Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 1903197, 17 pages
https://doi.org/10.1155/2022/1903197

Research Article

Certificateless Group to Many Broadcast Proxy Reencryptions for Data Sharing towards Multiple Parties in IoTs

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 31 March 2022; Accepted 11 June 2022; Published 29 June 2022

Academic Editor: Yan Huo

Proxy reencryption delegates encrypted data stored in a proxy to a third party. This proxy reencryption takes the form of one sender providing data to one receiver. However, this method incurs a significant overhead for both the sender and proxy as the number of users receiving the same data increases. In addition, in a large-scale environment, such as an Internet of Things or big data environment, a scenario where several workers jointly create and own an output may exist. In such an environment, ownership disputes can arise when only one operator owns a piece data used by other operators. In this study, to solve this problem, we propose a technique in which multiple users can jointly own one piece of data, and multiple recipients can receive the same data through proxy reencryption.

1. Introduction

The development of information technology has brought about numerous changes to data storage and utilization technology. The Internet, which is the most widely used network, has made it possible to transmit and use data anytime and anywhere without restrictions in time and place. Internet technologies have been developed to achieve improved speeds, allowing more data to be transmitted concurrently. In addition, the Internet can be used in a wireless form. Storage media that allow more data to be stored and used in a unit area have also been developed. Because more data can be stored in a smaller space, removable storage devices have emerged, and removable storage media have provided an environment in which data can be held and utilized more efficiently. The development of such network technologies and storage media has recently achieved a rapid growth and has taken on various forms, reaching the stage of virtual storage spaces such as cloud computing. We believe that this change in the environment is a transition from an environment using a storage medium to an environment using a storage space, and that the change in such an environment is accelerating.

Gartner, an American information technology research and advisory firm, publishes the Top Strategic Technology Trends and Hype Cycles [1]. Cloud computing is an important strategic technology to the extent that it is selected by this publication every year. However, despite the growing awareness and importance of cloud computing, many companies and institutions are hesitant to adopt it for security reasons. Because cloud computing technology is always connected to a network, it is continuously exposed to data leakage and multiple foes using the network. Therefore, security technology is essential when introducing cloud computing. The secure storage and transmission of data are essential for a secure cloud computing environment. In addition, cloud storage, a subclass of cloud computing technology, stores data and must provide availability for future use. Therefore, cloud computing must consider more security factors than portable storage media.

Cloud storage is a representative technology for storing data using cloud computing technology. As described above, cloud storage can be used as storage space by utilizing network technology, and in this way, the digital data can be stored and used without a physical storage medium. Using the advantages of cloud storage, one can not only store and use one's own data, such data, and also be shared with other users. Data sharing in this manner increases the efficiency because data can be passed through the cloud storage without being passed directly between the data owner and recipient. In addition, even when sharing the same data with multiple recipients, it achieves the advantage of being able to transmit data from cloud storage without the need for the owner to transmit the data each time the data are accessed. However, as described above, the cloud computing technology used over a network is continuously exposed to data leakage and security threats. Therefore, the security factor must be considered in data-sharing methods using cloud storage.

To securely share data using cloud storage, protection of both the data and transmission process must be considered. In general, a cloud storage server is a remote server managed by a data owner and other administrators. Such a server has an honest-but-curious characteristic, which processes the user's request accurately but always incurs the possibility of exposing the data. Therefore, if an owner's sensitive data are stored in cloud storage, there is a possibility that the content of the data will be exposed. Data encryption must be applied to solve this problem. Data encryption technology refers to a technology in which only a user who possesses a decryption key corresponding to the encryption key of the data can view the content of the encrypted data. Therefore, only a user who has a decryption key corresponding to the encryption key used for the data uploaded by the owner can view the content of the data. Two encryption algorithms may be primarily used for this encryption method, and a total of four encryption methods may be used by combining the two encryption algorithms. However, these four encryption methods cannot be applied to data-sharing methods using cloud storage because each of them has certain problems such as a key distribution, computational inefficiency, and exposure to the data source. To solve this, a proxy reencryption technique has been proposed.

Proxy reencryption technology securely shares data using a proxy server, as proposed by Blaze et al. in 1998 [2]. Proxy reencryption technology refers to a technology that stores data encrypted with the owner's encryption key in the proxy and then converts the encrypted data into a specified number of cipher texts. During this process, because the proxy does not decrypt the encrypted data, the contents of the data cannot be known, and the receiver can decrypt the data using its own private key. Therefore, the data are not exposed during the process of data storage and delivery. With this proxy reencryption technology, the proxy may be represented by cloud storage, and if such technology is used, data can be shared securely and efficiently in the cloud storage environment.

As large-scale network environments such as IoT, secure e-mail, and connected cars become more common, cases of data sharing between multiple users are increasing [3–5]. In such an environment, data sharing using cloud storage can be an effective way to deliver data securely and efficiently to multiple users. However, because general proxy reencryption technology uses a 1 : 1 data transmission method, it cannot support multiple data owners or multiple data receivers. In this case, to provide the same data to multiple recipients, it is necessary to generate a reencryption key and conduct as many reencryption operations as the number of recipients. In addition, even when multiple workers collaborate to create a single data point, only one worker can be the owner. In this case, because the data cannot be efficiently owned or shared in a large-scale data ownership and reception environment, an appropriate method that considers these issues is required. This study was conducted to provide a method that considers multiple owners and recipients simultaneously. Thus, it provides a method for flexibly and efficiently carrying out the ownership and sharing of data using proxy reencryption technology.

2. Related Works

This section describes related studies for a proper understanding of this study.

2.1. Secure Data Sharing. As a basic concept of data-sharing technology, data owners give permission for their data to be available to other users. In existing systems, such as Linux or Windows, ownership of data is provided in the same form as RWX, and the meanings of readable, writable, and executable are the same. This indicates that data ownership is further subdivided and provided as a logical form of usage rights. By contrast, from a cryptographic perspective, data ownership can be accessed in the form of determining whether data can be decrypted. That is, if one has a decryption key corresponding to a key having encrypted data, it can be determined that one has ownership of the data because the data source can be obtained through decryption. Therefore, the method of sharing data through such a cryptographic concept can be accessed by delegating the decryption authority of the encrypted data [6].

A method of providing the decryption rights of encrypted data to another user can be approached in four major ways using a symmetric key encryption algorithm, and a public key encryption algorithm is shown in Figure 1

(1) *Use of only symmetric key encryption:* with this method, the data that the sender uploads to the proxy are first encrypted with the sender's own symmetric key and uploaded. When the receiver requests data, the proxy delivers a ciphertext of the sender to the receiver, and the sender must deliver its symmetric key to the receiver. When this method is applied, both the sender and receiver can conduct encryption/decryption using a symmetric key. However, this process requires a symmetric key distribution process. Symmetric key eavesdropping by an attacker may occur during the process of symmetric key distribution. In addition, because the symmetric key delivered to the recipient cannot be delivered to

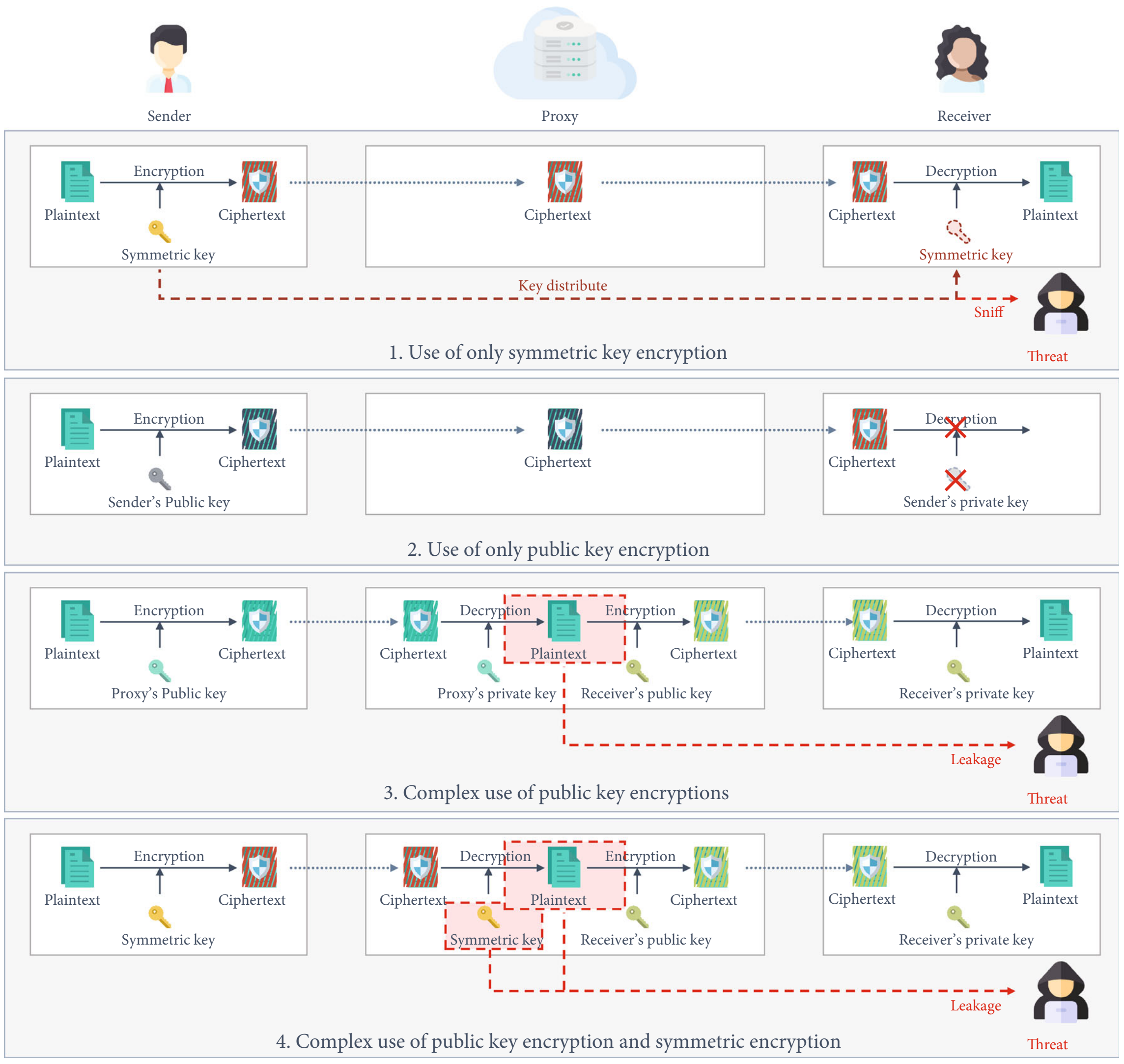

FIGURE 1: Problems of data sharing method using encryption.

another recipient, reusing the ciphertext uploaded to the proxy becomes impossible. Therefore, the data sharing method using symmetric key encryption is unsuitable in terms of security and efficiency

(2) *Use of only public key encryption:* with this method, the data that the sender uploads to the proxy are first encrypted with the sender's public key and then uploaded. When the receiver requests data, the proxy delivers the sender's ciphertext to the receiver. However, because this method can only be decrypted using the sender's private key, the sender must deliver his or her private key to the receiver. However, in this case, the sender's private key is exposed by other users, which can lead to serious security problems. Consequently, the receiver cannot decrypt the ciphertext of the sender without lowering the level of security

(3) *Complex use of public key encryptions:* with this method, the data uploaded by the sender to the proxy are first encrypted and uploaded with a symmetric key shared between the sender and the proxy. Upon receiving the data, the proxy decrypts the ciphertext of the sender using a symmetric key to obtain the original data. After that, just like the *2. Use of only public key encryption* method, the data source is encrypted with the recipient's public key and delivered to the recipient, who can decrypt it. As with the method that uses public key encryption multiple times, the data source is encrypted with the recipient's public key and delivered to the recipient, and the recipient can decrypt it. In this method, even if there are many recipients, the proxy can directly perform encryption with the public key of each recipient, so that the computational burden on the sender is not increased. As with the method of using public key encryption multiple times, even if the number of recipients increases, the computational burden on the sender does not increase because the proxy can conduct encryption directly

using the public key of each recipient. However, this process allows the proxy to know the list of recipients, exposing the contents of the data source to threats both inside and outside the proxy. Therefore, the method of using public key encryption and symmetric key encryption together has the efficiency of data sharing but without guaranteeing security

(4) *Complex use of public key encryption and symmetric encryption:* with this method, the data that the sender uploads to the proxy are first encrypted with the sender's public key and then uploaded. When the receiver requests data, the proxy delivers the sender's ciphertext to the receiver. However, because this method can only be decrypted using the sender's private key, the sender must deliver his or her private key to the receiver. However, in this case, the sender's private key is exposed by other users, which can lead to serious security problems. Consequently, the receiver cannot decrypt the ciphertext of the sender without lowering the level of security

As described above, use of the symmetric and public key encryption methods to securely share data through cloud storage does not provide sufficient security. Therefore, a method that can provide both security and efficiency throughout the data sharing process is required. Various studies have been conducted to satisfy such requirements, and proxy reencryption technology has been proposed.

2.2. Proxy Reencryption. In 1998, Blaze et al. proposed proxy reencryption (PRE) [2], which is a technology that transforms data through a proxy and delivers them securely to the receiver. This technology converts data encrypted using the sender's public key into data encrypted using the receiver's public key at a proxy. Through this process, the private keys of the sender and receiver, as well as the original data, are not exposed because data decryption is not applied. Using proxy reencryption, data can be securely stored in cloud storage and shared efficiently by converting the data into the recipient's ciphertext at the request of the recipient. The basic form of such a proxy reencryption is shown in Figure 2, and research on various sharing methods using proxy reencryption technology is currently underway.

Proxy reencryption comprises five steps: encryption, reencryption key generation, reencryption, decryption, and redecryption. The details of each step are as follows:

(i) *Encryption:* in this step, the data owner encrypts the data and uploads them to a proxy. To this end, the data owner encrypts the data using his or her own encryption key, such that the source of the data cannot be known. The encrypted data are then delivered to the proxy through the public network and stored. In this case, the proxy cannot know the contents of the data stored in the proxy, and even if the encrypted data are exposed or leaked, decryption corresponding to the encryption key is applied, and a user without a key cannot know its contents

(ii) *Reencryption key generation:* in this step, the data owner provides the receiver with the authority to decrypt his or her data stored in the proxy. For this, the data owner first receives the information of the recipient who requested the data. The data owner then creates a reencryption key by combining the information of the recipient with his or her own decryption key and secret information. The data owner can control the reencryption by passing the generated reencryption key to the proxy. In this case, the proxy and attacker should not be able to obtain the secret information of the data owner through the reencryption key

(iii) *Reencryption:* this step refers to the process of converting the encrypted data of the data owner into receiver data. To this end, the proxy applies a reencryption algorithm using the cipher text and reencryption key received from the data owner, and as a result, can obtain a reencrypted cipher text. In this case, the reencrypted cipher text is the cipher text in which the decryption authority is delegated from the data owner to the receiver, and the proxy cannot know the contents of the data during the reencryption process. The reencrypted ciphertext is then sent to the receiver

(iv) *Decryption:* in this step, the data owner decrypts the ciphertext. This step is conducted to obtain the data source by downloading the ciphertext uploaded by the data owner to the proxy during the encryption step again by the data owner. Accordingly, the data owner represents the data decryption process using a decryption key that corresponds to the encryption key used for data encryption. This process represents a typical encryption-decryption relationship and shows that data owners can reuse their data at will

(v) *Redecryption:* in this step, the receiver decrypts the reencrypted ciphertext. To this end, the receiver receives the reencrypted cipher text from the proxy and performs a process of decrypting the received cipher text using its decryption key. At this time, if the recipient is not the correct recipient, the data cannot be decrypted even if the reencrypted cipher text is received

Most proxy reencryption structures are as above, and various methods can be used to configure the above steps. Currently, most proxy reencryption studies use public-key encryption methods [7–16]. Because PKC performs encryption using a public key, it offers excellent accessibility and usability. However, additional computations and certificate management problems occur because procedures such as the generation of a certificate for the public key are essential. To solve this problem, identity-based PKC (IB-PKC) using a key issuance method through a key generation center (KGC) has been proposed [17]. Since IB-PKC was first proposed,

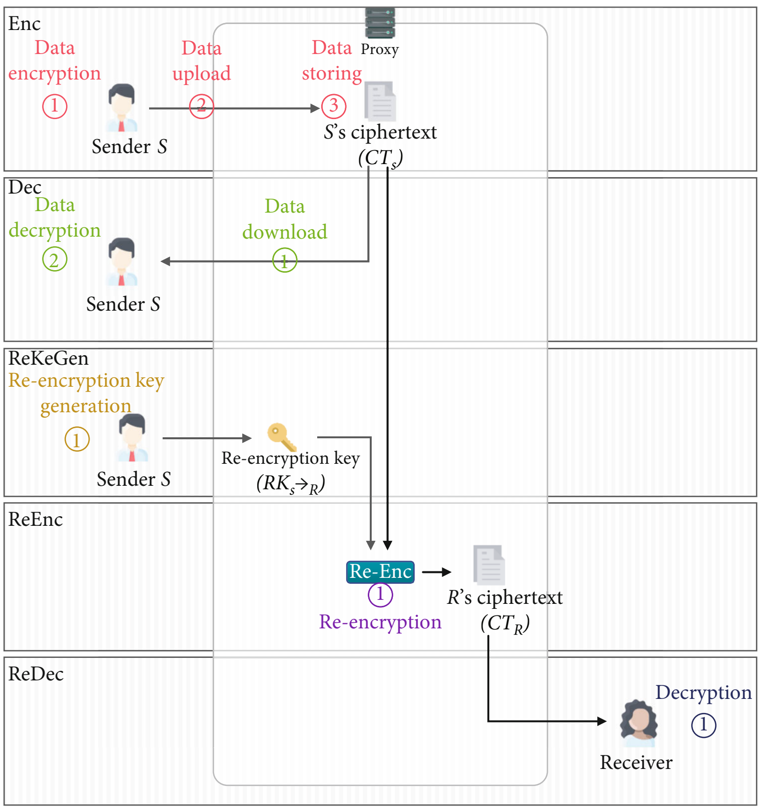

Figure 2: Basic proxy reencryption method.

various proxy reencryption studies based on IB-PKC have been conducted [7, 18–22]. However, in IB-PKC, because KGC directly issues the user's key, the problem of a key escrow by the KGC arises. To solve this problem, CL-PKC, a method in which a complete key is not generated by the KGC without the use of a certificate, has been proposed [23]. CL-PKC follows a method in which KGC issues only a partial secret key to each user, and the users then combine their secret information to complete a private key. Therefore, the key escrow problem of KGC does not occur. Accordingly, studies on certificateless proxy reencryption (CL-PRE) have recently been conducted using CL-PKC [24–27].

2.3. Multireceiver Encryption. Multireceiver encryption (MRE) is a technology that grants the same data decryption authority to multiple recipients with only a single encryption. MRE has been utilized in various studies based on PKC as shown in Figure 3 [28–36]. However, the existing MRE method has the problem of receiver identification. This is because the recipient can be identified by extracting the recipient information included in the ciphertext. To solve this problem, a method for specifying the receiver using a polynomial has been proposed [37]. Using this method, the receiver's information cannot be extracted by combining it with a polynomial. However, other studies have demonstrated that this scheme can obtain the recipient's identity [38, 39]. Fan et al. proposed an improved version of this scheme [40]. In addition, Zhang and Takagi proposed a method in which both the sender and receiver are anonymous [41]. However, Zhang and Mao found that this scheme does not provide complete anonymity; therefore, they proposed a new type of identity-based MRE (IB-MRE) [42]. However, after the key escrow problem of IB-PKC was presented, a study was conducted on applying CL-PKC to MRE.

Based on research conducted on CL-MRE, Sur et al. improved the implicit certificate-based MRE proposed in 2007 [43] and proposed CL-MRE in 2011 [44]. In addition, Islam et al. proposed a CL-MRE, which achieved confidentiality and anonymity in a random oracle model [45]. However, Hung et al. pointed out a large number of computations, similar to that indicted by Islam, which takes a lengthy computation time [46]. However, Hung et al. also had a problem in that the map-to-point (MTP) hash operation, which requires a lengthy operation time, increases linearly in proportion to the number of users. He et al. [47] proposed a method that does not use a map-to-point (MTP) hash to solve this problem. Although Deng et al. [48] and Zhu et al. [49] proposed CL-MRE to solve the key escrow problem, a considerable computational load was incurred using bilinear pairing, and the scheme developed by Zhu et al. did not provide additional receiver anonymity. Although Win et al. [50] did not use bilinear pairing, they also did not provide receiver anonymity or decryption fairness.

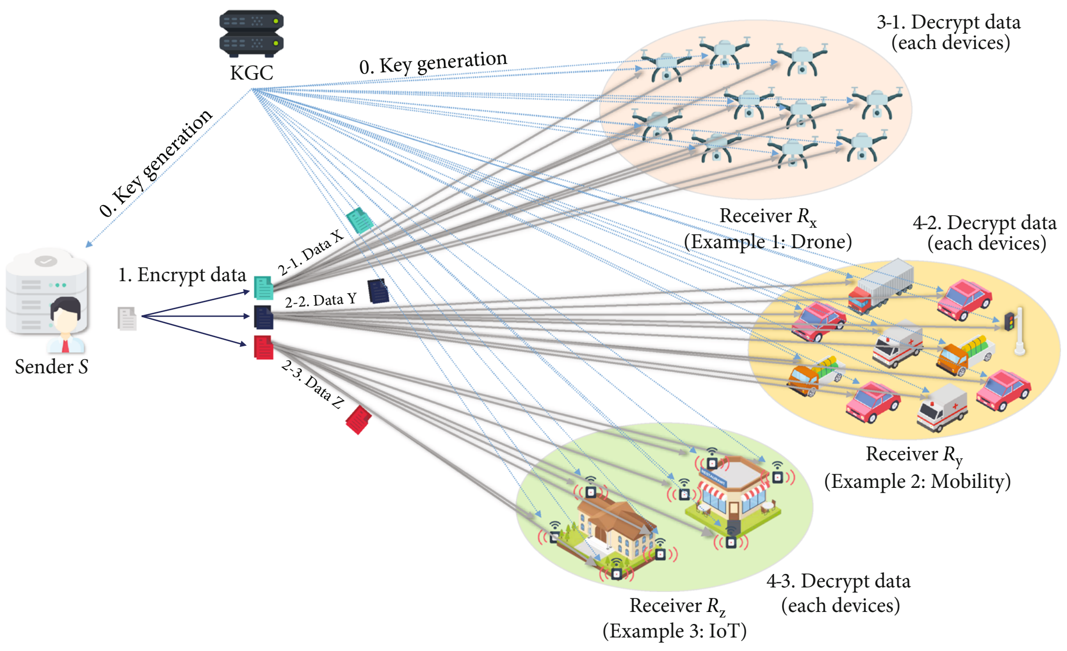
KGC
0. Key generation
0. Key generation
3-1. Decrypt data (each devices)
Sender S
1. Encrypt data
2-1. Data X
2-2. Data Y
2-3. Data Z
Receiver R_x (Example 1: Drone)
4-2. Decrypt data (each devices)
Receiver R_y (Example 2: Mobility)
Receiver R_z (Example 3: IoT)
4-3. Decrypt data (each devices)

Figure 3: Flow of multireceiver encryption.

3. Preliminaries

This section describes the basic environment and settings for understanding the scheme proposed in this study.

3.1. System Model. This section describes the system model used in the present study. The participants in this system model are divided into KGC, proxy, user, owner, and receiver, and the description of each participant is as follows.

(i) *Key generation center* (*KGC*): with this model, KGC plays a role in managing the system administrator or users in the system. KGC manages all users in the system and registers and manages users through preset settings. In addition, common parameters are created and disclosed such that all participants can conduct the operations of a predetermined algorithm. Using these parameters, all participants can generate their own keys or conduct such predetermined algorithm operations. At this time, to avoid the key escrow problem caused by the KGC, the KGC cannot know the user's complete key

(ii) Proxy: with this model, a proxy indicates a remote server that can store and distribute data between users. The most representative form of a proxy is cloud storage, which can store, transmit, and calculate data according to the user's request. With this model, because the proxy is considered a semitrusted environment, there is a possibility that the contents of the unencrypted data may be exposed or leaked

(iii) *User:* using this model, a user means all users including the owner and receiver. Each user has his/her own public and private keys and can encrypt and decrypt data using these keys

(iv) *Owner group*: with this model, the owner means the group of users who own the data. It is assumed that ownership of one piece of data is shared by several users. Examples of such environments include operations, organizations, and the military. Under this environment, because each user has equal ownership, decryption and reencryption keys can be generated using the threshold method to prevent abuse of authority by one owner

(v) *Receiver*: with this model, the receiver means all receivers who receive the data decryption right from the owner. These recipients may consist of one or more individuals, and multiple recipients who have been granted the same data rights have the same rights. In addition, each authorized recipient can decrypt the data using their own private keys

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 to the authorized user. To do this, the data must be encrypted using the encryption key, and a user who does not have a legitimate 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 process of being delivered to the cloud and the receiver and stored in the proxy. If the content is changed at all, 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 proxy must communicate with the KGC to generate a private key and public key pair. During this process, the KGC generates a user's full private key, and the KGC may increase the user's authority. This problem is called the 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 generated legitimately by the correct KGC

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

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

3.3. Algorithms. This section describes the algorithm used for the proposed scheme. Eleven algorithms were used in this study: Setup, Set-Secret-Value, Partial-Key-Extract, Set-Private-Key, Set-Public-Key, Set-Owner-Group, Enc, Re-Key-Gen, Re-Enc, Dec, and Re-Dec. The description of each algorithm is as follows.

(i) *Setup*: this algorithm is executed by inputting a security parameter. With this algorithm, the KGC generates public parameters and master secret keys and publishes the public parameters, which are made available for all users and proxies

(ii) *Set-Secret-Value*: this algorithm is applied by the user. With this algorithm, user i calculates T_i using a randomly selected t_i and sends T_i and ID_i to the KGC

(iii) *Partial-Key-Extract*: this algorithm is performed by KGC. Using this algorithm, the KGC generates the partial key (R_i, k_i) of user i using (T_i, ID_i) and mpk received from user i and sends it to user i

(iv) *Set-Private-Key*: this algorithm is applied by the user. With this algorithm, the user calculates private key sk_i using partial key (R_i, k_i) received from the KGC. The sk_i obtained is kept confidential

(v) *Set-Public-Key*: this algorithm is applied by the user. Using this algorithm, the user calculates the public key pk_i by using the partial key (R_i, k_i) received from the KGC and the secret value t_i generated by user i. The pk_i values obtained are disclosed

(vi) *Initialization, Group Agreement:* this algorithm is run by users to be included in the owner group. With this algorithm, users $\mathscr{G}_j$ that are to be included in the owner group $\mathscr{G}$ exchange the public key $\mathrm{gpk}_{\mathscr{G}}$ with each other to generate the group key

(vii) *Enc:* this algorithm is applied by users included in the owner group. In this algorithm, member $\mathscr{G}_j$ of owner group $\mathscr{G}_j$ encrypts plaintext m with public key $\mathrm{gpk}_{\mathscr{G}}$ of owner group $\mathscr{G}$ to obtain ciphertext CT. Subsequently, the obtained ciphertext, CT, is transmitted to the proxy and stored

(viii) *Re-Key-Gen:* this algorithm is applied by users included in the owner group. With this algorithm, member $\mathscr{G}_j$ of the owner group $\mathscr{G}$ uses the group private key $\mathrm{gsk}_{\mathscr{G}}$ and calculates the reencryption key $\mathrm{RK}_{\mathscr{G}\longrightarrow\mathscr{R}}$ using the receiver's public key $\mathrm{pk}_{\mathscr{R}}$. In this case, the receiver consists of one or more persons. Member $\mathscr{G}_j$ of 1owner group $\mathscr{G}$ passes the reencryption key $\mathrm{RK}_{\mathscr{G}\longrightarrow\mathscr{R}}$ to the proxy

(ix) *Re-Enc:* this algorithm is conducted by a proxy. Using this algorithm, the proxy applies reencryption using the cipher text CT uploaded by the owner group $\mathscr{G}$ and reencryption key $\mathrm{RK}_{\mathscr{G}\longrightarrow\mathscr{R}}$. The reencrypted ciphertext CT_R is then obtained. Subsequently, the acquired CT_R is broadcast

(x) *Dec:* this algorithm is applied by a user included in the owner group. Using this algorithm, a member $\mathscr{G}_j$ of the owner group $\mathscr{G}$ can download ciphertext CT stored in the proxy. Subsequently, members $\mathscr{G}_j$ may obtain plaintext m by decrypting the ciphertext CT with their group private key $\mathrm{gsk}_{\mathscr{G}}$

(xi) *Re-Dec:* this algorithm is conducted using the receiver. With this algorithm, the recipient r_j included in the receiver set $\mathscr{R}$ decrypts the reencrypted ciphertext CT_R received from the proxy with its private key sk_{r_j}, and the plaintext m can thus be obtained

4. Proposed G2M Broadcast Proxy Reencryption

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

4.1. Technical Overview. The basic model of the proposed scheme, as shown in Figure 4, can be broadly divided into five phases: *a Setup Phase*, *Key Generation Phase*, *Group Agreement Phase, Data Storage Phase*, and *Data Broadcast*

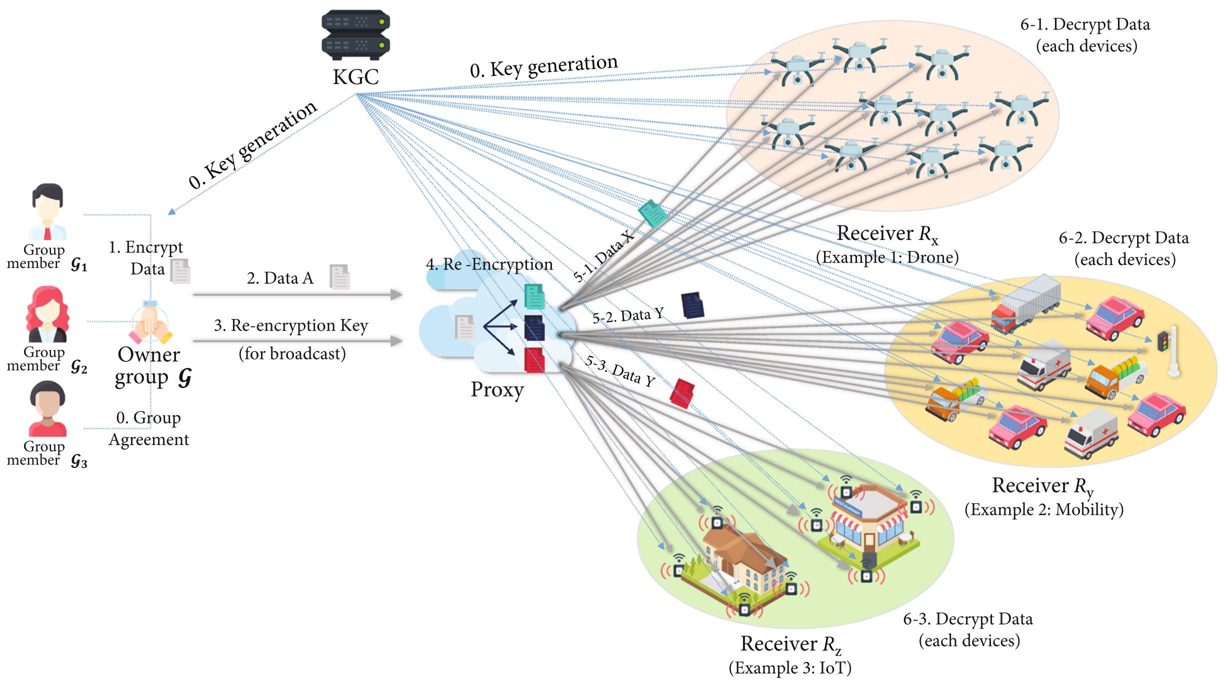
KGC
0. Key generation
0. Key generation
6-1. Decrypt Data (each devices)
Group member $\mathcal{G}_1$
1. Encrypt Data
2. Data A
3. Re-encryption Key (for broadcast)
4. Re -Encryption
5-1. Data X
Receiver R_x (Example 1: Drone)
6-2. Decrypt Data (each devices)
Group member $\mathcal{G}_2$
Owner group $\mathcal{G}$
5-2. Data Y
5-3. Data Y
Proxy
Group member $\mathcal{G}_3$
0. Group Agreement
Receiver R_y (Example 2: Mobility)
6-3. Decrypt Data (each devices)
Receiver R_z (Example 3: IoT)

FIGURE 4: System model of proposed scheme.

Phase. More details regarding these phases are presented in Sections 4.2 and 4.3.

4.2. System Parameters. The system parameters used in the proposed scheme are as follows:

(i) $*$ Participants (KGC, user i, owner group $\mathcal{G}$, owner group member $\mathcal{G}_j$, receiver set $\mathcal{R}$, receiver r_j)

(ii) p, q: λ-bits prime integer

(iii) $\mathbb{E}$: elliptic curve

(iv) $\mathbb{F}_q$: finite field for q

(v) λ: security parameter

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

(vii) P: random generator in $\mathbb{G}_q$ $(P \in \mathbb{G}_q)$

(viii) $\mathbb{G}$: additive group on the elliptical curve, $\mathbb{E}$

(ix) $\mathbb{G}_q$: subgroup of $\mathbb{G}$ with prime order q

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

(xi) msk: KGC system master secret key

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

(xiii) sk_i: user i's private key

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

(xv) $\text{RK}_{\mathcal{G} \longrightarrow \mathcal{R}}$: reencryption key (owner group $\mathcal{G}$ delegates to receiver set $\mathcal{R}$)

(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, $\{0,1\}^* \longrightarrow \mathbb{Z}_q^*$

(xxi) H_2 : one-way hash function, $\{0,1\}^* \times \mathbb{Z}_q^* \longrightarrow \mathbb{Z}_q^*$

(xxii) H_3 : one-way hash function, $\{0,1\}^* \times \mathbb{G}_q \times \mathbb{G}_q \times \mathbb{G}_q \times \mathbb{G}_q \longrightarrow \mathbb{Z}_q^*$

(xxiii) H_4 : one-way hash function, $\mathbb{G}_q \longrightarrow \mathbb{Z}_q^*$

(xxiv) H_5 : one-way hash function, $\mathbb{G}_q \times \{0,1\}^* \longrightarrow \{0,1\}^{l_2}$

(xxv) H_6 : one-way hash function, $\mathbb{G}_q \times \mathbb{G}_q \longrightarrow \{0,1\}^{l_1+l_2}$

(xxvi) H_7 : one-way hash function, $\mathbb{G}_q \times \mathbb{G}_q \times \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$

4.3. Main Algorithm. The scheme was designed based on Kim et al. [51] and Braeken [52]. This scheme is mainly composed of five phases, each of which comprises a *Setup Phase*, *Key Generation Phase*, *Group Agreement Phase*, *Data Storage Phase*, and *Data Broadcast Phase* as shown in Figure 5. A detailed description of each phase is given.

4.3.1. Setup Phase. This phase includes a *Setup* algorithm. This phase is performed by the KGC in advance so that each user can use the proxy. Here, a master public key that can be

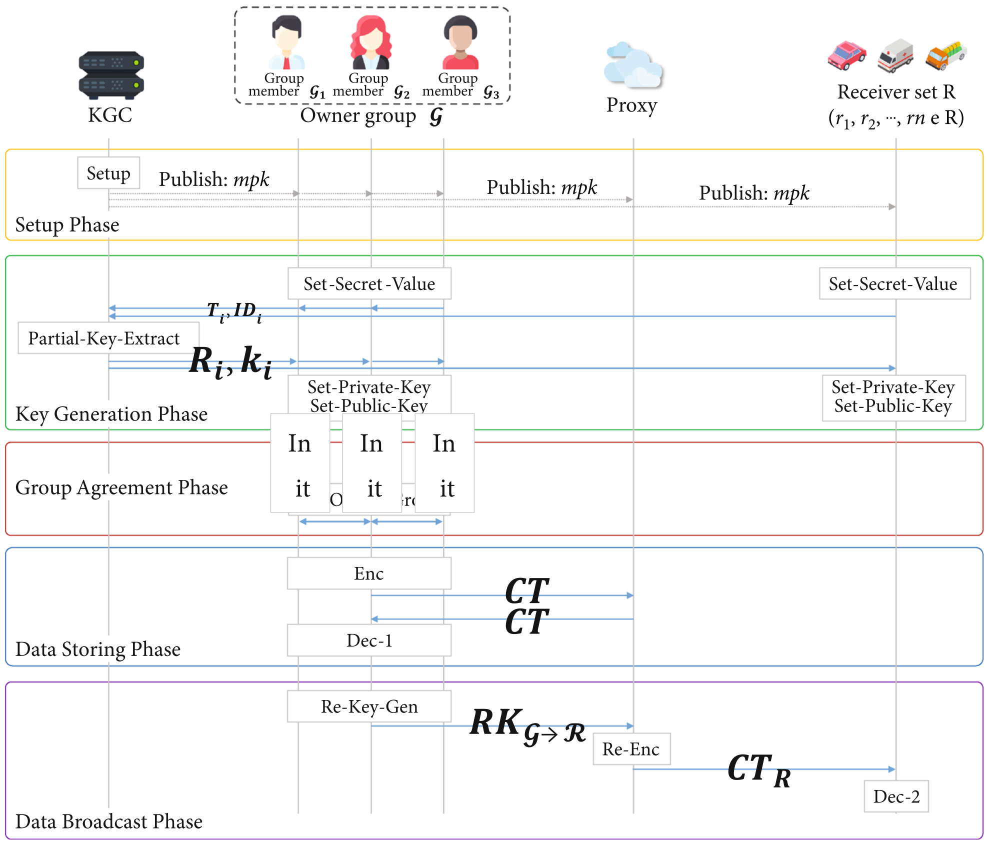

Figure 5: Overview of proposed scheme.

commonly used by each user and a master secret key known only to the KGC are generated.

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

(1) Choose two λ-bits prime integers p, q and elliptic curve $\mathbb{E}$ defined on $\mathbb{F}_p$. Let $\mathbb{G}$ be an additive group on the elliptic curve $\mathbb{E}$ and $\mathbb{G}_q$ be a subgroup of $\mathbb{G}$ with prime order q

(2) Select randomly a generator $P \in \mathbb{G}_q$

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

Select five secure one-way hash functions as follows:

$$H_1 : \{0,1\}^* \longrightarrow \mathbb{Z}_q^*.$$

$$H_2 : \{0,1\}^* \times \mathbb{Z}_q^* \longrightarrow \mathbb{Z}_q^*$$

$$H_3 : \{0,1\}^* \times \mathbb{G}_q \times \mathbb{G}_q \times \mathbb{G}_q \times \mathbb{G}_q \longrightarrow \mathbb{Z}_q^*$$

$$H_4 : \mathbb{G}_q \longrightarrow \mathbb{Z}_q^*$$

$$H_5 : \mathbb{G}_q \times \{0,1\}^* \longrightarrow \{0,1\}^{l_2}$$

$$H_6 : \mathbb{G}_q \times \mathbb{G}_q \longrightarrow \mathbb{Z}_q^*$$

$$H_7 : \mathbb{G}_q \times \mathbb{G}_q \times \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$$

Here, l_1 and l_2 are the lengths of the bit string and are determined by the security parameter λ.

(4) Publish the system's maser public key mpk $= \{p, q, l_1, l_2, E, G, G_q, P, P_{\text{pub}}, H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8, H_9, H_{10}\}$ and message space $= \{0,1\}^{l_1}$

4.3.2. Key Generation Phase

4.3.2. Key Generation Phase. In this phase, the *Set-Secret-Value*, *Partial-Key-Extract*, *Set-Private-Key*, and *Set-Public-Key* algorithms are executed. Each user generates his/her own private key and public key pair so that he/she can use the proxy. Furthermore, each user communicates with the KGC to receive a partial key and uses the partial key to generate his/her own public and private key pair, as shown in Figure 6.

(ii) *Set-Secret-Value*: this algorithm is executed by user i. User i randomly selects $t_i \in \mathbb{Z}_q^*$ and maintains security. User i computes $T_i = t_i \bullet P$ as the public key, and user i sends (T_i, ID_i) to the KGC

(iii) *Partial-Key-Extract*: this algorithm is performed by the KGC. According to the identity ID_i of user i, the KGC performs the following steps.

(1) Randomly select $r_i \in \mathbb{Z}_q^*$ and compute $R_i = r_i \bullet P$

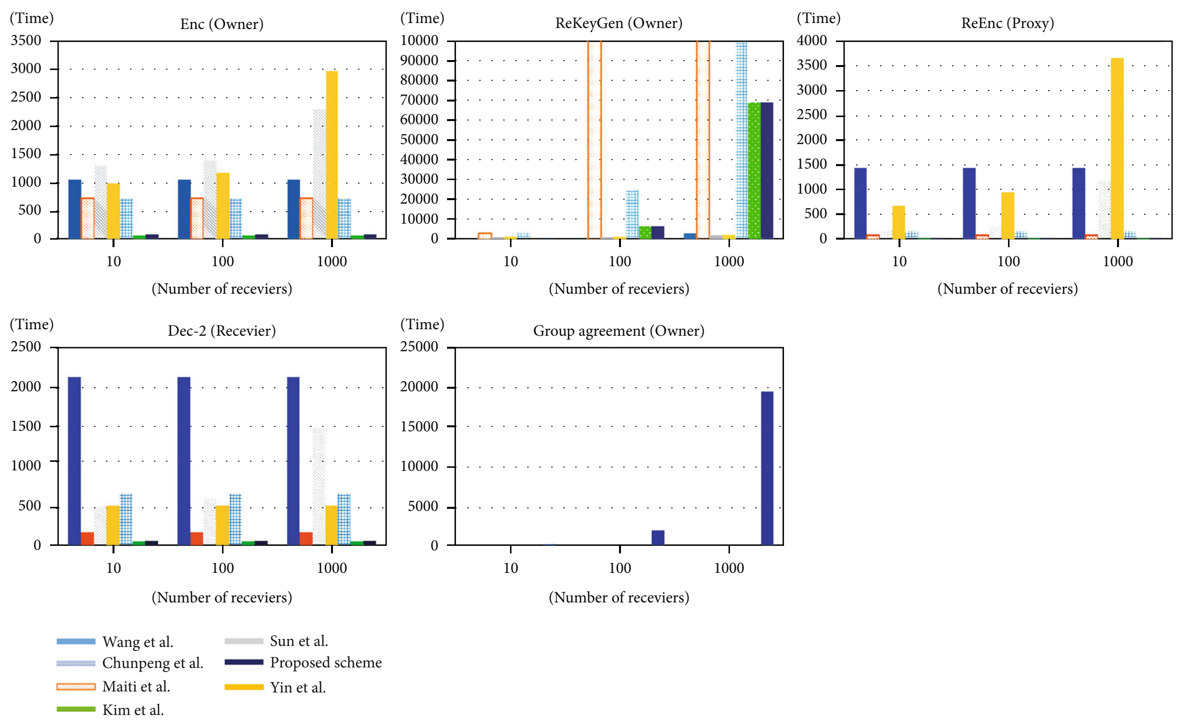

FIGURE 6: Comparison with other schemes.

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

$$k_i \longleftarrow r_i + dH_{10}(R_i, T_i, \mathrm{ID}_i) + H_8(dT_i, \mathrm{ID}_i) (\mathrm{mod}\ q). \tag{1}$$

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

(iv) *Set-Private-Key*: this algorithm is executed by user i. After receiving the partial key (R_i, k_i) from the KGC, user i verifies it as shown in Eqs. (2) and (3). If the key is verified, user i computes the private key $\mathrm{sk}_i = (s_i, t_i)$ as follows:

(4) Verify whether the following equation holds:

$$k_i \bullet P \overset{?}{=} R_i + H_7(R_i, T_i, \mathrm{ID}_i)P_{\mathrm{Pub}} + H_5(t_i P_{\mathrm{Pub}}, \mathrm{ID}_i)P. \tag{2}$$

(5) If not, return $\perp$; otherwise, user i compute s_i.

$$s_i \longleftarrow k_i - H_7(t_i P_{\mathrm{Pub}}, \mathrm{ID}_i). \tag{3}$$

(6) Subsequently, user i keeps secret $\mathrm{sk}_i = (s_i, t_i, k_i)$ as his/her full private key

(v) *Set-Public-Key*: this algorithm is performed by user i . User i keeps $\mathrm{pk}_i = (R_i, T_i)$ as a full public key

4.3.3. Group Agreement Phase. This phase includes the *Initialization* and *Group Agreement* algorithms. It represents the process of forming a group of users who jointly own data. Through this process, all users belonging to a group have equal ownership.

(vi) *Group Agreement*: this algorithm is performed by all group members $\mathcal{G}_i$ who will form group $\mathcal{G}$. Each member creates a secret to share with other members using their private sk_i and public keys pk_i. Each member transmits the generated shared secret to other members and generates a group public key $\mathrm{gpk}_{\mathcal{G}}$ and a group private key $\mathrm{gsk}_{\mathcal{G}}$ using the shared secret sent by other members and their own shared secret as follows:

(7) Group member $\mathcal{G}_i$ computes S_i using $sk_i = (s_i, t_i, k_i)$

$$S_i \longleftarrow s_i \cdot P. \tag{4}$$

(8) Group member $\mathcal{G}_i$ computes h_1 and h_2 for each other group member $\mathcal{G}_j$ $(1 \leq j \leq n, j \neq i)$

$$\begin{aligned} h_1 &\longleftarrow H_3(\mathrm{ID}_i, \mathrm{ID}_j, R_i, R_j, S_i, S_j), \\ h_2 &\longleftarrow H_3(\mathrm{ID}_j, \mathrm{ID}_i, R_j, R_i, S_j, S_i). \end{aligned} \tag{5}$$

(9) Group member $\mathcal{G}_i$ chooses $a_i \in \mathbb{Z}_q^*$ and computes session key ssk_{ij} between $\mathcal{G}_i$ and $\mathcal{G}_j$ and encrypts a_i using a symmetric encryption algorithm

$$\begin{aligned} ssk_{i,j} &\longleftarrow H_4\left((h_1 t_i + s_i)(h_2 T_j + S_j)\right), \\ x_{i,j} &\longleftarrow E_{\mathrm{ssk}_{i,j}}(a_i). \end{aligned} \tag{6}$$

(10) Group member $\mathcal{G}_i$ sends $x_{i,j}\ (1 \le j \le n, j \ne i)$ to each group member and receives $x_{j,i}\ (1 \le j \le n, j \ne i)$ from the other members

(11) All group members of group $\mathcal{G}$ obtain the $a_i\ (1 \le j \le n)$ generated by each group member through the following operation:

$$\begin{aligned} ssk_{i,j} &\longleftarrow H_4\left((h_1 T_i + S_i)(h_2 t_j + s_i)\right), \\ a_i &\longleftarrow D_{\mathrm{ssk}_{i,j}}(x_{i,j}). \end{aligned} \tag{7}$$

(12) Group member $\mathcal{G}_i$ computes group private key $gsk_{\mathcal{G}} = t_{\mathcal{G}}$ and group public key $gpk_{\mathcal{G}} = T_{\mathcal{G}}$

$$\begin{aligned} t_{\mathcal{G}} &\longleftarrow a_1 + a_2 + \cdots + a_n, \\ T_{\mathcal{G}} &\longleftarrow t_{\mathcal{G}} \cdot P. \end{aligned} \tag{8}$$

4.3.4. Data Storing Phase. The *Enc* and *Dec*-1 algorithms are executed in this phase. This phase represents the process of group member $\mathcal{G}_i$ encrypting his/her data with the group public key $\mathrm{gpk}_{\mathcal{G}}$ and storing it in a proxy. In addition, group member $\mathcal{G}_i$ downloads his/her own data stored in the proxy, and a decryption process is included using the group private key $\mathrm{gsk}_{\mathcal{G}}$ to obtain the data source again.

(vii) *Enc*: this algorithm is performed by group member $\mathcal{G}_i$. Group member $\mathcal{G}_i$ encrypts message m with ciphertext CT by entering the group public key $\mathrm{gpk}_{\mathcal{G}} = T_{\mathcal{G}}$ and message $m \in M$. Then, the ciphertext CT is uploaded to the proxy

(13) Group member $\mathcal{G}_i$ computes w, z, and Z using given message $m \in M$ and $gpk_{\mathcal{G}} = T_{\mathcal{G}}$

$$\begin{aligned} w &\longleftarrow H_5(T_{\mathcal{G}}, ID_{\mathcal{G}}), \\ z &\longleftarrow H_1(m \| w), \\ Z &\longleftarrow zP. \end{aligned} \tag{9}$$

(14) Group member $\mathcal{G}_i$ chooses $\alpha \in \mathbb{Z}_q^*$ and calculates β, θ, and C as follows:

$$\begin{aligned} \beta &\longleftarrow \alpha \cdot P, \\ \theta &\longleftarrow T_{\mathcal{G}} \cdot \alpha, \\ C &\longleftarrow H_6(Z, \theta) \oplus (m \| w). \end{aligned} \tag{10}$$

(15) Group member $\mathcal{G}_i$ generates the ciphertext $\mathrm{CT} \longleftarrow (C_1, C_2, C_3) = (C, Z, \beta)$. The generated CT is then uploaded and stored as a proxy

(viii) *Dec*-1: this algorithm is performed by group member $\mathcal{G}_i$. Group member $\mathcal{G}_i$ can download the ciphertext $\mathrm{CT} \longleftarrow (C_1, C_2, C_3) = (C, Z, \beta)$ from the proxy. Group member $\mathcal{G}_i$ who has downloaded the ciphertext CT can obtain the plaintext m by decrypting the ciphertext CT with his/her group private key $\mathrm{gsk}_{\mathcal{G}} = t_{\mathcal{G}}$

(16) Group member $\mathcal{G}_i$ calculates θ' by inputting $\mathrm{gsk}_{\mathcal{G}}$ and C_3

$$\theta' \longleftarrow C_3 \cdot t_{\mathcal{G}.} \tag{11}$$

(17) Group member $\mathcal{G}_i$ computes m by inputting C_1, C_2, θ'

$$\begin{aligned} (m \| w) &\longleftarrow C_1 \oplus H_6\left(C_2, \theta'\right), \\ \because \mathrm{C}_1 \oplus \mathrm{H}_6\left(\mathrm{C}_2, \theta'\right) &= \mathrm{H}_6(\mathrm{Z}, \theta) \oplus (\mathrm{m} \| \mathrm{w}) \oplus \mathrm{H}_6\left(\mathrm{C}_2, \theta'\right) \\ &= \mathrm{H}_6(\mathrm{Z}, \theta) \oplus (\mathrm{m} \| \mathrm{w}) \oplus \mathrm{H}_6\left(\mathrm{Z}, \theta'\right) \\ &= (\mathrm{m} \| \mathrm{w}). \end{aligned} \tag{12}$$

(18) Verify whether the following equation holds. If not, return $\perp$; otherwise, group member $\mathcal{G}_i$ keeps the plaintext m

$$\begin{aligned} C_2 &\overset{?}{=} H_2(m \| H_5(T_{\mathcal{G}}, ID_{\mathcal{G}}))P, \\ \because \mathrm{C}_2 &= H_1(m \| H_5(T_{\mathcal{G}}, \mathrm{ID}_{\mathcal{G}}))P \\ &= H_1(\mathrm{m} \| \mathrm{w})P = \mathrm{z}P = Z. \end{aligned} \tag{13}$$

4.3.5. Data Broadcast Phase. This phase includes the *Re-Key-Gen*, *Re-Enc*, and *Dec*-2 algorithms. In this phase, group member $\mathcal{G}_i$ 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 them to the recipients. A receiver that has received the broadcast ciphertext can obtain the message by decrypting the ciphertext with its private key.

(ix) *Re-Key-Gen*: in this algorithm, group member $\mathcal{G}_i$ specifies a set of recipients $\mathcal{R} = (\mathscr{r}_1, \mathscr{r}_2, \cdots, \mathscr{r}_n)$ and generates a reencryption key $\mathrm{RK}_{\mathcal{G} \longrightarrow \mathcal{R}}$ to delegate the ciphertext CT

(19) Group member $\mathcal{G}_i$ computes U_j for all receiver $\mathscr{r}_j\ (\mathscr{r}_j \in \mathcal{R})$

$$U_j \longleftarrow z \bullet \left(R_j + H_7\left(R_j, T_j, \mathrm{ID}_j\right) P_{\mathrm{pub}} + T_j\right). \tag{14}$$

(20) Group member $\mathcal{G}_i$ computes a polynomial $f(x)$ with degree n using $\gamma \in \mathbb{Z}_q^*$ as follows:

$$\begin{aligned}\mu &\longleftarrow \alpha\cdot\gamma,\\ f(x) &= \prod_{i=0}^{n}\left(x-U_j\right)+\mu\,(\mathrm{mod}\ q)\\ &= x^n+\varphi_{n-1}x^{n-1}+\cdots+\varphi_1x+\varphi_0,\end{aligned}\tag{15}$$

where $\varphi_i \in \mathbb{Z}_p^*$ $(i=0,1,\cdots,n-1)$

(21) Group member $\mathscr{G}_i$ computes ζ using $\mathrm{gsk}_{\mathscr{G}}=t_{\mathscr{G}}$ and γ as follows:

$$\zeta \longleftarrow (\gamma+1)\cdot t_{\mathscr{G}}.\tag{16}$$

(22) Group member $\mathscr{G}_i$ generates a reencryption key $\mathrm{RK}_{\mathscr{G}\longrightarrow\mathscr{R}}=(\mathrm{rk}_1,\mathrm{rk}_2)=(\zeta,\{\varphi_0,\varphi_1,\cdots,\varphi_{n-1}\})$ and sends $\mathrm{RK}_{\mathscr{G}\longrightarrow\mathscr{R}}$ to the proxy

(x) *Re-Enc*: this algorithm is executed using a proxy. This algorithm reencrypts the ciphertext $\mathrm{CT}\longleftarrow(C_1,C_2,C_3)=(C,Z,\beta)$ into ciphertext CT_R using the reencryption key $\mathrm{RK}_{\mathscr{G}\longrightarrow\mathscr{R}}$

(23) Compute CT_R using ciphertext CT and reencryption key $\mathrm{RK}_{\mathscr{G}\longrightarrow\mathscr{R}}$

$$\begin{aligned}C_1' &\longleftarrow C_1,\\ C_2' &\longleftarrow C_2,\\ C_3' &\longleftarrow C_3\cdot\mathrm{rk}_1,\\ C_4' &\longleftarrow \mathrm{rk}_2.\end{aligned}\tag{17}$$

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

(xi) *Dec*-2: this algorithm is executed by the selected receiver r_j to extract plaintext from the received ciphertext $CT_R=(C_1',C_2',C_3',C_4')$. Receiver r_j performs the following steps:

(25) Compute U_j

$$U_j' \longleftarrow \left(s_j+t_j\right)\bullet C_1'.\tag{18}$$

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

$$\begin{aligned}f(x) &= x^n+\varphi_{n-1}x^{n-1}+\cdots+\varphi_1x+\varphi_0,\\ \mu' &= f\left(U_j'\right).\end{aligned}\tag{19}$$

(27) Compute θ' as an input C_3' and β'

$$\theta' = C_3' - \mu'\cdot T_{\mathscr{G}},$$

$$\begin{aligned}\because C_3'-\mu'\cdot T_{\mathscr{G}} &= C_3\cdot rk_1-\alpha\cdot\gamma\cdot T_{\mathscr{G}}\\ &= \beta\cdot\zeta-\alpha\cdot\gamma\cdot T_{\mathscr{G}}\\ &= \alpha\cdot P\cdot(\gamma+1)\cdot t_{\mathscr{G}}-\alpha\cdot\gamma\cdot T_{\mathscr{G}}\\ &= \alpha\cdot P\cdot\gamma\cdot t_{\mathscr{G}}+\alpha\cdot P\cdot t_{\mathscr{G}}-\alpha\cdot\gamma\cdot T_{\mathscr{G}}\\ &= \alpha\cdot P\cdot t_{\mathscr{G}}=\theta.\end{aligned}\tag{20}$$

(28) Compute m as an input C_1', C_2', θ'

$$(m\|w) \longleftarrow C_1'\oplus H_6\left(C_2',\theta'\right),$$

$$\begin{aligned}\because C_1'\oplus H_6\left(C_2',\theta'\right) &= (m\|w)\oplus H_6(Z,\theta)\oplus H_6\left(C_2',\theta'\right)\\ &= (m\|w)\oplus H_6(Z,\theta)\oplus H_6\left(Z,\theta'\right)\\ &= (m\|w),\end{aligned}\tag{21}$$

where $C_1'=C_1=Z$

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

$$\begin{aligned}C_2' &\overset{?}{=} H_1(m\|w)P,\\ \because C_2' &= H_1(m\|w)P=\mathrm{z}P=Z,\end{aligned}\tag{22}$$

where $Z=zP$ and $z=H_1(m\|w)$

5. Analysis of the Proposed G2M BPRE Scheme

In this section, we perform a security analysis and computational analysis of the security requirements of the proposed scheme.

5.1. Analysis of the Security Requirements. In this section, we analyze the security requirements presented in Section 3.2. Here, we analyze the security of the seven security requirements, as shown in Table 1.

(i) *Confidentiality*: this proposed method performs an encryption operation based on elliptic curve encryption. Because elliptic curve encryption provides high security, even with a short key, efficient encryption is possible. The proposed method uses this elliptic curve encryption method such that a user without a decryption key cannot know the contents of the data. First, the proposed method encrypts a message using a public key:

TABLE 1: Comparison of the security requirements.

	Group ownership	Bilinear pairing	Key escrow problem	Receiver anonymity	Re-Key genneration
Wang and Wang [53]	Not provided	Used	Insecure	Offer	KGC/BC
Maiti and Misra [54]	Not provided	Used	Insecure	Offer	Sender
Sun et al. [55]	Not provided	Used	Insecure	Offer	Sender
Yin et al. [56]	Not provided	Used	Insecure	Offer	Sender
Chunpeng et al. [57]	Not provided	Used	Insecure	Offer	Sender
Kim et al. [51]	Not provided	Not used	Secure	Offer	Sender
Proposed scheme	Provided	Not used	Secure	Offer	Sender

$$\begin{aligned} w &\longleftarrow H_5(T_{\mathscr{G}}, ID_{\mathscr{G}}), \\ z &\longleftarrow H_1(m\|w), \\ Z &\longleftarrow zP, \\ \beta &\longleftarrow \alpha \cdot P, \\ \theta &\longleftarrow T_{\mathscr{G}} \cdot \alpha, \\ C &\longleftarrow H_6(Z, \theta) \oplus (m\|w). \end{aligned} \tag{23}$$

Here, message encryption is performed by the XOR operation, and θ in the XOR operation is created with the owner's public key. In addition, the owner's private key is required to create θ using the ciphertext C_3. Accordingly, the ciphertext of the proposed method can only be decrypted with the group private key $\mathrm{psk}_{\mathscr{G}}$ paired with the group public key $\mathrm{gpk}_{\mathscr{G}}$ used for encryption.

(ii) *Integrity*: recipients who decrypt the data can verify the integrity of the data using the values contained in the integrity ciphertext and parameters of the public KGC. The proofing methods are as follows.

$$\begin{aligned} C_2' &\overset{?}{=} H_1(m\|w)P, \\ \because C_2' &= H_1(m\|w)P = \mathrm{z}P = Z, \end{aligned} \tag{24}$$

where $Z = zP$ and $z = H_1(m\|w)$.

The receiver that decrypts the ciphertext CT_R can obtain message m and verification value w. Here, $H_1(m\|w)$ is equal to z; thus, the integrity of the message can be verified by determining whether $H_1(m\|w)P$ is equal to $C_2 = Z$.

(iii) *Key escrow problem*: in the certificate-based public key encryption method, a certificate corresponding to the public key must be issued and stored. To solve this problem, a certificateless public-key encryption method may be used. However, in the general certificate public-key encryption method, the KGC generates and delivers the user's private key. Thus, because the KGC user's complete private key is known, the key escrow problem of the KGC may occur. In this study, an algorithm is designed using the partial-key method to solve this problem

First, the user creates his/her secret value t_i, converts it into T_i, and transmits it to the KGC. Upon receiving T_i, KGC generates a secret value r_i for the user, generates k_i through the following calculation process, and delivers (R_i, k_i) to the user.

$$\begin{aligned} R_i &= r_i \bullet P, \\ k_i &\longleftarrow r_i + dH_7(R_i, T_i, \mathrm{ID}_i) + H_5(dT_i, \mathrm{ID}_i) (\bmod\ q). \end{aligned} \tag{25}$$

The user who receives (R_i, k_i) from the KGC calculates s_i using k_i and t_i known only to the user as follows:

$$s_i \longleftarrow k_i - H_7(t_i P_{\mathrm{Pub}}, \mathrm{ID}_i). \tag{26}$$

Thereafter, the user uses (s_i, t_i, k_i) as private keys and (R_i, T_i) as public keys.

Finally, T_i generated by the user and R_i generated by the KGC are used as public keys. Consequently, the partial key known to the KGC and the unknown partial key are as follows:

KGC only knows $pk_i = (T_i, R_i)$ and k_i

KGC cannot knows $sk_i = (s_i, t_i)$

(iv) *Partial key verifiability*: the proposed scheme uses a partial key in the key generation process to solve the key-escrow problem. However, it is possible for the malicious KGC to deliver the generated partial key with a value other than the T_i passed to the KGC by the user. To solve this problem, the proposed scheme provides a partial key verification function through the following operation:

$$\begin{aligned} k_i \bullet P &\overset{?}{=} R_i + H_7(R_i, T_i, \mathrm{ID}_i) P_{\mathrm{Pub}} + H_5(t_i P_{\mathrm{Pub}}, \mathrm{ID}_i) P, \\ \because k_i \bullet P &= r_i \bullet P + H_7(R_i, T_i, \mathrm{ID}_i) \cdot \mathrm{d} \cdot P + H_5(t_i P_{\mathrm{Pub}}, \mathrm{ID}_i) P \\ &= (r_i + H_7(R_i, T_i, \mathrm{ID}_i) \cdot d + H_5(t_i \cdot \mathrm{d} \cdot P, \mathrm{ID}_i)) P \\ &= (r_i + \mathrm{d} \cdot H_7(R_i, T_i, \mathrm{ID}_i) + H_5(T_i \cdot d_i, \mathrm{ID}_i)) P = (k_i) P, \end{aligned} \tag{27}$$

where $k_i = r_i + dH_7(R_i, T_i, \mathrm{ID}_i) + H_5(dT_i, \mathrm{ID}_i)$,

$$R_i = r_i P, T_i = t_i P, P_{\mathrm{pub}} = dP. \tag{28}$$

Table 2: Comparison of the computation efficiency.

	Enc	Re-Key gen	Re-Enc	Dec-2	Group agreement
Wang and Wang [53]	$(2)T_M + (4)T_e + (1)T_P$	$(10 + 3n)T_M + (1)T_e$	$(6)T_e$	$(7)T_M + (7)T_e + (5)T_P$	—
Maiti and Misra [54]	$(4)T_M + (3)T_e$	$\left(3 + n^2 + n\right)T_M + (3 + n)T_e$	$(1)T_M + (1)T_P$	$(1)T_M + (2)T_P$	—
Sun et al. [55]	$(2 + \mathrm{n})T_M + (5)T_e + (1)T_P$	$(3 + n)T_M + (6)T_e + (1)T_p$	$(1 + \mathrm{n})T_M + (2)T_P$	$(4 + \mathrm{n})T_M + (2)T_e$	—
Yin et al. [56]	$(4 + 2n)T_M + (4)T_e$	$(4 + 2n)T_M + (4)T_e$	$(4 + 3n)T_M + (2)T_e + (2)T_p$	$(7)T_M + (1)T_e + (3)T_P$	—
Chunpeng et al. [57]	$(2)T_M + (3)T_e$	$(5 + n)T_M + (5 + n)T_e + (1)T_p$	$(1)T_M + (2)T_p$	$(6)T_M + (2)T_e + (2)T_p$	—
Kim et al. [51]	$(2)T_{\mathrm{EM}} + (2)T_{\mathrm{EA}}$	$(2 + n)T_M + (2n)T_{\mathrm{EM}} + (2n)T_{\mathrm{EA}}$	$(1)T_{\mathrm{EM}}$	$(2)T_{\mathrm{EM}}$	—
Proposed scheme	$(3)T_{\mathrm{EM}}$	$(1 + n)T_M + (2n)T_{\mathrm{EM}} + (2n)T_{\mathrm{EA}}$	$(1)T_{\mathrm{EM}}$	$(2)T_{\mathrm{EM}} + (1)T_{\mathrm{EA}}$	$(1 + 6n))T_{\mathrm{EM}} + (4(n - 1))T_{\mathrm{EA}}$

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.

(v) *Receiver anonymity*: in the proposed scheme, the public key and ID of the recipient are used to designate multiple recipients. This method was designed based on multireceiver encryption. However, in the existing multireceiver encryption, other users can identify the recipient because the ciphertext contains information that can identify the recipient. To solve this problem, in this study, a receiver identification process was designed using a polynomial, as follows:

$$\begin{aligned} f(x) &= \prod_{i=0}^{n}(x - U_i) + \beta \pmod q \\ &= (x - U_1)\bullet(x - U_2)\bullet\cdots\bullet(x - U_n) \\ &\quad + \mu \pmod q \\ &= x^n + \varphi_{n-1}x^{n-1} + \cdots + \varphi_1 x + \varphi_0, \end{aligned}$$

$$\mathrm{U}_\mathrm{j}' = \left(\mathrm{s}_\mathrm{j} + \mathrm{t}_\mathrm{j}\right)\bullet \mathrm{C}_1'. \tag{29}$$

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

$$U_i' \longleftarrow \overset{?}{z} \bullet \left(R_i + H_7(R_i, T_i, ID_i)P_{\text{pub}} + T_i\right). \tag{30}$$

As a result, the attacker cannot identify the recipient.

(vi) *Decryption fairness*: as described in the receiver anonymity section, each receiver's public key and ID are used to designate multiple receivers. However, in the design process, there is a threat that a specific receiver performs more operations during the decoding process or makes decoding impossible. This is known as the decryption fairness problem. Such problems can be caused by removing or changing some elements in the data that specify and validate the recipient. In the proposed scheme, an algorithm is designed using polynomials to address this problem. These polynomials, which can only be changed and falsified by the user who created them, are as follows:

$$\begin{aligned} f(x) &= \prod_{i=0}^{n}(x - U_i) + \beta \pmod q \\ &= (x - U_1)\bullet(x - U_2)\bullet\cdots\bullet(x - U_n) + \mu \pmod q \\ &= x^n + \varphi_{n-1}x^{n-1} + \cdots + \varphi_1 x + \varphi_0. \end{aligned} \tag{31}$$

5.2. Analysis of Computational Efficiency. The scheme proposed in this study was designed to provide extended functions based on the method proposed by Kim et al. Accordingly, its overall structure is similar to that reported by Kim et al., but its detailed calculations are different. As shown in Figure 6 and Table 2, the computation time of the proposed scheme is almost the same as that of Kim et al. There are differences in some calculations; however, they are not so large in terms of the total number of calculations. In addition, compared with other schemes, the reencryption key generation algorithm requires a relatively larger amount of computation time than the other algorithms in the scheme. In addition, in the proposed scheme, a group agreement algorithm is additionally used to provide a group joint ownership function. Accordingly, although its total computation time is greater than that of other schemes, the proposed method is able to perform group-owned functions that cannot be executed by other schemes.

6. Conclusion

This study examined the extended form of proxy reencryption. Existing proxy reencryption technology provides a data delegation method that assumes one owner and one receiver. It provides an intuitive and clear form of data communication. However, owing to recent technological developments, an environment in which multiple devices exchange data, such as device-to-device communication, rather than human-to-human communication, is becoming common. A typical example of this is the IoT environment. The IoT environment is an environment in which multiple devices communicate with each other and share and use data for various purposes. However, in this environment, existing proxy reencryption for 1 : 1 communication is inevitably inefficient. In an IoT environment, where the same data must be delivered to multiple devices in the same way, when using the existing proxy reencryption, the same data must be reencrypted several times. This method inevitably reduces the data transfer efficiency. In addition, in a large-scale communication environment, an environment in which multiple users form a group to create and own data can be presented. However, because the existing proxy reencryption is a form in which only one user can be the owner, data ownership disputes may arise. To solve this problem, this study proposes proxy reencryption, which can support multiple owners and recipients. In addition, to increase the security and efficiency of the proposed technology, only elliptic curve encryption is used, and security is improved using the partial key form. However, because the proposed scheme uses a group key method that has not been used in other existing schemes, the group agreement algorithm is additionally applied and requires a relatively large amount of computation time. As a result, the proposed method provides more functions than the existing proxy reencryption and improved security; however, it requires additional computation. This method can be used more effectively in environments in which scalability is more important than computational efficiency.

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 work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1A2B5B01002490) and Soonchunhyang University Research Fund

References

[1] Gartner, "Gartner top strategic technology trends for 2022," Technical report, Gartner, 2022.

[2] 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, Espoo, Finland, 1998.

[3] 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*, vol. 7, no. 2, pp. 766–775, 2020.

[4] Z. Cai, X. Zheng, J. Wang, and Z. He, "Private data trading towards range counting queries in Internet of Things," *IEEE Transactions on Mobile Computing*, pp. 1–17, 2022.

[5] J. Byabazaire, G. O'Hare, and D. Delaney, "Data quality and trust: review of challenges and opportunities for data sharing in iot," *Electronics*, vol. 9, no. 12, p. 2083, 2020.

[6] 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.

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

[8] R. H. Deng, 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 *11th International Workshop on Practice and Theory in Public-Key Cryptography*, pp. 360–379, Barcelona, Spain, 2008.

[10] J. Shao and Z. Cao, "Cca-secure proxy re-encryption without pairings," in *International Workshop on Public Key Cryptography*, pp. 357–376, Irvine, CA, USA, 2009.

[11] 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.

[12] 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.

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

[14] 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, p. 181, 2013.

[15] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan, "Generative adversarial networks," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–38, 2021.

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

[17] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Workshop on the Theory and Application of Cryptographic Techniques*, pp. 47–53, Springer, 1985.

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

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

[20] 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 *European symposium on research in computer security*, pp. 257–272, Wroclaw, Poland, 2014.

[21] A. Paul, S. Varshika Srinivasavaradhan, S. D. Selvi, and C. P. Rangan, "A ca-secure collusion-resistant identity-based proxy re-encryption scheme," in *International Conference on Provable Security*, pp. 111–128, Jeju, South Korea, 2018.

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

[23] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *International conference on the theory and application of cryptology and information security*, pp. 452–473, Taipei, Taiwan, 2003.

[24] X. Lei, W. Xiaoxin, 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.

[25] W. Xiaoxin, X. Lei, and X. Zhang, "Poster: a certificateless proxy re-encryption scheme for cloud-based data sharing," in *Proceedings of the 18th ACM conference on computer and communications security*, pp. 869–872, Chicago Illinois USA, 2011.

[26] K. Yang, X. Jing, and Z. Zhang, "Certificateless proxy re-encryption without pairings," in *International Conference on Information Security and Cryptology*, pp. 67–88, Seoul, Korea, 2014.

[27] X. Zheng, Y. Zhou, Y. Ye, and F. Li, "A cloud data deduplication scheme based on certificateless proxy re-encryption," *Journal of Systems Architecture*, vol. 102, article 101666, 2020.

[28] J. Baek, R. Safavi-Naini, and W. Susilo, "Efficient multi-receiver identity-based encryption and its application to broadcast encryption," in *International Workshop on Public Key Cryptography*, pp. 380–397, Springer, 2005.

[29] S. Chatterjee and P. Sarkar, "Multi-receiver identity-based key encapsulation with shortened ciphertext," in *International Conference on Cryptology in India*, pp. 394–408, Kolkata, India, 2006.

[30] P. Vijayakumar, S. Bose, A. Kannan, and L. J. Deborah, "Computation and communication efficient key distribution protocol for secure multicast communication," *KSII Transactions on Internet and Information Systems*, vol. 7, no. 4, pp. 878–894, 2013.

[31] I. Kim and S. O. Hwang, "An optimal identity-based broadcast encryption scheme for wireless sensor networks," *IEICE Transactions on Communications*, vol. E96.B, no. 3, pp. 891–895, 2013.

[32] 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.

[33] J. Kim, W. Susilo, A. Man Ho, and J. Seberry, "Adaptively secure identity-based broadcast encryption with a constant-sized ciphertext," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 679–693, 2015.

[34] F.-C. Zhou, M.-Q. Lin, Y. Zhou, and Y.-X. Li, "Efficient Anonymous broadcast encryption with adaptive security," *KSII Transactions on Internet and Information Systems*, vol. 9, no. 11, pp. 4680–4700, 2015.

[35] J. Li, Y. Qihong, and Y. Zhang, "Identity-based broadcast encryption with continuous leakage resilience," *Information Sciences*, vol. 429, pp. 177–193, 2018.

[36] J. Lai, M. Yi, F. Guo, P. Jiang, and S. Ma, "Identity- based broadcast encryption for inner products," *The Computer Journal*, vol. 61, no. 8, pp. 1240–1251, 2018.

[37] C.-I. Fan, L.-Y. Huang, and P.-H. Ho, "Anonymous multireceiver identity-based encryption," *IEEE Transactions on Computers*, vol. 59, no. 9, pp. 1239–1249, 2010.

[38] Y. Huaqun Wang, Z.,. H. Xiong, and B. Qin, "Cryptanalysis and improvements of an anonymous multi-receiver identity-based encryption scheme," *IET Information Security*, vol. 6, no. 1, pp. 20–27, 2012.

[39] H.-Y. Chien, "Improved anonymous multi-receiver identity-based Encryption," *The Computer Journal*, vol. 55, no. 4, pp. 439–446, 2012.

[40] C.-I. Fan, P.-J. Tsai, J.-J. Huang, and W.-T. Chen, "Anonymous multi-receiver certificate-based encryption," in *2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 19–26, Beijing, China, 2013.

[41] M. Zhang and T. Takagi, "Efficient constructions of anonymous multireceiver encryption protocol and their deployment in group e-mail systems with privacy preservation," *IEEE Systems Journal*, vol. 7, no. 3, pp. 410–419, 2013.

[42] J. Zhang and J. Mao, "An improved anonymous multi-receiver identity-based encryption scheme," *International Journal of Communication Systems*, vol. 28, no. 4, pp. 645–658, 2015.

[43] C. Sur, C. D. Jung, and K. H. Rhee, "Multi-receiver certificate-based encryption and application to public key broadcast encryption," in *2007 ECSIS Symposium on Bio-inspired, Learning, and Intelligent Systems for Security (BLISS 2007)*, pp. 35–40, Edinburgh, UK, 2007.

[44] C. Sur, Y.-H. Park, and K.-H. Rhee, "A multi-receiver certificateless encryption scheme and its application," *Journal of Korea Multimedia Society*, vol. 14, no. 6, pp. 775–784, 2011.

[45] S. K. Hafizul Islam, M. K. Khan, and A. M. Al-Khouri, "Anonymous and provably secure certificateless multireceiver encryption without bilinear pairing," *Security and Communication Networks*, vol. 8, no. 13, p. 2231, 2015.

[46] Y.-H. Hung, S.-S. Huang, Y.-M. Tseng, and T.-T. Tsai, "Efficient anonymous multireceiver certificateless encryption," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2602–2613, 2017.

[47] 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.

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

[49] J. Zhu, L.-L. Chen, X. Zhu, and L. Xie, "A new efficient certificateless multi-receiver public key encryption scheme," *International Journal of Computer Science Issues*, vol. 13, no. 6, pp. 1–7, 2016.

[50] E. K. Win, T. Yoshihisa, Y. Ishi, T. Kawakami, Y. Teranishi, and S. Shimojo, "A lightweight multi-receiver encryption scheme with mutual authentication," in *2017 IEEE 41st annual computer software and applications conference (COMPSAC)*, pp. 491–497, Turin, Italy, 2017.

[51] W.-B. Kim, S.-H. Kim, D. Seo, and I.-Y. Lee, "Broadcast proxy reencryption based on certificateless public key cryptography for secure data sharing," *Wireless Communications and Mobile Computing*, vol. 2021, 16 pages, 2021.

[52] A. Braeken, "Pairing free certified common asymmetric group key agreement protocol for data sharing among users with different access rights," *Wireless Personal Communications*, vol. 121, no. 1, pp. 307–318, 2021.

[53] X. 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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, pp. 1214–1226, 2019.