,

$$\begin{aligned} R_i &= r_i P, \\ T_i &= t_i P, \\ P_{\text{pub}} &= dP. \end{aligned} \tag{46}$$

Through the above calculation, user i can know that the partial key that it has received is based on secret value r_i generated by user i and that it is generated by a legitimate KGC

(v) *Receiver anonymity*: in the proposed scheme, a Lagrange interpolation polynomial is applied to provide the recipient's anonymity. In this method, the information of the user included in the polynomial cannot be obtained because the recipient is only confirmed by a polynomial. The formula for this polynomial is as follows:

$$\begin{aligned} f(x) &= \prod_{i=0}^{n} (x - \mu_i) + \beta \,(\text{mod } q) \\ &= (x - \mu_1) \bullet (x - \mu_2) \bullet \cdots \bullet (x - \mu_n) + \beta \,(\text{mod } q) \\ &= x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0, \end{aligned}$$

$$U_j' = (s_j + t_j) \bullet C_1' \tag{47}$$

To identify a specific recipient in the above polynomial, it is possible to generate μ_i' of the specific receiver. However, as in the confidentiality item above, an attacker cannot forge U_i'.

$$U_i' \longleftarrow \stackrel{?}{z} \bullet (R_i + H_7(R_i, T_i, \text{ID}_i) P_{\text{Pub}} + T_i). \tag{48}$$

As a result, the attacker cannot identify the recipient.

(vi) *Decryption fairness*: in the decoding process of the recipient data included in the recipient list, the decoding should not be disadvantageous because of the intervention of a third party or the KGC. To this end, in the proposed scheme, it is not possible to change the list of recipients by configuring a polynomial or adding an amount of computation by designating only specific recipients. This takes advantage of the property of the following polynomial, and in order for an attacker to make a specific recipient disadvantageous, he/she must be able to completely forge $f(x)$, a polynomial that targets all receivers.

$$\begin{aligned} f(x) &= \prod_{i=0}^{n} (x - \mu_i) + \beta \,(\text{mod } q) \\ &= (x - \mu_1) \bullet (x - \mu_2) \bullet \cdots \bullet (x - \mu_n) \\ &\quad + \beta (\text{mod } q) = x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0. \end{aligned} \tag{49}$$

5.2. Comparison of Schemes

(i) *Security requirements*: the proposed scheme of this study was designed to satisfy various requirements that the existing schemes do not provide. Wang and Yang [41] and Maiti and Misra [37], and Sun et al. [38] proposed IBC-based BPRE. Since these two schemes operate in the IBC method, the KGC generates and issues the user's private key. Since these two schemes operate in the IBC method, the KGC generates and issues the user's private key. Sur et al. generated the user's private key through the $\text{keygen}_{\text{IBE}}$ algorithm as follows:

$$\text{sk}_{\text{ID}} = (d_0, d_1, d_2) = \left(g^{\alpha_2} \left(g_1^{\text{ID}} g \right)^u, g^u, g^{u/\alpha} \right), \tag{50}$$

where master secret key $\text{mk} = g_2^{\alpha}$ and random value $\alpha, u, x \in Z_p^{*}$.

According to the above formula, each user's private key can be generated only by the KGC that owns the master secret key and a complete private key is generated, which may cause a key escrow problem. Maiti et al. also generated the user's private key in the KG algorithm. In this process, the KGC generates a complete private key through the

Table 2: Comparison of the computation efficiency.

	Enc	Re-key-gen	Re-enc	Dec-2
Wang and Yang [41]	$(2)T_M + (4)T_e + (1)T_P$	$(10+3n)T_M + (1)T_e$	$(6)T_e$	$(7)T_M + (7)T_e + (5)T_P$
Maiti and Misra [37]	$(4)T_M + (3)T_e$	$(3+n^2+n)T_M + (3+n)T_e$	$(1)T_M + (1)T_P$	$(1)T_M + (2)T_P$
Sun et al. [38]	$(2+\mathrm{n})T_M + (5)T_e + (1)T_P$	$(3+n)T_M + (6)T_e + (1)T_p$	$(1+\mathrm{n})T_M + (2)T_P$	$(4+\mathrm{n})T_M + (2)T_e$
Yin et al. [39]	$(4+2n)T_M + (4)T_e$	$(5+n)T_M + (6)T_e$	$(4+3n)T_M + (2)T_e + (2)T_p$	$(7)T_M + (1)T_e + (3)T_P$
Chunpeng et al. [40]	$(2)T_M + (3)T_e$	$(5+n)T_M + (5+\mathrm{n})T_e + (1)T_p$	$(1)T_M + (2)T_\mathrm{p}$	$(6)T_M + (2)T_e + (2)T_p$
Proposed scheme	$(2)T_{\mathrm{EM}} + (2)T_{\mathrm{EA}}$	$(1+n)T_M + (2n)T_{\mathrm{EM}} + (2n)T_{\mathrm{EA}}$	$(1)T_{\mathrm{EM}}$	$(2)T_{\mathrm{EM}}$

T_M: computation time of modular multiplication operation; T_{EM}: computation time of ECC multiplication operation; T_{EA}: computation time of ECC point add operation; T_e: computation time of exponent operation; T_p: computation time of bilinear pairing operation.

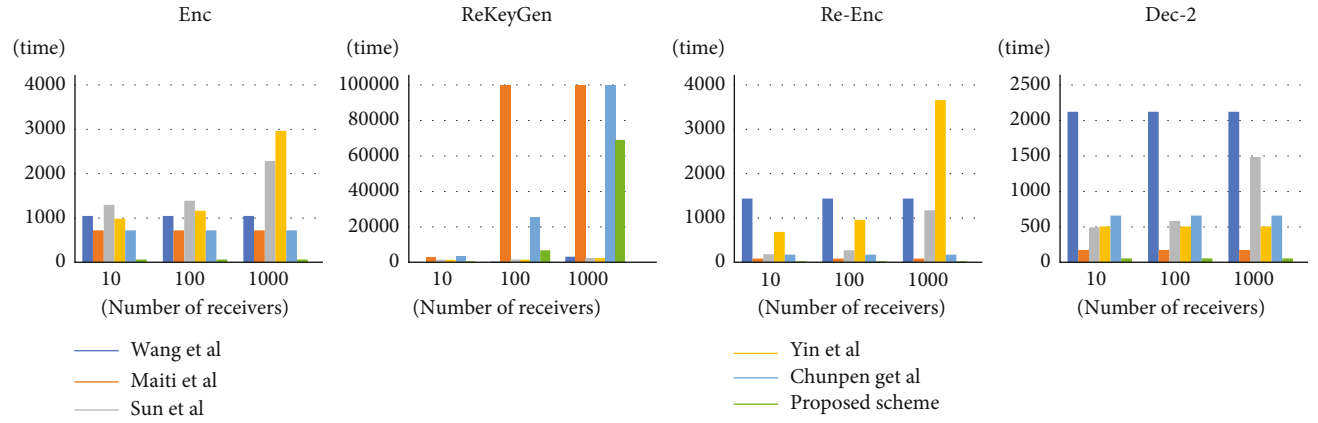

Figure 8: Computation time of schemes.

following operation, which may cause a key escrow problem.

$$\mathrm{sk}_{\mathrm{ID}_A} = g^{1/(\alpha+H_1(\mathrm{ID}_A))}, \tag{51}$$

where the system secret key is α.

Sun et al. also generated the user's private key in the key-gen algorithm as follows.

$$\mathrm{sk}_i = g_i^{\gamma}, \tag{52}$$

where system secret key msk = γ.

This also causes Sun et al. key escrow problems. In addition, both Wang et al. and Maiti et al., Sun et al., Yin et al., and Chunpeng et al. use a pairing operation, which takes a lot of computation time. Yin et al. and Chunpeng et al. all proposed certificateless BPRE. However, since all three methods use a pairing operation, a lot of computation time is required. Additionally, Yin et al. pose a threat of privacy invasion because the anonymity of the recipient is not guaranteed.

$$\mathrm{rk} = \left(\mathrm{rk}_1, \mathrm{rk}_2, \mathrm{rk}_3, \mathrm{rk}_4, \mathrm{rk}_5, (\mathrm{rk}_{6,i})_{i\in\{1,2,\cdots,k+1\}}\right), \tag{53}$$

where $\mathrm{rk}_{6,i} = \mu_i^s$, for

$$i \in \{1, 2, \cdots, k+1\}. \tag{54}$$

Therefore, the proposed scheme uses the CL method and does not use the pairing operation, so that the BPRE can be performed in less time. In addition, it is possible to use BPRE more safely and efficiently by solving the problem of key escrow and recipient anonymity.

(ii) *Computational efficiency*: the proposed scheme of this study was designed with a lower number of calculations compared to the existing schemes as shown in Table 2. Since the pairing operation is not basically used, BPRE can be performed with less computation time compared to the existing methods. In addition, it has a lower number of operations by simplifying encryption and decryption operations. However, the number of operations for generating the reencryption key has increased. Therefore, some computational efficiency may be lowered in an environment where the list of recipients is constantly changing. However, since the amount of data encryption and decryption operations is low, the burden on the user terminal can be reduced. The computation time chart is shown in Figure 8.

6. Conclusion

Data sharing using the cloud is related to data confidentiality and key management issues. First, the cloud is a semitrusted

environment and data can be exposed at any time by an insider or external attack. Therefore, in order to solve this problem, the application of encryption is essential. However, in order to share encrypted data with other users, it is essential to distribute a key that can decrypt the data. However, in a data storage and sharing environment using the cloud, it is very difficult to distribute the key because the key cannot be delivered face to face. To solve this problem, proxy reencryption has been proposed that allows data being stored encrypted in the cloud to be shared with other users. Proxy reencryption is a technology that reencrypts data that has been encrypted once to data that other users can decrypt without having to decrypt the data and share the private key. However, since the existing proxy reencryption can reencrypt by specifying only one recipient at a time, if the number of recipients increases, the number of reencryption also increases and the number of times to generate a reencryption key for reencryption also increases. Therefore, broadcast proxy reencryption has been proposed to solve this problem. Broadcast proxy reencryption is a combination of broadcast encryption technology and proxy reencryption technology. The broadcast encryption method is effective when distributing the same data to multiple recipients at the same time because multiple recipients can be specified with only one encryption. By combining these features with proxy reencryption, broadcast encryption can be used when data encrypted once is shared with multiple users at the same time. Therefore, it can be applied to various environments such as update servers that distribute data to many recipients at the same time, secure email, and IoT. However, the receiver anonymity, key escrow problem, decryption fairness, partial key verification problem, etc. that appear in the broadcast encryption method also appear in the broadcast proxy reencryption. Therefore, you can safely use broadcast proxy reencryption only after solving these problems. To this end, in this study, in the process of designating a plurality of receivers, the receiver cannot be identified using a polynomial and the receiver is additionally modulated by modulating the polynomial to change the receiver or a specific receiver is designed so that it does not have a disadvantage in decoding. In addition, by not using the pairing operation in this process, the calculation time is reduced and the amount of calculation is simplified, so that data can be broadcast even with a lower operation. In the key generation process, the key escrow problem caused by KGC was solved by using the certificateless method instead of the existing IBC type. As a result, the proposed scheme solves the security threats of the existing schemes and at the same time reduces the amount of computation and the computation time, so that it is possible to provide a more secure and efficient broadcast proxy reencryption.

Data Availability

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

Conflicts of Interest

The authors declare no conflict of interest.

Authors' Contributions

Won-Bin Kim and Su-Hyun Kim contributed equally to this work.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2019R1A2C1085718) and the Soonchunhyang University Research Fund and the BK21 FOUR (Fostering Outstanding Universities for Research) (No. :5199990914048).

References

[1] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial IoTs," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 38, no. 5, pp. 968–979, 2020.

[2] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science and Engineering (TNSE)*, vol. 7, no. 2, pp. 766–775, 2020.

[3] Z. Cai and Z. He, "Trading private range counting over big IoT data," in *The 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019)*, Dallas, TX, USA, 2019.

[4] S. Ji, J. He, A. S. Uluagac, R. Beyah, and Y. Li, "Cell-based snapshot and continuous data collection in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 9, no. 4, pp. 1–29, 2013.

[5] S. Ji and Z. Cai, "Distributed data collection in large-scale asynchronous wireless sensor networks under the generalized physical interference model," *IEEE/ACM Transactions on Networking*, vol. 21, no. 4, pp. 1270–1283, 2013.

[6] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 127–144, Konstanz, Germany, 1998.

[7] 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*, vol. 9, no. 1, pp. 1–30, 2006.

[8] D. Robert, H. J. Weng, S. Liu, and K. Chen, "Chosen-ciphertext secure proxy re-encryption without pairings," in *International Conference on Cryptology and Network Security*, pp. 1–17, Hong Kong, China, 2008.

[9] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," in *International Workshop on Public Key Cryptography*, pp. 360–379, Barcelona, Spain, 2008.

[10] K. Varad and C. Pandu Rangan, "RSA-TBOS signcryption with proxy re-encryption," in *Proceedings of the 8th ACM workshop on Digital rights management*, Alexandria Virginia USA, 2008.

[11] S. Jun and Z. Cao, "CCA-secure proxy re-encryption without pairings," in *International Workshop on Public Key Cryptography*, pp. 357–376, Irvine, CA, USA, 2009.

[12] G. Ateniese, K. Benson, and S. Hohenberger, "Key-private proxy re-encryption," in *Cryptographers' Track at the RSA Conference*, pp. 279–294, San Francisco, CA, USA, 2009.

[13] J. Shao, P. Liu, G. Wei, and Y. Ling, "Anonymous proxy re-encryption," *Security and Communication Networks*, vol. 5, no. 5, 449 pages, 2012.

[14] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *International Conference on Applied Cryptography and Network Security*, Zhuhai, China, 2007.

[15] H. Wang and Z. Cao, "More efficient CCA-secure unidirectional proxy re-encryption schemes without random oracles," *Security and Communication Networks*, vol. 6, no. 2, 181 pages, 2013.

[16] S. S. M. Chow, J. Weng, Y. Yang, and R. H. Deng, "Efficient unidirectional proxy re-encryption," in *International Conference on Cryptology in Africa*, pp. 316–332, Stellenbosch, South Africa, 2010.

[17] G. Hanaoka, Y. Kawai, N. Kunihiro, T. Matsuda, and J. Weng, "Generic construction of chosen ciphertext secure proxy re-encryption," in *Cryptographers' Track at the RSA Conference*, pp. 349–364, San Francisco, CA, USA, 2012.

[18] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pp. 185–194, Alexandria Virginia USA, 2007.

[19] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Annual International Cryptology Conference*, pp. 213–229, Santa Barbara, CA, USA, 2001.

[20] T. Matsuo, "Proxy re-encryption systems for identity-based encryption," in *International Conference on Pairing-Based Cryptography*, pp. 247–267, Tokyo, Japan, 2007.

[21] C. K. Chu and W. G. Tzeng, "Identity-based proxy re-encryption without random oracles," in *International Conference on Information Security*, pp. 189–202, Valparaíso, Chile, 2007.

[22] C. Sur, C. Jung, Y. Park, and K. Rhee, "Chosen-ciphertext secure certificateless proxy re-encryption," in *IFIP International Conference on Communications and Multimedia Security*, pp. 214–232, Linz, Austria, 2010.

[23] L. Xu, X. Wu, and X. Zhang, "CL-PRE: a certificateless proxy re-encryption scheme for secure data sharing with public cloud," in *Proceedings of the 7th ACM symposium on information, computer and communications security*, pp. 87-88, Seoul Korea, 2012.

[24] K. Yang, J. Xu, and Z. Zhang, "Certificateless proxy re-encryption without pairings," in *International Conference on Information Security and Cryptology*, pp. 67–88, Seoul, Korea (Republic of), 2013.

[25] A. Srinivasan and C. P. Rangan, "Certificateless proxy re-encryption without pairing: revisited," in *Proceedings of the 3rd International Workshop on Security in Cloud Computing*, pp. 41–52, Singapore Republic of Singapore, 2015.

[26] S. Berkovits, "How to broadcast a secret," in *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 535–541, Brighton, United Kingdom, 1991.

[27] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," in *Annual International Cryptology Conference*, pp. 41–62, Santa Barbara, CA, USA, 2001.

[28] D. Halevy and A. Shamir, "The LSD broadcast encryption scheme," in *Annual International Cryptology Conference*, pp. 47–60, Santa Barbara, CA, USA, 2002.

[29] Y. Dodis and N. Fazio, "Public key broadcast encryption for stateless receivers," in *ACM Workshop on Digital Rights Management*, pp. 61–80, Washington, DC, USA, 2002.

[30] C. Delerablée, P. Paillier, and D. Pointcheval, "Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys," in *International Conference on Pairing-Based Cryptography*, pp. 39–59, Tokyo, Japan, 2007.

[31] C. Delerablée, "Identity-based broadcast encryption with constant size ciphertexts and private keys," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 200–215, Kuching, Malaysia, 2007.

[32] C. Gentry and B. Waters, "Adaptive security in broadcast encryption systems (with short ciphertexts)," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 171–188, Cologne, Germany, 2009.

[33] J. Hur, C. Park, and S. O. Hwang, "Privacy-preserving identity-based broadcast encryption," *Information Fusion*, vol. 13, no. 4, pp. 296–303, 2012.

[34] Z. Zhou, D. Huang, and Z. Wang, "Efficient privacy-preserving ciphertext-policy attribute based-encryption and broadcast encryption," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 126–138, 2015.

[35] W. Kim, D. Seo, D. Kim, and I. Lee, "Data distribution for multiple receivers in a connected car environment using 5G communication," *Security and Communication Networks*, vol. 2021, Article ID 5599996, 14 pages, 2021.

[36] C. K. Chu, J. Weng, S. S. M. Chow, J. Zhou, and R. H. Deng, "Conditional proxy broadcast re-encryption," *Lecture Notes in Computer Science*, vol. 5594, pp. 327–342, 2009.

[37] S. Maiti and S. Misra, "P2B: privacy preserving identity-based broadcast proxy re-encryption," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5610–5617, 2020.

[38] M. Sun, C. Ge, L. Fang, and J. Wang, "A proxy broadcast re-encryption for cloud data sharing," *Multimedia Tools and Applications*, vol. 77, no. 9, pp. 10455–10469, 2018.

[39] S. Yin, H. Li, and L. Teng, "A novel proxy re-encryption scheme based on identity property and stateless broadcast encryption under cloud environment," *International Journal of Network Security*, vol. 21, no. 5, pp. 797–803, 2019.

[40] G. Chunpeng, Z. Liu, J. Xia, and F. Liming, "Revocable identity-based broadcast proxy re-encryption for data sharing in clouds," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, 2019.

[41] X. An Wang and X. Yang, "Identity based broadcast encryption based on one to many identity based proxy re-encryption," in *2009 2nd IEEE International Conference on Computer Science and Information Technology*, Beijing, China, 2009.