Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 7641389, 12 pages
https://doi.org/10.1155/2021/7641389

Research Article

Group Delegated ID-Based Proxy Reencryption for the Enterprise IoT-Cloud Storage Environment

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

[1]*Department of Software Convergence, Soonchunhyang University, Asan 31538, Republic of Korea*
[2]*Faculty of Artificial Intelligence and Data Engineering, Sangmyung University, Seoul 03016, Republic of Korea*
[3]*Department of Computer Science, Georgia State University, Atlanta 30303, USA*

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

Received 24 April 2021; Accepted 31 May 2021; Published 15 June 2021

Academic Editor: Zhihan Lv

In general, ID-based proxy reencryption (IBPRE) includes data transfer in a 1 : 1 manner between a sender and receiver. Therefore, only the data owner has the authority to decrypt or reencrypt the data that is encrypted with his/her public key. However, in an environment with data self-sovereignty, such as an enterprise IoT-cloud environment, the data are directly managed by cloud once data is uploaded from user-controlled IoT devices. In such a situation, there is no way of sharing data if the data owner has no access over the data due to being outside the workplace and other issues. In this study, to solve this problem, data can be shared even when the data cannot be accessed by delegating the authority of the data owner to generate the reencryption key to other users. In addition, by solving the security threats that may appear in this process, data sharing can be performed securely and efficiently in the corporate environment.

1. Introduction

Cloud storage technology is a technology that can store and use data remotely through a network by combining network technology with existing offline storage technology. When using cloud storage, the data owner uploads his or her data to the cloud storage. If another user requests this data in the future, it can be provided to the requestor through the provision of the data access path and authority of the cloud storage.

These cloud storage technologies are used not only for personal data storage purposes but also for securely storing and utilizing all data generated inhouse in a corporate environment. From the perspective of a company, when using cloud storage, data generated within the company can be securely stored and managed and can be used for other purposes as needed. However, if the data source is uploaded to the cloud storage in original form, the data content can be exposed to other users. Hence, encryption must be applied to solve this problem.

Encryption can be used for secure storage and sharing of data through cloud storage [1–5]. However, in general symmetric key encryption or public key encryption, it is difficult to change a key distribution problem or the user of already encrypted data. Therefore, proxy reencryption has been proposed to solve this problem. Proxy reencryption is a form of encryption technology that allows the sender to securely share data with the receiver [6, 7]. However, a feature that is different from the general encryption scheme is that the sender can provide data encrypted with his or her public key by converting it into an encrypted text that can be decrypted by the receiver with only the receiver's public key. This feature allows the sender to avoid performing decryption and encryption or sharing a secret key with the receiver in order to provide encrypted data to the receiver. Therefore, in a cloud storage environment, it is possible to provide the encrypted text generated by the user with his/her key by converting it into data encrypted with the requestor's key upon request.

However, a corporate-like environment has one additional property. For security reasons, access to cloud storage may be blocked from outside the enterprise. Let us assume that the data owner is outside the company on a business trip.

If data is requested from other employees and departments within the company, the request cannot be processed. In preparation for this situation, there are ways to share the private key with other employees, but this can lead to a serious security threat. Employees who have access to the private key of the owner can exercise all the rights of the owner. Therefore, it is necessary to be able to share designated data without sharing the private key. In this research, studies were conducted to satisfy these conditions. Our study provides an environment in which the owner of the data can provide the right to reencrypt specific data to other users. We propose a traceable group delegated ID-based proxy reencryption (IBPRE).

2. Related Works

This section describes related studies and theoretical constructs for understanding our study.

2.1. Proxy Reencryption. Proxy reencryption is an encryption technology that converts data encrypted with the sender's public key into data encrypted with the receiver's public key through a proxy. To this end, the sender generates a reencryption key using his/her own private key and the recipient's public key of the recipient and delivers it to the proxy. Upon receiving the reencryption key, the proxy reencrypts the sender's cipher text using the reencryption key to obtain the receiver's cipher text as shown in Figure 1(e) [6–8].

Thereafter, the recipient can obtain the plaintext from the ciphertext using his/her private key. The advantage of proxy reencryption is that the senders can share data without exposing their private keys or the original message to the proxy. Therefore, only the sender and receiver can know the source of the data.

IBPRE uses ID-based encryption to encrypt a user's ID as a public key or derive a public key from an ID. In this manner, a user can identify another user through an ID using a system such as a cloud and share data with the user [9–20]. Basically, proxy reencryption is a technique that allows data encrypted with the key of the data sender to be decrypted by a third party, the receiver, as shown in Figure 2.

In 2012, Xu et al. proposed a certificate-free IBPRE [18]. Noncertificate ID-based encryption is a technology that generates a public key through a user's ID without using a certificate. Therefore, the amount of computation is reduced compared to that in the scheme of generating the certificate. Additionally, it is secure with regards to the key escrow problem and the key exposure problem because the KGC (Key Generate Center) does not generate the key directly [19, 20].

As evident from the above algorithm configuration, this scheme assumes a form of traditional 1 : 1 data sharing. Therefore, it is difficult to resolve a situation in which a data owner cannot control data, such as an enterprise environment as covered in this study. However, in this scheme, the existing IBPRE has been applied to present the multiproxy crtificateless proxy reencryption (CL-PRE) and the randomized CL-PRE, which aims at diversity in the traditional IBPRE approach [21, 22].

2.2. Proxy Reencryption in the Enterprise Environment. In recent years, an increasing number of companies have begun operating private clouds. The cloud of a company is a technology that enables the storage and use of data produced within the company. It helps to efficiently manage sensitive data for data management and prevention of leakage. In general, it is necessary to distinguish between personal data and public data and even the data generated during work within the company. Therefore, data encryption should be applied in order to prevent other employees from viewing personal business data. Additionally, as encrypted data cannot be decrypted by a third party, the data owner must be able to decrypt the shared user with his or her private key in order to share data with the third party [23, 24]. As a result, the owner's ciphertext needs to be converted into the requestor's ciphertext through a process such as proxy reencryption [25, 26].

This process follows the IBPRE process described above. Therefore, data inside the company can be securely stored by blocking external access for data security as shown in Figure 3.

However, if the data owner is outside the premises of the organization, the data owner may not be able to share data as the owner cannot access the cloud outside the company as shown in Figure 3. In this case, the data is not transmitted, which causes a delay. Therefore, a scheme to solve this problem is required.

We hereby require a scheme for reencrypting data even when the owner of the data is absent. However, in the general IBPRE scheme, when there is a data request, a reencryption key is generated using the requestor's public key and the owner's private key. Therefore, it is impossible to create a reencryption key in advance because the data owner cannot know the identity of the data requestor before venturing out of the organization. As a result, a reencryption key must be generated whenever there is a data sharing request. However, reencryption is impossible because it is impossible to obtain the other party's public key and upload the reencryption key from outside the company. We have previously conducted research to solve this problem [27]. In the previous study, we tried to solve by using the group delegation method in a more restrictive environment. However, in previous studies, problems in computational aspects and some improvement in security aspects were required.

Therefore, in this study, we propose a scheme that allows the sender to generate a reencryption key through a delegatee. In this process, the group delegation is not used. Additionally, we present a scheme to trace the delegated reencryption key generator to confirm that the delegatee does not abuse the sender's authority.

3. System Model

This section describes the necessary mathematical basis before describing the proposed scheme and presents the construction and security requirements accordingly.

In this study, the concept of group delegated ID-based proxy reencryption (GD-PRE) is proposed. The form of this concept is shown in Figure 4. In the absence of the data

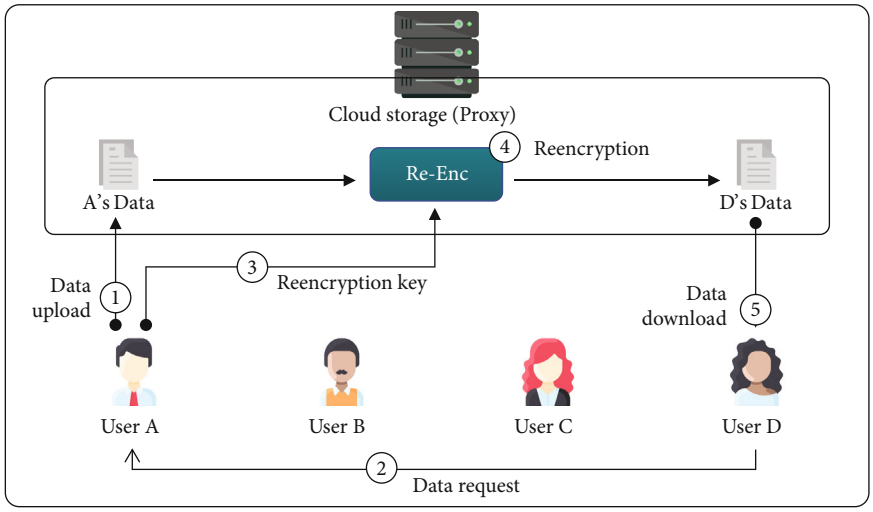

Figure 1: Basic form of proxy reencryption.

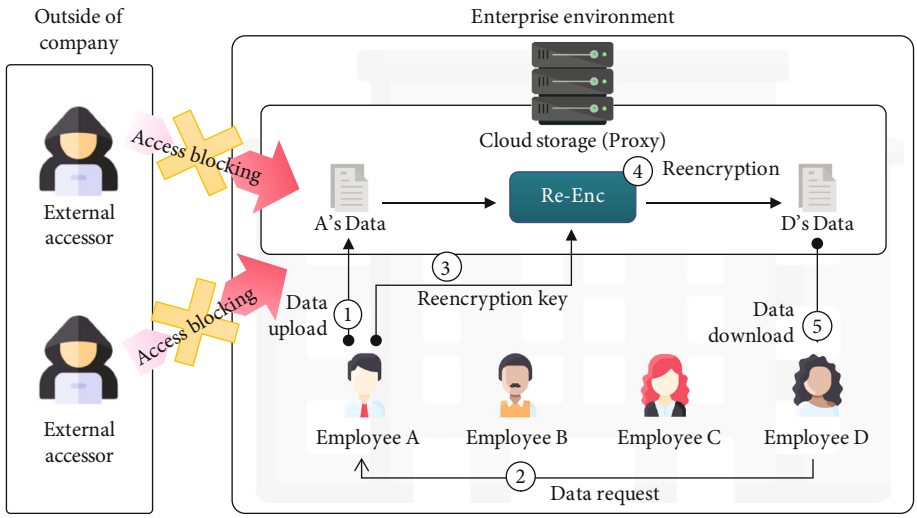

Figure 2: PRE in the enterprise environment.

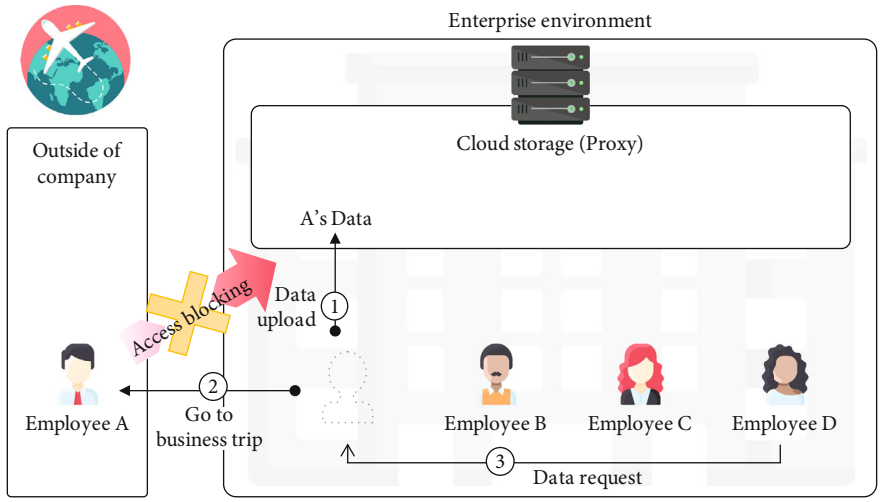

Figure 3: Problem of PRE in the enterprise environment.

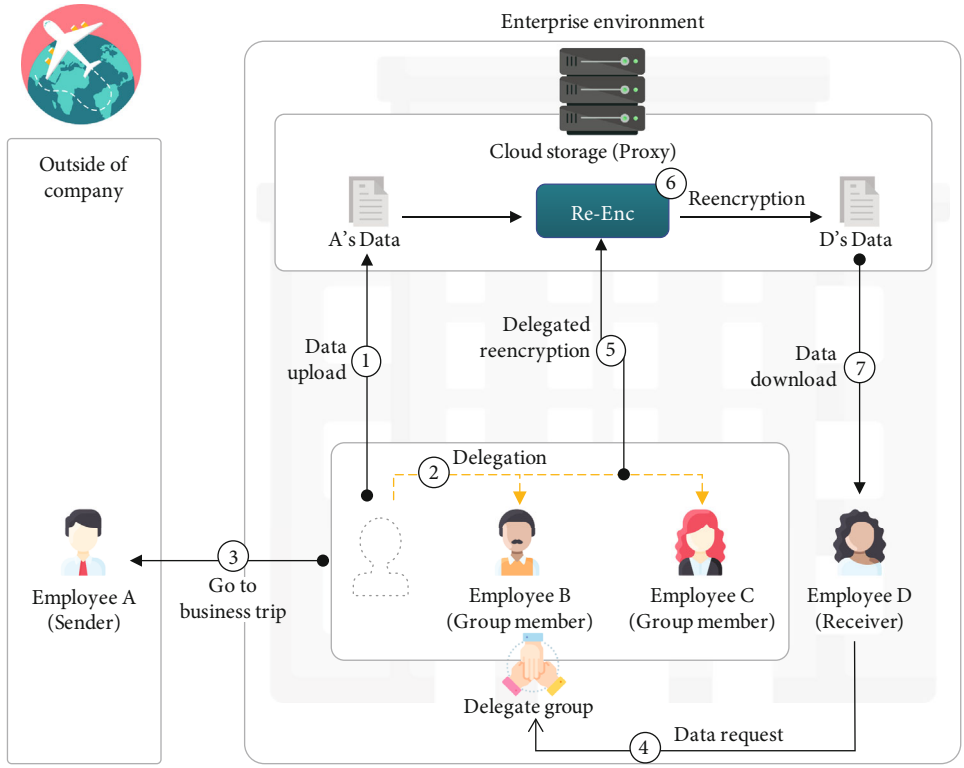

Figure 4: System model of GD-PRE.

owner, another user who has been delegated the owner's authority creates a delegated reencryption key and can share the data with a third party on behalf of the owner. In this process, the user who has been delegated the owner's authority cannot know the owner's private key or the contents of the original data and can only generate the reencryption key for the data designated by the owner. The composition and role of participants in this scheme are described in the next section.

3.1. Role of Participants

3.1.1. KGC (Key Generation Center). It is an administrator that issues and manages the keys to all participants in the system. In order to use this system, a private key must be issued and registered by the KGC, and the parameters disclosed by the KGC must be used.

3.1.2. Proxy. This is a device that stores user data and performs reencryption upon request. Represented by cloud storage, it responds honestly to user requests, but has the characteristic of honest-but-curious, i.e., wants to know the contents of the data.

3.1.3. Users. Users include all senders, receivers, and delegates who use the system provided by the KGC. Any user can act as a sender, receiver, or delegate group member.

3.1.4. Sender. The sender is the owner of the data. Therefore, after encrypting the data using his/her own public key, they upload to the proxy. Thereafter, a reencryption key is generated according to the receiver's request, and the reencrypted encrypted text is provided to the receiver through a proxy. In addition, the sender can delegate the authority to generate the reencryption key to another user in case they are absent, and the delegated user is called the delegate group member.

3.1.5. Delegate Group. The delegate group is a group composed of the sender and delegate group member. The delegate group member is a user who has been delegated the authority to generate reencryption keys from the sender. The delegate group member has been delegated some authorities from the sender, but cannot know the sender's private key or the origin of the message, and can only generate the delegated reencryption key and deliver it to the proxy.

3.1.6. Receiver. The recipient is among the users. Encrypted data can be obtained by requesting data from the sender along with his/her public key. Recipients who have properly obtained data from the sender can decrypt the encrypted data using his/her own private key.

3.2. Algorithms of GD-PRE. This section describes the algorithms included in the proposed technique. The algorithms are of ten types and include setup to encryption, decryption, and reencryption. Detailed formulae and explanations for the algorithms are presented in Section 4.

3.2.1. Setup (λ). Algorithm performed by the KGC, the KGC uses the security parameter λ as an input and executes a probability-reducing algorithm that outputs the parameter params shared with all the users, while the master secret key x remains private.

3.2.2. Partial Private Key Extraction (params, x, ID_i). This is the process in which the KGC receives an input from the master secret key x and the user's identifier $ID_i \in \{0,1\}^*$, outputs the individual partial private key D_i that corresponds to the ID_i, and sends it securely to the user.

3.2.3. Private Key Generation (params, D_i). Algorithm performed by the user, the user generates a complete private key sk_i using the partial private key D_i received from the KGC. The private key is then kept securely.

3.2.4. Public Key Generation (params, ID_i). Algorithm performed by the user, the user creates a public key pk_i using params and his/her ID_i and distributes this public key pk_i.

3.2.5. Encryption (params, $ID_{\mathcal{S}}, pk_{\mathcal{S}}, m$). Algorithm performed by the user, the user computes first-level ciphertext $C_{\mathcal{S}}$ by encrypting message m by the sender inputting message $m \in M$ and $ID_{\mathcal{S}}$.

3.2.6. Reencryption Key Generation (params, $sk_{\mathcal{S}}, pk_{\mathcal{R}}$). Algorithm performed by the sender, the sender generates a reencryption key $RK_{\mathcal{S}\to\mathcal{R}}$ to delegate $C_{\mathcal{R}}$ ciphertext to the receiver. For this purpose, the sender generates a reencryption key $RK_{\mathcal{S}\to\mathcal{R}}$ by using its own private key and public key of the receiver.

3.2.7. Reencryption (params, $RK_{\mathcal{S}\to\mathcal{R}}, C_{\mathcal{S}}$). Algorithm performed by the proxy, the proxy takes the first-level ciphertext C_S and the reencryption key $RK_{\mathcal{S}\to\mathcal{R}}$ as input and outputs a reencrypted ciphertext $C_{\mathcal{R}}$, called the second-level ciphertext.

3.2.8. Group Delegation (params, $ID_{\mathcal{G}_j}$). Algorithm performed by the sender, the sender establishes a delegate group $\mathcal{G}$ who will be delegated the authority to generate his/her reencryption key $RK_{\mathcal{S}\to\mathcal{R}}$ in case he/her are away from the company.

3.2.9. Proxy Delegation (params, $ID_{\mathcal{G}_j}$). Algorithm performed by the sender, the sender adds the value σ'' to his ciphertext uploaded to the proxy so that reencryption can be performed by the delegated group member. Through this, the delegated group member can generate the delegated reencryption key $DRK_{\mathcal{S}\to\mathcal{R}}$ instead of the sender even if they do not know the sender's private key $sk_{\mathcal{S}}$.

3.2.10. Delegated Group Member Setup (params, σ'). Algorithm performed by a delegated group member, each member can obtain v required to generate the delegated reencryption key $DRK_{\mathcal{S}\to\mathcal{R}}$ using the σ' received from the sender.

3.2.11. Delegated Reencryption Key Generation (params, v, $sk_{\mathcal{G}_j}$). Algorithm performed by the delegated group member, the delegated group member generates a delegated reencryption key $DRK_{\mathcal{S}\to\mathcal{R}}$ for the receiver without the sender's private key $sk_{\mathcal{S}}$. $DRK_{\mathcal{S}\to\mathcal{R}}$ enables the creation of $C_{\mathcal{R}}$ by reencrypting the sender's first-level ciphertext $C_{\mathcal{S}}$.

3.2.12. Delegated Reencryption (params, $DRK_{\mathcal{S}\to\mathcal{R}}, C_{\mathcal{S}}$). Algorithm performed by the proxy, the proxy receives the first-level ciphertext C_S and the delegated reencryption key $DRK_{\mathcal{S}\to\mathcal{R}}$ as input and outputs a reencrypted ciphertext $C_{\mathcal{R}}$, called the second-level ciphertext.

3.2.13. Decryption/Redecryption/Delegated Redecryption (params, $C_{\mathcal{R}}, sk_{\mathcal{R}}$). The decryption algorithms are algorithms executed by the owner. When the ciphertext $C_{\mathcal{R}}$ and the receiver $\mathcal{R}$'s private key $sk_{\mathcal{R}}$ are input, the algorithm outputs a message $m \in M$ or an error message.

3.2.14. Trace (params, $DRK_{\mathcal{S}\to\mathcal{R}}, ID_{\mathcal{G}_j}$). Algorithm performed by the sender, the sender can search for the delegated group member who generated the delegated reencryption key on their behalf. For this, the delegated reencryption key $DRK_{\mathcal{S}\to\mathcal{R}}$ generated by the delegated group member and the delegated group member's $ID_{\mathcal{G}_j}$ are used.

3.3. Security Requirements. In order to design a more secure GD-IBPRE, there are seven security requirements in this study, which are as follows.

3.3.1. Confidentiality. In all processes, users without data authority must not be able to know the contents of the data. The owner of the data can provide data decryption authority to the user requesting the data through reencryption.

3.3.2. Integrity. In all processes, such as data transfer and storage, data must remain intact. If the contents of the data are changed, the data owner and the user who has access to the data must know that the data has been changed.

3.3.3. Availability. Legitimate users must be able to access stored data in the proxy. To this end, users who access the data must prove that they are legitimate users and must be able to use the desired data at any time.

3.3.4. Access Control. Users who want to access data can use it by showing that they have permission to use it. To do this, one needs to verify the identity of the user, which is accessed by the proxy (cloud) or makes it available only to legitimate users of the data itself.

3.3.5. Forward Secrecy. It should not be possible to obtain the data owner's private key and personal information using a reencrypted ciphertext or reencrypted key. In addition, it should not be possible to derive a new reencryption key or reencryption statement using the public reencryption key or reencryption statement.

3.3.6. Delegated Reencryption. If the data owner cannot access the data due to absence or other reasons, a delegated group member (delegatee) should be able to reencrypt the data. In this process, the group member must not know the private key of the data owner.

3.3.7. Traceability. The sender delegates the authority to reencrypt his/her data to the delegated group member. However, this privilege can be abused to provide data to unauthorized users. Therefore, the sender must be able to identify the member who abused the authority by identifying

who generated the reencryption key among the members of the delegated group.

4. Proposed Scheme

This section describes the proposed scheme. For this, first, the system parameters are described. Subsequently, detailed equations and explanations for this proposed scheme are presented.

4.1. System Parameters. The parameters and functions used in the proposed scheme are as follows.

$\mathcal{U}_i$: user i ($\mathcal{U}_i \in \mathcal{U}$)

$\mathcal{S}$: sender ($\mathcal{S} \in \mathcal{U}$)

$\mathcal{R}$: receiver ($\mathcal{R} \in \mathcal{U}$)

k: number of delegated group $\mathcal{G}$ members

$\mathcal{G}_j$: delegated group member j ($\mathcal{G}_j \in \mathcal{U}$)

$\mathcal{G}$: delegated group ($\mathcal{G} = \{\mathcal{U}_1, \cdots \mathcal{U}_k\}$)

ID_*: identifier of user

$\mathbb{G}_1, \mathbb{G}_2$: circulation group on prime p

$\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$: bilinear mapping

H_1:hash function $\{0,1\}^* \to \mathbb{G}_1$

H_2:hash function $\mathbb{G}_2 \to \mathbb{G}_1$

$H_3 - H_4$: hash function $\{0,1\}^* \to \mathbb{Z}_q^*$

H_5: hash function $\{0,1\}^* \to \{0,1\}^{l_1+l_2}$

x: master key of KGC

D_i: partial private key of user $\mathcal{U}_i$ created by KGC

sk_i: user $\mathcal{U}_i$'s complete private key

pk_i: user $\mathcal{U}_i$'s public key

$RK_{\mathcal{S}\to\mathcal{R}}$: reencryption key to reencrypt $\mathcal{S}$'s ciphertext to $\mathcal{R}$'s ciphertext

$DRK_{\mathcal{S}\to\mathcal{R}}$: delegated reencryption key created by delegated group member

m: plaintext

C_i: ciphertext of user $\mathcal{U}_i$

$Enc_*(\cdot)$: data encryption algorithm (explained in 4.2.2 Data Storing Phase)

$Dec_*(\cdot)$: data decryption algorithm (explained in 4.2.2 Data Storing Phase)

4.2. Proposed GD-PRE Scheme. The algorithm composition of the proposed scheme and each role described in Section 3.1.3 are shown in Figures 5 and 6. Each algorithm can be classified into a key generation, group delegation, data storing, data sharing, and delegated data sharing phase according to the role and procedure. The detailed procedure is as follows.

4.2.1. Key Generation Phase. First, with the given security parameter λ, the KGC sets the common and secret parameters. Thereafter, each user receives a public key and a private key from the KGC using a common parameter.

(1) Setup. Let $\mathbb{G}_1, \mathbb{G}_2$ be two cyclic groups of prime order q and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map. Set the two λ-bit integers l_1, l_2 and the message space $M \in \{0,1\}^{l_1}$. A random generator $g \in \mathbb{G}_1$ is chosen. The KGC randomly picks an integer $x \in \mathbb{Z}_q^*$ as the master key and publishes params = ($\mathbb{G}_1$, $\mathbb{G}_2$, e, H_1, H_2, H_3, g^x, g, λ, l_1, l_2) .

(2) Partial Private Key Extraction. User $\mathcal{U}_i$ sends his/her identity ID_i to the KGC. The KGC calculates $g_i = H_1(ID_i)$, $D_i = g_i^x$ using ID_i and the master key x and sends user $\mathcal{U}_i$'s partial private key D_i to user $\mathcal{U}_i$. The same operation continues with the other users.

(3) Private Key Generation. User $\mathcal{U}_i$ randomly selects $d_i \in \mathbb{Z}_q^*$ and keeps secret and computes his/her private key sk_i:

$$\mathrm{sk}_i \leftarrow D_i^{d_i} = g_i^{x \cdot d_i}. \tag{1}$$

(4) Public Key Generation. User $\mathcal{U}_i$ computes his/her public key pk_i:

$$\mathrm{pk}_i = \left(g_i, g^{x \cdot d_i}\right). \tag{2}$$

The user creates a public key pk_i using params and his/her ID_i and distributes this public key pk_i. User $\mathcal{U}_i$ publishes pk_i, and anyone can use pk_i to provide ciphertext to user $\mathcal{U}_i$. When the key generation phase is completed, the users who want to form the group perform the initialize of the group delegation phase described in Section 4.2.4.

4.2.2. Data Storing Phase. In this phase, the sender encrypts the data using his/her own public key and stores it in the proxy as shown in Figure 5. After that, the sender can obtain his ciphertext uploaded to the proxy and decrypt it with his/her own private key.

(1) Encryption. Sender $\mathcal{S}$ selects a random integer $r \in \mathbb{Z}_q^*$ and the public key pk_i of the target $\mathcal{U}_i$ and calculates the message $m \in \mathbb{G}_2$ as a ciphertext that can only decrypt U_i as follows:

$$c_1 \leftarrow g^r, c_2 \leftarrow m \cdot e\left(g_i^r, (g^x)^{d_i}\right). \tag{3}$$

As a result, the ciphertext becomes $C_i = (c_1, c_2)$. In this proposed environment, sender $\mathcal{S}$ creates $C_{\mathcal{S}}$ using his public key and uploads $C_{\mathcal{S}}$ to proxy.

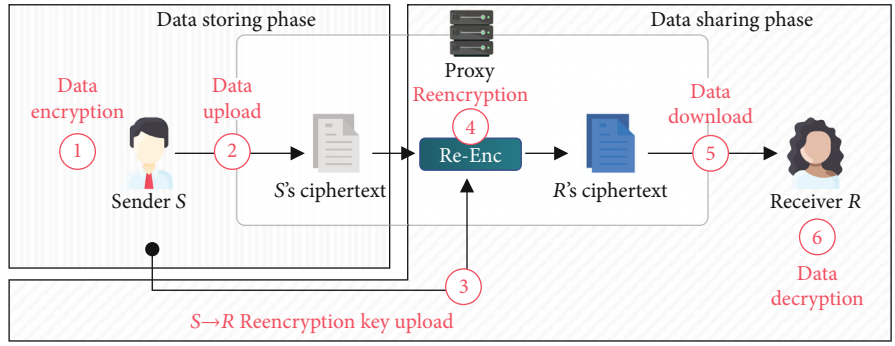

Figure 5: System flow of data storing and sharing phase.

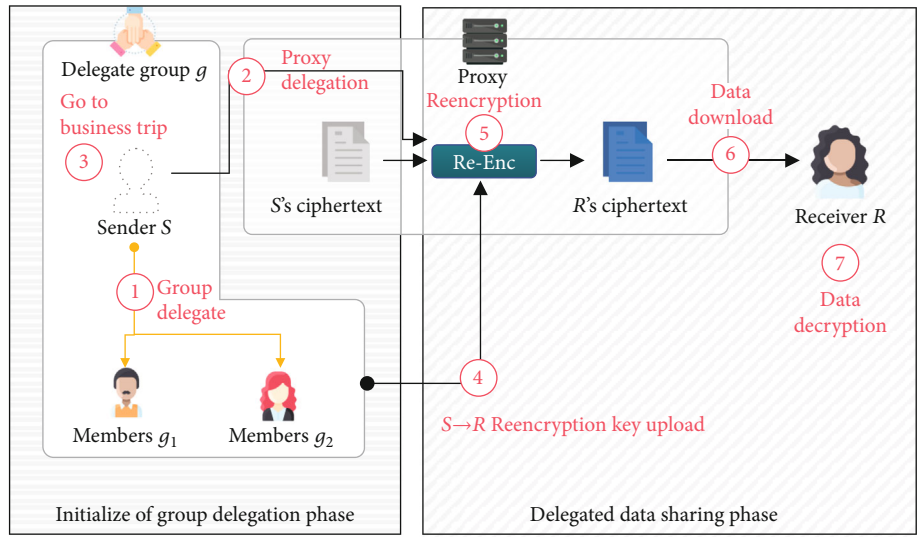

Figure 6: System flow of initialize of the group delegation phase and delegated data sharing phase.

(2) Decryption. User $\mathcal{U}_i$ can download the C_i stored in the proxy and decrypt it using his/her private key sk_i as follows.

$$\begin{aligned} C_i &= (c_1, c_2), \\ m \leftarrow c_2 / e(\mathrm{sk}_i, c_1) &= m \cdot e\left(g_i^r, g^{x \cdot d_i}\right) / e\left(g_i^{x \cdot d_i}, g^r\right) \\ &= m \cdot e(g_i, g)^{r \cdot x \cdot d_i} / e(g_i, g)^{r \cdot x \cdot d_i}. \end{aligned} \tag{4}$$

4.2.3. Data Sharing Phase. This phase is performed when the sender uploads the ciphertext to the proxy and then the receiver requests data. In this phase, the sender who receives the request from the receiver generates a reencryption key using his/her private key and the receiver's public key and delivers it to the proxy. The proxy receiving the reencryption key may obtain the receiver's cipher text by reencrypting the sender's ciphertext with the reencryption key as shown in Figure 5.

(1) Reencryption Key Generation. The sender $\mathcal{S}$ receiving the request of the recipient $\mathcal{R}$ selects random value $\pi \in \mathbb{G}_2$ and $t \in \mathbb{Z}_q^*$. Then, the reencryption key $\mathrm{RK}_{\mathcal{S} \to \mathcal{R}}$ is generated for converting $C_{\mathcal{S}}$ to $C_{\mathcal{R}}$.

$$\begin{aligned} C_{\mathcal{R}}(\pi) &\leftarrow \mathrm{Enc}_{\mathrm{pk}_{\mathcal{R}}}(\pi), \\ \mathrm{RK}_{\mathcal{S} \to \mathcal{R}} &\leftarrow \left(g_{\mathcal{S}}^{-x \cdot d_{\mathcal{S}}} \cdot H_2^t(\pi), C_{\mathcal{R}}(\pi), g^{tr}\right). \end{aligned} \tag{5}$$

$\mathrm{RK}_{\mathcal{S} \to \mathcal{R}}$ is then sent to the proxy by sender $\mathcal{S}$.

(2) Reencryption. The proxy generates the reencrypted ciphertext $C_{\mathcal{R}}$ using the following operation using the ciphertext $C_{\mathcal{S}}$ and the reencryption key $\mathrm{RK}_{\mathcal{S} \to \mathcal{R}}$ of sender $\mathcal{S}$.

$$\begin{aligned} c_1 &\leftarrow g^{tr}, \\ c_2' &\leftarrow c_2 / e\left(g_{\mathcal{S}}^{-x \cdot d_{\mathcal{S}}} \cdot H_2^t(\pi), c_1\right) = m \cdot e\left(H_2^t(\pi), g^r\right), \\ c_3 &\leftarrow C_{\mathcal{R}}(\pi) = \mathrm{Enc}_{\mathrm{pk}_{\mathcal{R}}}(\pi). \end{aligned} \tag{6}$$

$C_{\mathcal{R}} = (c_1, c_2', c_3)$ is then sent to receiver $\mathcal{R}$ by proxy.

If sender $\mathcal{S}$ cannot access the proxy, any member of group $\mathcal{G}$ can perform the 4.2.5 delegated data sharing phase on behalf of the sender $\mathcal{S}$.

(3) Redecryption. Receiver R decrypts a ciphertext $C_{\mathcal{R}}$ received from a proxy by using his/her private key $\mathrm{sk}_{\mathcal{R}}$ to obtain $w = \mathrm{Dec}_{\mathrm{sk}_{\mathcal{R}}}(c_4)$. Finally, receiver $\mathcal{R}$ can compute the plaintext m.

$$\begin{aligned} C_{\mathcal{R}} &\leftarrow \left(c_1, c_2', c_3\right), \\ m &\leftarrow c_2'/e(H_2(\pi), c_1) = m \cdot \frac{e\left(H_2^t(\pi), g^r\right)}{e(H_2(\pi), g^{tr})}. \end{aligned} \tag{7}$$

4.2.4. Initialize of the Group Delegation Phase. In this phase, the sender generates and transmits a value to the proxy and the delegates who will generate the reencryption key on their behalf, as shown in Figure 6.

(1) Group Delegation. Sender $\mathcal{S}$ sets up the delegate group $\mathcal{G} = \{\mathcal{G}_1, \cdots, \mathcal{G}_k\}$ $(\mathcal{G}_k \in \mathcal{U}_i)$ to delegate its authority. Sender $\mathcal{S}$ randomly selects $a, \delta \in \mathbb{Z}_q^*$ and $w \in \{0,1\}^{l_2}$ and computes g^a, z.

$$\begin{aligned} v &\leftarrow t \bullet \delta, \\ z &\leftarrow H_3(s, w, g^a). \end{aligned} \tag{8}$$

Then, sender $\mathcal{S}$ computes L_i and y_j for $j = 1, 2, \cdots, k$.

$$\begin{aligned} L_j &\leftarrow e\left(g_j^a, g^{x \bullet d_j}\right). \\ y_j &\leftarrow H_4(z, L_i, g^a). \end{aligned} \tag{9}$$

Sender $\mathcal{S}$ computes $f(x)$ and E and outputs σ.

$$\begin{aligned} f_1(x) &\leftarrow \prod_{j=1}^{k}\left(x - y_j\right) + s = x^n + g_{n-1}' s^{n-1} + \cdots + g_1' x^1 + g_0', \\ E &\leftarrow H_5(s, z, g^a) \oplus v \| w, \\ \sigma' &= \left(E, g^a, z, g_{n-1}', \cdots, g_1', g_0'\right). \end{aligned} \tag{10}$$

Sender $\mathcal{S}$ sends sender $\mathcal{S}$ uploads $\sigma'' = (c_4, g^{tr}, g_{n-1}', \cdots, g_1', g_0')$ to the delegated group members.

(2) Proxy Delegation. Sender $\mathcal{S}$ uploads an additional value σ'' to the proxy to enable delegate reencryption. σ'' is a value that enables the proxy to be reencrypted by the delegated group member.

$$\begin{aligned} T_j &\leftarrow H_4\left(H_3\left(e\left(g_j^a, g^{x \bullet d_j}\right)\right), g_j\right), \\ f_2(x) &\leftarrow \prod_{j=1}^{k}\left(x - T_j\right) + \gamma = x^n + g_{n-1}' s^{n-1} + \cdots + g_1' x^1 + g_0', \\ c_4 &\leftarrow \left(\gamma g_{\mathcal{S}}^{-x \cdot d_{\mathcal{S}} \bullet \delta}, g^{\frac{r}{\delta}}\right), \\ \sigma'' &= \left(c_4, g^{tr}, g_{n-1}', \cdots, g_1', g_0'\right). \end{aligned} \tag{11}$$

Sender $\mathcal{S}$ uploads $\sigma'' = (c_4, g^{tr}, g_{n-1}', \cdots, g_1', g_0')$ to the proxy.

4.2.5. Delegated Data Sharing Phase. When the sender leaves the company due to a business trip, etc., they can add the delegation information to the data, which is to be delegated to the concerned authority. Thereafter, when the sender is absent, the delegated group member can generate a delegation reencryption key using the delegated information on behalf of the sender and provide the reencrypted data to the receiver as shown in Figure 6.

(1) Delegated Group Member Setup. Sender $\mathcal{S}$ sends σ' to all delegate group members. And each group member $\mathcal{G}_j$ constructs $f(x)$ and obtains m using $\sigma' = (E, g^a, z, g_{n-1}, \cdots, g_1, g_0)$ as follows:

$$\begin{aligned} f(x) &= x^n + g_{n-1} s^{n-1} + \cdots + g_1 x^1 + g_0 = \prod_{j=1}^{k}\left(x - y_j\right) + s, \\ L_j &\leftarrow e\left(g_j^{x \bullet d_j}, g^a\right), \\ y_j &\leftarrow H_4(z, L_i, g^a), \\ s &\leftarrow f\left(y_j\right) = \prod_{j=1}^{k}\left(y_j - y_j\right) + s, \\ v \| w &\leftarrow H_5(s, z, g^a) \oplus E. \end{aligned} \tag{12}$$

Each group member $\mathcal{G}_j$ checks z as follows:

$$z \stackrel{?}{=} H_3(s, w, g^a). \tag{13}$$

(2) Delegated Reencryption Key Generation. Group member $\mathcal{G}_j$ generates a delegated reencryption key $\mathrm{DRK}_{\mathcal{S} \to \mathcal{R}}$ using the acquired v.

$$\begin{aligned} c_3 &\leftarrow C_{\mathcal{R}}(\pi) = \mathrm{Enc}_{\mathrm{pk}_{\mathcal{R}}}(\pi), \\ \mathrm{DRK}_{\mathcal{S} \to \mathcal{R}} &= \left(c_3, H_2^v(\pi), H_3\left(e\left(g_j^a, g^{x \bullet d_j}\right)\right)\right). \end{aligned} \tag{14}$$

$\mathrm{DRK}_{\mathcal{S} \to \mathcal{R}}$ is then sent to the proxy by one of group member $\mathcal{G}_j$.

(3) Delegated Reencryption. The proxy generates the reencrypted ciphertext $C_{\mathscr{R}}$ using the following operation using ciphertext $C_{\mathscr{S}}$ and the delegated reencryption key DR $K_{\mathscr{S}\to\mathscr{R}} = (c_3, H_2^v(\pi), H_3(e(g_j^a, g^{x\bullet d_j}))\,)$ of sender $\mathscr{S}$.

$$
\begin{aligned}
&C_{\mathscr{S}} = (c_1, c_2) = \left(g^r, m \cdot e\left(g_{\mathscr{S}}^r, g^{x\bullet d_{\mathscr{S}}}\right)\right),\\
&T_j \leftarrow H_4\left(H_3\left(e\left(g_j^a, g^{x\bullet d_j}\right)\right), H_1\left(ID_j\right)\right),\\
&f_2(x) = x^n + g'_{n-1}s^{n-1} + \cdots + g'_1 x^1 + g'_0 = \prod_{j=1}^{k}(x - T_j) + \gamma,\\
&\gamma \leftarrow f_2(T_j) = \prod_{j=1}^{k}(T_j - T_j) + \gamma,\\
&c_1 \leftarrow g^{tr},\\
&c_2' \leftarrow c_2 \cdot e\left(\frac{\gamma g_{\mathscr{S}}^{-x\cdot d_{\mathscr{S}}\bullet\delta}}{\gamma} \bullet H_2^v(\pi), g^{\frac{r}{\delta}}\right)\\
&\quad = c_2 \cdot e\left(g_{\mathscr{S}}^{-x\cdot d_{\mathscr{S}}\bullet\delta} \bullet H_2^{t\bullet\delta}(\pi), g^{\frac{r}{\delta}}\right) = c_2 \cdot e\left(g_{\mathscr{S}}^{-x\cdot d_{\mathscr{S}}} \bullet H_2^t(\pi), g^r\right)^{\frac{\delta}{\delta}}\\
&\quad = m \cdot e\left(g_{\mathscr{S}}^r, g^{x\bullet d_{\mathscr{S}}}\right) \cdot e\left(g_{\mathscr{S}}^{-x\cdot d_{\mathscr{S}}} \bullet H_2^t(\pi), g^r\right) = m \cdot e\left(H_2^t(\pi), g^r\right),\\
&c_3 = C_{\mathscr{R}}(\pi) = \mathrm{Enc}_{\mathrm{pk}_{\mathscr{R}}}(\pi),\\
&C_{\mathscr{R}} = \left(c_1, c_2', c_3\right).
\end{aligned}
\tag{15}
$$

$C_{\mathscr{R}} = (c_1, c_2', c_3)$ is then sent to receiver $\mathscr{R}$ by the proxy.

(4) Delegated Redecryption. Receiver $\mathscr{R}$ decrypts a ciphertext $C_{\mathscr{R}}$ received from a proxy by using his/her private key $\mathrm{sk}_{\mathscr{R}}$ to obtain ω. Finally, receiver $\mathscr{R}$ can compute the plaintext m.

$$
\begin{aligned}
&C_{\mathscr{R}} = \left(c_1, c_2', c_3\right),\\
&\omega = Dec_{sk_{\mathscr{R}}}(c_3),\\
&m \leftarrow c_3'/e(H_3(\omega), c_1) = m \cdot \frac{e\left(H_3^t(\omega), g^r\right)}{e(H_3(\omega), g^{tr})}.
\end{aligned}
\tag{16}
$$

(5) Trace. If illegal generation of the delegated reencryption key is confirmed, sender $\mathscr{S}$ can browse the information of the group member who generated the delegated reencryption key. To do this, sender $\mathscr{S}$ creates γ using the IDs of all members in the group. In this process, if the ID of the user who created the delegated reencryption key DR $K_{\mathscr{S}\to\mathscr{R}}$ is used, the correct γ is created, and the perpetrator can be identified through this.

$$
\begin{aligned}
&\mathscr{G}_? = \mathrm{DRK}_{\mathscr{S}\to\mathscr{R}} = \left(c_3, H_2^v(\pi), H_3\left(e\left(g_?^a, g^{x\bullet d_?}\right)\right)\right),\\
&\text{find } \gamma \stackrel{?}{=} T_? \leftarrow H_4\left(H_3\left(e\left(g_?^a, g^{x\bullet d_?}\right)\right), H_1\left(ID_j\right)\right) \text{ for } j = 1, \cdots, k.
\end{aligned}
\tag{17}
$$

5. Analysis of Proposed Scheme

This section analyzes the proposed scheme and explains the results. We analyze the achievement of the proposed scheme with respect to the security requirements and then the efficiency.

5.1. Analysis of Security Requirements. This section analyzes whether the proposed scheme meets the security requirements.

5.1.1. Confidentiality. The owner of the data encrypts the data source m with his/her public key pk_i, converts it to C_i, and uploads it before uploading the data to the proxy. In addition, the decryption of the encrypted data must use the private key sk_i corresponding to the public key pk_i used for data encryption. The decryption operation is as follows, and the data source m can be obtained through the decryption operation.

$$
m \stackrel{?}{=} \frac{c_3}{e(sk_i, c_1)} = m \cdot \left(\frac{e\left(g_i, g^{x\cdot d_i}\right)^r}{e\left(g_i^{x\cdot d_i}, g^r\right)}\right) = m \cdot \left(\frac{e(g_i, g)^{x\cdot d_i\cdot r}}{e(g_i, g)^{x\cdot d_i\cdot r}}\right).
\tag{18}
$$

5.1.2. Integrity. The ciphertext C_i stored in the proxy consists of $C_i = (c_1, c_2)$. c_1 is necessary for decrypting c_2, and if c_1 or c_2 is tampered with, the data owner cannot confirm that the source data is correct, and thus the data is tampered with.

5.1.3. Availability. Sender $\mathscr{S}$ can decrypt data using his/her private key $\mathrm{sk}_{\mathscr{S}}$, and receiver $\mathscr{R}$ can decrypt ciphertext $C_{\mathscr{S}}$ with his/her private key $\mathrm{sk}_{\mathscr{R}}$ as follows.

$$
m \leftarrow \frac{c_3}{e(sk_{\mathscr{S}}, c_1)} = m \cdot \left(\frac{e\left(g_{\mathscr{S}}, g^{x\cdot d_{\mathscr{S}}}\right)^r}{e\left(g_{\mathscr{S}}^{x\cdot d_{\mathscr{S}}}, g^r\right)}\right) = m \cdot \left(\frac{e(g_{\mathscr{S}}, g)^{x\cdot d_{\mathscr{S}}\cdot r}}{e(g_{\mathscr{S}}, g)^{x\cdot d_{\mathscr{S}}\cdot r}}\right).
\tag{19}
$$

5.1.4. Access Control. Sender $\mathscr{S}$ stores their generated ciphertext in a proxy. Thereafter, when another user requests the use of the data, the sender $\mathscr{S}$ can generate a reencryption key $\mathrm{RK}_{\mathscr{S}\to\mathscr{R}}$ by combining his/her private key and the receiver $\mathscr{R}$'s public key. The generated reencryption key R $K_{\mathscr{S}\to\mathscr{R}}$ is transmitted to the proxy to generate a ciphertext $C_{\mathscr{R}}$ that the receiver $\mathscr{R}$ can decrypt as follows.

Table 1: Comparison of computational amount of each algorithm.

Encryption time	Reencryption time	Delegated reencryption time	Decryption time
$3T_e + 1T_m + 1T_{bi}$	$2T_e + 2T_m + 1T_{bi} + 1T_h$	$3T_m + 2T_{bi} + 1T_h$	$1T_m + 1T_{bi}$

T_e : time for a exponentiation operation; T_m : time for a multiplication operation; T_{bi} : time for a bilinear pairing operation; T_h : time for a hash function operation; n : number of users.

$$c_3' \leftarrow \frac{m \cdot e\left(g_{\mathcal{S}}, g^{x \cdot d_{\mathcal{S}}}\right)^r}{e\left(g_{\mathcal{S}}^{-x \cdot d_{\mathcal{S}}} \cdot H_2^t(\omega), c_1\right)} = \frac{m \cdot e\left(g_{\mathcal{S}}^{x \cdot d_{\mathcal{S}}}, g^r\right)}{e\left(g_{\mathcal{S}}^{-x \cdot d_{\mathcal{S}}} \cdot H_2^t(\omega), g^r\right)} = m \cdot e\left(g_{\mathcal{S}}^{x \cdot d_S} \cdot g_{\mathcal{S}}^{-x \cdot d_{\mathcal{S}}} \cdot H_2^t(\omega), g^r\right) = m \cdot e\left(H_2^t(\omega), g^r\right). \tag{20}$$

Thereafter, the receiver who has received the reencrypted data $C_{\mathcal{R}}$ can obtain the data source by decrypting the corresponding ciphertext with his/her private key $\mathrm{sk}_{\mathcal{R}}$.

$$m \leftarrow \frac{c_2'}{e(H_3(\omega), c_2)} = m \cdot \left(\frac{e\left(H_3^t(\omega), g^r\right)}{e(H_3(\omega), g^{tr})}\right) = m \cdot \left(\frac{e(H_3(\omega), g^{tr})}{e(H_3(\omega), g^{tr})}\right). \tag{21}$$

5.1.5. Forward Secrecy. Sender $\mathcal{S}$ uses his/her private key $\mathrm{sk}_{\mathcal{S}}$ and the receiver $\mathcal{R}$'s public key $\mathrm{pk}_{\mathcal{R}}$ to generate a reencryption key $\mathrm{RK}_{\mathcal{S} \to \mathcal{R}}$. In addition, the sender $\mathcal{S}$ creates the reencryption key $\mathrm{RK}_{\mathcal{S} \to \mathcal{R}}$ using the one-way function $H_2(\cdot)$ and the difficulty of discrete algebra (DLP) such that he/she cannot extract his/her private key $\mathrm{sk}_{\mathcal{S}}$ from the reencryption key.

5.1.6. Delegated Reencryption. If sender $\mathcal{S}$ is unable to access the data, the other members of the preconfigured group are delegated the authority to reencrypt the encrypted data without the sender $\mathcal{S}$'s private key $\mathrm{sk}_{\mathcal{S}}$. For this, the sender $\mathcal{S}$ performs a separate operation as follows so that the delegated group member $\mathcal{G}_j$ can generate the delegated reencryption key $\mathrm{DRK}_{\mathcal{S} \to \mathcal{R}}$.

5.2. Efficiency Analysis of the Proposed GD-PRE Scheme. Table 1 shows the characteristics of the proposed scheme. This proposed scheme was designed by applying an encryption scheme using the existing multireceiver encryption [28–31]. Here, the delegate group includes the sender and the delegate group member, and users included in the delegate group can generate the delegate reencryption key. In this proposed scheme, there is no separate group formation process, and the sender designates a user to delegate his/her authority and transmits the value in one direction. Therefore, even if the number of members of the delegation group increases, the number of communications does not increase. As a result, it is possible to share data through a member of a delegated group when the sender is absent, without showing significant difference in terms of general proxy reencryption and computational requirements.

6. Conclusions

Proxy reencryption is a technology that converts (reencrypts) data encrypted with a public key such that other users can decrypt it with their private key. Therefore, it is not necessary to decrypt the encrypted data for data sharing; hence, it has the efficiency of operation and communication. In addition, when proxy reencryption is used, data is encrypted and uploaded to the cloud storage, such that it can later be easily shared through the generation of a reencryption key at the request of another user. By utilizing these features, corporate cloud storage can efficiently manage data such as collaboration and business data delivery.

However, in a corporate environment, employees are not always resident in the company, but often leave the workplace on business trips, vacations, etc. When the data owner is away and receives a request to share data from another employee, the data owner must return to the company to provide the data. If the data owner urgently needs important business data while on a long-term overseas business trip, they have no alternative other than providing his/her own private key. However, such private key sharing is a serious security breach and can pose considerable risk to the company and the individual themselves. This study was conducted to solve this problem.

In this study, we propose a scheme that allows a preformed group to perform reencryption key generation in the event of an emergency by applying existing IBPRE techniques. The proposed scheme is designed for situations in which general IBPRE techniques cannot be used by owners of data in data self-sovereign environments, such as enterprise environments. In such environments, as the individual directly controls his/her own data, the sovereignty of his/her data can be guaranteed. However, if an individual cannot exercise sovereignty, such as when they are rendered unconscious in an emergency, data that are necessary for handling the emergency cannot be accessed. Therefore, in the proposed scheme, users form groups with other trusted individuals in advance. Therefore, in the event of an emergency, group members can control data by reencrypting the data of other users on their behalf, which can solve the limitations of self-sovereign data in existing enterprise environments.

Data Availability

No data were used.

Conflicts of Interest

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the MSIT ((Ministry of Science, ICT), Korea, under the High-Potential Individuals Global Training Program) (2020-0-01596) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation) and the Soonchunhyang University Research Fund and the BK21 FOUR (Fostering Outstanding Universities for Research) (No. 5199990914048).

References

[1] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology — CRYPTO 2001. CRYPTO 2001*, J. Kilian, Ed., vol. 2139 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2001.

[2] C. Cocks, "An identity based encryption scheme based on quadratic residues," in *Cryptography and Coding. Cryptography and Coding 2001*, B. Honary, Ed., vol. 2260 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2001.

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

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

[5] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan, "Generative adversarial networks: a survey towards private and secure applications," *ACM Computing Surveys*, vol. 37, no. 4, pp. 1–37, 2021.

[6] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Advances in Cryptology — EUROCRYPT'98. EUROCRYPT 1998*, K. Nyberg, Ed., vol. 1403 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1998.

[7] A.-A. Ivan and Y. Dodis, *Proxy cryptography re-visited*, NDSS, 2003.

[8] T. H. Yuen, Y. Zhang, S. M. Yiu, and J. K. Liu, "Identity-based encryption with post-challengeauxiliary inputs for secure cloud applications and sensor networks," in *European Symposium on Research in Computer Security*, Springer, Cham, 2014.

[9] Y.-P. Chiu, C.-L. Lei, and C.-Y. Huang, "Secure multicast using proxy encryption," in *Information and Communications Security. ICICS 2005*, S. Qing, W. Mao, J. López, and G. Wang, Eds., vol. 3783 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2005.

[10] J. Shao, P. Liu, G. Wei, and Y. Ling, "Anonymous proxy re-encryption," *Security andCommunication Networks*, vol. 5, no. 5, pp. 439–449, 2012.

[11] 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.

[12] C.-K. Chu and W.-G. Tzeng, "Identity-based proxy re-encryption without random oracles," in *Information Security. ISC 2007*, J. A. Garay, A. K. Lenstra, M. Mambo, and R. Peralta, Eds., vol. 4779 of Lecture Notes in Computer Science, pp. 189–202, Springer, Berlin, Heidelberg, 2007.

[13] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Applied Cryptography and Network Security. ACNS 2007*, J. Katz and M. Yung, Eds., vol. 4521 of Lecture Notes in Computer Science, pp. 288–306, Springer, Berlin, Heidelberg, 2007.

[14] 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 *Computer Security - ESORICS 2014. ESORICS 2014*, M. Kutyłowski and J. Vaidya, Eds., vol. 8712 of Lecture Notes in Computer Science, pp. 257–272, Springer, Cham, 2014.

[15] X. Zhao, D. Wei, and H. Wang, "Asymmetric group key agreement with traitor traceability," in *Asymmetric Group Key Agreement with Traitor Traceability*, Nanning, China, 2010.

[16] A. Paul, V. Srinivasavaradhan, S. S. D. Selvi, and C. P. Rangan, "A CCA-secure collusion-resistant identity-based proxy re-encryption scheme," in *Provable Security. ProvSec 2018*, J. Baek, W. Susilo, and J. Kim, Eds., vol. 11192 of Lecture Notes in Computer Science, Springer, Cham, 2018.

[17] L. Wang, L. Wang, M. Mambo, and E. Okamoto, "New identity-based proxy re-encryption schemes to prevent collusion attacks," in *Pairing-Based Cryptography - Pairing 2010. Pairing 2010*, L. Wang, M. Mambo, and E. Okamoto, Eds., vol. 6487 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2010.

[18] L. Xu, X. Wu, and X. Zhang, "CL-PRE: a certificate-less proxy re-encryption scheme for secure data sharing with public cloud," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security - ASIACCS '12*, Seoul Korea, 2012.

[19] X. Lv, H. Li, and B. Wang, "Authenticated asymmetric group key agreement based on certificateless cryptosystem," *International Journal of Computer Mathematics*, vol. 91, no. 3, pp. 447–460, 2014.

[20] Y. Dodis, S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan, "Public-key encryption schemes with auxiliary inputs," in *Theory of Cryptography. TCC 2010*, D. Micciancio, Ed., vol. 5978 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2010.

[21] Y. Polyakov, K. Rohloff, G. Sahu, and V. Vaikuntanathan, "Fast proxy re-encryption for publish/subscribe systems," *ACM Transactions on Privacy and Security*, vol. 20, no. 4, pp. 1–31, 2017.

[22] M. Mambo and E. Okamoto, "Proxy cryptosystems: delegation of the power to decrypt ciphertexts," *IEICE transactions onfundamentals of electronics, Communications and computer sciences*, vol. 80, no. 1, pp. 54–63, 1997.

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

[24] K. Li, G. Luo, Y. Yang, W. Li, S. Ji, and Z. Cai, "Adversarial privacy-preserving graph embedding against inference attack," *IEEE Internet of Things*, vol. 8, pp. 6904–6915, 2021.

[25] S. S. M. Chow, J. Weng, Y. Yang, and R. H. Deng, "Efficient unidirectional proxy re-encryption," in *Progress in Cryptology – AFRICACRYPT 2010. AFRICACRYPT 2010*, D. J. Bernstein and T. Lange, Eds., vol. 6055 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2010.

[26] J. J. Qiu, J. B. Jo, and H. J. Lee, "Collusion-resistant identity-based proxy re-encryption without random oracles," *International Journal of Security and Its Applications*, vol. 9, no. 9, pp. 337–344, 2015.

[27] W.-B. Kim, I. Y. Lee, and K. B. Yim, "Group delegated ID-based proxy re-encryption for PHR," in *Innovative Mobile and Internet Services in Ubiquitous Computing. IMIS 2020*, L. Barolli, A. Poniszewska-Maranda, and H. Park, Eds., vol. 1195 of Advances in Intelligent Systems and Computing, pp. 447–456, Springer, Cham, 2020.

[28] L. Deng, "Anonymous certificateless multi-receiver encryption scheme for smart community management systems," *Soft Computing*, vol. 24, no. 1, pp. 281–292, 2020.

[29] D. He, H. Wang, L. Wang, J. Shen, and X. Yang, "Efficient certificateless anonymous multi-receiver encryption scheme for mobile devices," *Soft Computing*, vol. 21, no. 22, pp. 6801–6810, 2017.

[30] J. Baek, R. Safavi-Naini, and W. Susilo, "Efficient multi-receiver identity-based encryption and its application to broadcast encryption," in *Public Key Cryptography - PKC 2005. PKC 2005*, S. Vaudenay, Ed., vol. 3386 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2005.

[31] J. Zhang and J. Mao, "Anonymous multi-receiver broadcast encryption scheme with strong security," *International Journal of Embedded Systems*, vol. 9, no. 2, pp. 177–187, 2017.