Research Article

A Secure Multimedia Data Sharing Scheme for Wireless Network

Liming Fang,^{1,2} Liang Liu ,^{1,2} Jinyue Xia,³ and Maosheng Sun⁴

¹Nanjing University of Aeronautics and Astronautics, 29 Yudao Street, Nanjing, Jiangsu 210016, China

²Key Laboratory of Computer Network Technology of Jiangsu Province, China

³IBM, 3039 E Cornwallis Rd, Research Triangle Park, NC 27709, USA

⁴Yangzhou University, 88 South University Ave., Yangzhou, Jiangsu 225009, USA

Correspondence should be addressed to Liang Liu; flm12311231@hotmail.com

Received 12 June 2018; Accepted 3 September 2018; Published 18 October 2018

Guest Editor: Zhaoqing Pan

Copyright © 2018 Liming Fang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A large number of wireless devices like WiFi cameras and 4G robots have been deployed in the rapidly growing wireless network such as Internet of Things. All of the devices (sensors) are collecting and analyzing multimedia data all the time while they are actively working, and it is also required to share data among these the sensors. Typically, the wireless data is transmitted through the network gateway or the cloud platforms. In such a wireless environment, if there is no appropriate protection to the data, it is easy to cause potential data leakage. In reality, the owner of the sensor might only want to share the multimedia data stored in the sensor with a trusted third party (e.g., a family member or a coworker) through an internet gateway or the cloud platform. Ideally, the gateway or the cloud platform in the wireless network should transform one user's encrypted data (wireless multimedia data) directly into another ciphertext under a set of new users (e.g., a trusted third party) without accessing the user's plaintext data. In this work, a new secure notion called fuzzy-conditional proxy broadcast re-encryption (FC-PBRE) is presented to address the concern. In a FC-PBRE scheme, the proxy (the gateway or cloud server) uses a broadcast re-encryption key to re-encrypt the encrypted wireless multimedia data which can be decrypted by a set of delegates if and only if the broadcast key's conditional set W is close to the conditional set W' of the ciphertext. With the FC-PBRE scheme, the wireless multimedia data is not disclosed and cannot be learnt by the proxy (the gateway or cloud server). In this paper, we first present the definition of security against chosen-ciphertext attacks for FC-PBRE. Second, we propose an efficient fuzzy-conditional proxy broadcast re-encryption scheme. Third, we prove that our FC-PBRE scheme is CCA-secure in the random oracle model based on the Decisional nBDHE assumption.

1. Introduction

Generally, a user (can be an attacker) has the access to the data stored in the wireless devices via (1) a direct connection with the devices, (2) the gateway of the network, and (3) a cloud platform. Because many wireless devices need to localize the configuration before they start working, they usually have a backend configuration interface (e.g., WEB, PC) open. Moreover the devices sometimes go offline even when they are actively working. As a result, the attacker can simply hack the login system of the configuration interface and obtain the multimedia data. In this case, even though the device and the cloud have been authenticated and encrypted (e.g., using SSL), the attack cannot be prevented. So, ensuring the multimedia data to be stored and shared securely has become extremely important [1–8]. To share the data partially, it

would be good if all the multimedia data are encrypted via the data owner's public key. For example, the device owner can give the data permission of the wireless camera configuration to the maintenance engineer, but the maintenance is not allowed to access to the video data. Obviously, such a scheme cannot allow sharing the private key directly with the maintenance engineer because sharing the private key means exposing the data to the engineer.

One approach to ensure data confidentiality in the wireless network is that user data can be encrypted before it is updated into the server. The encryption technology is an effective way to protect user data; however it does have some drawbacks. For example, when a cloud user Alice shares her data with another user Bob, her encrypted data cannot be as the plain data and cloud servers should not be directly transmitted to the shared user's ciphertext, because users

(including Bob) cannot decrypt the received data. Of course, one easy way for Alice to fix this issue is to download her own encrypted data that is saved in the cloud, then decrypt and upload the decrypted data to the cloud again, and finally send it to Bob who Alice wants to share the data. However, using this method, the user's data will be obtained by the untrusted third-party cloud, which cannot guarantee the data confidentiality. Therefore, in the cloud storage environment, a security mechanism is needed to allow the cloud server to transform the encrypted data of users directly into another shared user's encrypted data without accessing the user's plaintext data.

Since the data security problem of users in the nontrusted third-party cloud server is increasingly prominent and traditional encryption technology has been unable to meet these application needs, the cloud server really should be able to convert a user's ciphertext to a second's user ciphertext on the basis of not involving decryption. There have been quite a few researches done in this area to address this concern. For example, proxy re-encryption [9] can directly transfer encrypted data of a user Alice, stored in a nontrusted third-party cloud under the authorization of Alice, into user Bob's ciphertext, so as to achieve data sharing of Alice and Bob. Because of this feature, proxy re-encryption has been applied in IoT, cloud computing, email forwarding systems [9], and distributed file systems [10]. In proxy re-encryption, the third party can convert all Alice's ciphertexts into Bob's ciphertext, but in many applications, Alice hopes that the third party can only transform some specific conditions of ciphertext instead of all ciphertext. For example, the owners of wireless devices might only want to share part of the encrypted multimedia data instead of sharing all the data with a group of other users. To achieve that goal, conditional proxy re-encryption was proposed to provide such a mechanism that allows Alice to freely determine which encrypted data needs to be shared with Bob [11]. The third party in the conditional proxy re-encryption scheme has the ability to convert a ciphertext only if it meets certain specific conditions.

For applications like group photo sharing, however, conditional proxy re-encryption scheme becomes a problem if one person's multimedia needs to be shared with a group of users via a cloud platform. For example, in a scenario of a picture's owner (Alice) who wishes to share the encrypted photo data with the family members, the cloud cannot directly forward the owner Alice's encrypted data of camera to a group of family members, since only Alice has the private key to decrypt after forwarding. Although conditional proxy re-encryption can convert Alice's ciphertext into a different ciphertext, only the one person can decrypt the ciphertext, not a group of people to decrypt. Thus it cannot be adapted to the situation of group data sharing. Recently, the conditional proxy broadcast re-encryption (C-PBRE) was presented [12] to resolve the issue. In a C-PBRE scheme, a user's ciphertext can be transformed to another ciphertext for a group of users by a third party proxy. Moreover, the third-party proxy can only convert ciphertext with specific conditions.

Although the C-PBRE has addressed some concerns, still there is a flaw, which is that the C-PBRE scheme cannot

support fuzzy condition matching. Here, we give an example of an online medical service system to show the importance of fuzzy condition matching and the fuzzy-conditional proxy broadcast re-encryption (FC-PBRE). Usually a multimedia electronic medical record is a system that tracks the electronic medical record of the patients. It integrates image, video, audio, and text and everything can be stored in the cloud at the same time. Doing so enables the medical staff to look up all medical records related to the patient such as an MRI or X-ray image from a PC or a smart phone. In order to protect patient privacy of multimedia electronic medical records, we need to encrypt and protect relevant data. But how to implement the sharing of multimedia electronic medical records after encryption is a difficult problem and FC-PBRE comes to solve this problem. More specifically, in an online medical system, patients are more likely to find a doctor who meets the following requirements for the treatment of a cold. We can simply denote the requirements as follows: $R_1 = (\text{"Cold"} \wedge \text{"fever"} \wedge \text{"runny nose"} \wedge \text{"sore throat"})$. With R_1 a patient can encrypt her health record before she uploads it to the medical system. However, the medical system cannot directly access the disease record in this case, because it does not necessarily have a matching secret key under R_1 . What the system can do is re-encrypt the ciphertext so that other doctors may be able to see the case as long as these doctors meet at least d ($d \leq |R_1|$) conditions of R_1 . By adopting FC-PBRE, a doctor sets up a different access policy R_2 and sends a re-encryption key rk_{R_1} to the proxy. When a doctor is away, the proxy can re-encrypt the ciphertext if and only if $|R_1 \cap R_2| \geq d$. However, in many situations Alice may want to cooperate with a set of colleagues satisfying R_2 to consultation on the patient's condition. In traditional FC-PRE, if Alice wants to consult with N colleagues, the proxy needs to perform N re-encryption operations. The problem, though, is that the proxy's computation is linear with N ; this may not be desirable in terms of the complexity. In contrast, in FC-PBRE, the proxy can re-encrypt Alice's ciphertext to a group of users at one time. As a result, an FC-PBRE scheme is likely to resolve this complexity problem.

In the multimedia data sharing environment, it is needed to have a secure mechanism that allows the cloud server to transform one user's encrypted data directly into another ciphertext under a set of new users without accessing the user's plaintext data. In this paper, fuzzy-conditional proxy broadcast re-encryption (FC-PBRE) is proposed to address the concern. In FC-PBRE, the proxy uses a broadcast re-encryption key to re-encrypt a ciphertext which can be decrypted by a set of delegates if and only if the broadcast key's conditional set W is close to the conditional set W' of the ciphertext. With the FC-PBRE scheme, the plaintext data is not disclosed and cannot be learnt by the proxy. The paper is structured as follows. First we introduce the security definition against chosen-ciphertext attacks for FC-PBRE. Second, our efficient fuzzy-conditional proxy broadcast re-encryption scheme is presented. Finally, we prove that our FC-PBRE scheme is CCA-secure in the random oracle model based on the Decisional nBDHE assumption.

2. Related Work

Proxy re-encryption [9] can directly transfer encrypted data of user Alice stored in nontrusted third-party cloud under the authorization of Alice into user Bob ciphertext, so as to achieve data sharing of Alice and Bob. But the third party can transform all Alice's ciphertext into Bobs ciphertext in PRE scheme, but in many applications, Alice hopes that the third party can only transform some specific conditions of ciphertext instead of all ciphertext. In a C-PRE scheme [11], only the one ciphertext from Alice that meets certain specific conditions can be converted by third parties into a Bob's ciphertext. Therefore, with conditional proxy re-encryption, Alice can determine which ciphertext for third parties to re-encrypt; thus a flexible control of the ciphertext can be observed. Weng et al. [11] proposed conditional proxy re-encryption scheme and they proved that it is chosen-ciphertext attack secure in the random oracle model. Similarly, Tang [13] proposed a type based proxy re-encryption scheme, a secure proxy re-encryption with keyword search scheme, and it is proven secure in the random oracle model [14]. On top of the keyword search scheme, an anonymous conditional proxy re-encryption with keyword search scheme was proposed by Fang et al. [15]. On the paper, they also proved that their scheme is chosen-ciphertext secure. Fang et al. [16] presented a fuzzy-conditional proxy re-encryption scheme and proved the security in the random oracle model. The scheme can support fuzzy matching between multiple keywords; that is, only some key words in the re-encryption key satisfy the matching, and the third party can complete the re-encryption.

Without the random oracle, a conditional proxy re-encryption scheme [17] was proposed to support a more fine-grained access strategy under the standard model. Reference [18] designed an identity based proxy re-encryption mechanism for multiple hop (Multihop) under the standard model. In this scheme, the re-encrypt ciphertext can still be re-encrypted repeatedly, and the length of ciphertext does not increase with the number of re-encryptions. So the length of the ciphertext is constant.

With regard to anonymity, Ateniese et al. [19] came out with the idea of anonymous proxy re-encryption and proved that their scheme is chosen plaintext attack security under the standard model. With anonymous proxy re-encryption, an attacker cannot obtain user's identity from the key. Later, a new scheme was proposed to achieve the security of chosen-ciphertext attack with the random oracle [20]. Subsequently, Shao et al. improved the scheme by using the standard model for the security proof [21]. Followed by Shao et al.'s work, a security model is enhanced for anonymous proxy re-encryption [22]. In the security model of [22], it allows attackers to get re-encrypted queries directly, instead of obtaining re-encrypted query by acquiring re-encryption key query. Shao et al. [23] proposed an anonymous ID based proxy re-encryption to extend anonymous proxy re-encryption to identity based anonymous proxy re-encryption. Their scheme is proven secure in the random oracle model. An anonymous identity based multiuser identity based proxy re-encryption scheme was proven CCA-secure in the standard

model [18]. The same literature also analyzed its application in privacy protection and data sharing in big data storage system. The above schemes are about proxy anonymous encryption with user identity anonymity. A keyword anonymity conditional proxy re-encryption scheme [24] was demonstrated to achieve the anonymity of conditions.

3. Preliminaries

3.1. Bilinear Map. G and G_T denote two multiplicative cyclic groups with the same prime order p . g is a generator of group G . A bilinear pairing is a bilinear map $e : G \times G \rightarrow G_T$ with the following properties:

- (1) $e(U^a, V^b) = e(U, V)^{ab}$ for all $a, b \xleftarrow{R} Z_p^*$ and $U, V \in G$.
- (2) $e(U, U) \neq 1$.
- (3) $e(U, V)$ can be computed in polynomial time for all $U, V \in G$.

3.2. The n -BDHE Assumption. If p is a prime, let Z_p denote the set $\{0, 1, \dots, p-1\}$ and Z_p^* denote the set $\{1, 2, \dots, p-1\}$. Let $e : G \times G \rightarrow G_T$ be a bilinear map. Given $2n+1$ elements

$$(\mu, \nu, \nu^\gamma, \nu^{\gamma^2}, \dots, \nu^{\gamma^n}, \nu^{\gamma^{n+2}}, \dots, \nu^{\gamma^{2n}}) \in G^{2n+1} \quad (1)$$

and an element $T \in G_T$, the adversary's task is to decide if $T \stackrel{?}{=} e(\mu, \nu)^{\gamma^{n+1}}$.

Denote ν^{γ^i} as ν_i , and define the advantage of an adversary $\mathcal{A}$ as

$$\begin{aligned} & Adv_{G, \mathcal{A}}^{n-BDHE} \\ &= \left| \Pr [\mathcal{A}(\mu, \nu, \nu_1, \dots, \nu_n, \nu_{n+2}, \dots, \nu_{2n}, e(\nu_{n+1}, \mu)) = 1] - \Pr [\mathcal{A}(\mu, \nu, \nu_1, \dots, \nu_n, \nu_{n+2}, \dots, \nu_{2n}, T) = 1] \right| \quad (2) \end{aligned}$$

where $\nu, \mu \in G$, $\gamma \in Z_p^*$, and $T \in G_T$ are randomly chosen. We conclude that the n -BDHE assumption relative to (G, G_T) holds [25], if $Adv_{G, \mathcal{A}}^{n-BDHE}$ is negligible for all probability polynomial time (PPT) adversary $\mathcal{A}$.

4. FC-PBRE Model and Security Notion

Two security definitions for fuzzy-conditional proxy broadcast re-encryption as well as its model are introduced in this section.

Definition 1 (FC-PBRE). A (single-use) proxy broadcast re-encryption scheme runs the following algorithms:

- (i) *Setup*(λ, n, d): in the setup step, for the input, λ is the security parameter, n is the maximum allowed number of users, and d is the threshold. The system at this step generates a public key PK and a master secret key MK .
- (ii) *KenGen*(PK, MK, i): given the public key PK , the master key MK , and a user i , the system generates secret key sk_i for the user i .

- (iii) $Encrypt(PK, S, m, W)$: the $Encrypt()$ algorithm takes the public key PK , a user sets $S \subseteq \{1, 2, \dots, n\}$, a message m , and a set of keywords W as the input, it then outputs the ciphertext C for the user set S with condition set W .
- (iv) $ReKGen(PK, sk_i, S', W')$: on inputting PK , the user's private key sk_i , a set of users ($S' \subseteq \{1, 2, \dots, n\}$) and a keywords set W' , outputs the re-encryption key $rkey_{i,S',W'}$.
- (v) $REnc(PK, rkey_{i,S',W'}, i, S, S', C)$: with the public key PK , a re-encryption key $rkey_{i,S',W'}$, the user i , two different user sets S and S' , and the original ciphertext C , this algorithm computes the re-encrypted ciphertext C_R . This step requires $|W \cap W'| \geq d$, where d is the threshold and if the condition cannot be met, an error symbol $\perp$ will be output instead.
- (vi) $DecryptL2(PK, sk_i, i, S, C)$: $DecryptL2$ represents the decryption algorithm for the original ciphertext. It takes the public key PK , user i , the private key sk_i of the user i , a set of users S , and ciphertext C for user set S , and it outputs the plaintext m .
- (vii) $DecryptL1(PK, sk_j, i, j, S, S', C_R)$: $DecryptL1$ represents the re-encrypted ciphertext decryption algorithm. It outputs the plaintext m from the input of a public key, a user's private key sk_j , users i, j , user sets S, S' , and a re-encrypted C_R . If the algorithm aborts, it outputs $\perp$.

Correctness. The definition of the correctness of a FC-PBRE scheme is as follows; given two user sets S, S' , condition sets W, W' , $C = Encrypt(PK, S, m, W)$, and $rkey_{i,S',W'} = ReKGen(PK, sk_i, S', W')$, then $C_R = REnc(PK, rkey_{i,S',W'}, i, S, S', C)$, if $|W \cap W'| \geq d$

$$\begin{aligned} Pr[DecryptL2(PK, sk_i, i, S, C) = m] &= 1, \text{ if } i \in S; \\ Pr[DecryptL1(PK, sk_j, i, j, S, S', C_R) = m] &= 1, \text{ if } j \in S'. \end{aligned}$$

The game-based model is used to define the security for the FC-PBRE scheme. Similar to a security model from [25], the CCA security of the FC-PBRE scheme is considered in the selective-set model. In such a model, the adversary is supposed to commit ahead the challenge user set S^* and the condition set W^* .

Definition 2 (IND-set-CCA game). The following lists two games between one adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Game 1 is to consider the security of the original ciphertexts. We define the IND-OR-CCA game as follows:

- (1) Init. In the initial phase, the adversary $\mathcal{A}$ selects a target users set $S^* \subseteq \{1, 2, \dots, n\}$ and the condition set $W^* = \{\omega_1^*, \omega_2^*, \dots, \omega_n^*\}$.
- (2) Setup. At this stage of the game, the challenger $\mathcal{C}$ runs $Setup(n)$, and as a result, he can generate the public key PK and the master key MK . Then he gives PK to $\mathcal{A}$.
- (3) Query phase 1. $\mathcal{A}$ makes the following queries:

- (a) $Extract(i)$: the challenger runs $sk_i = KeyGen(PK, MK, i)$ and returns sk_i to $\mathcal{A}$.
- (b) $ReKGen(i, S', W')$: $\mathcal{C}$ runs $rkey_{i,S',W'} = ReKGen(PK, sk_i, S', W')$ and $sk_i = KeyGen(PK, MK, i)$ and returns $rkey_{i,S',W'}$ to $\mathcal{A}$.
- (c) $REnc(i, S, S', C)$: $\mathcal{C}$ runs $REnc(PK, rkey_{i,S',W'}, i, S, S', C)$, where $rkey_{i,S',W'} = ReKGen(PK, sk_i, S', W')$ and $sk_i = KeyGen(PK, MK, i)$, and returns the output to $\mathcal{A}$.
- (d) $DecryptL2(i, S, C)$: $\mathcal{C}$ runs $DecryptL2(PK, sk_i, i, S, C)$, where $sk_i = KeyGen(PK, MK, i)$, and returns the output to $\mathcal{A}$.
- (e) $DecryptL1(i, j, S, S', C_R)$: $\mathcal{C}$ runs $DecryptL1(PK, sk_j, i, j, S, S', C_R)$, where $sk_j = KeyGen(PK, MK, j)$, and returns the output to $\mathcal{A}$.

$\mathcal{A}$ has to follow the restrictions in this phase. First, $\mathcal{A}$ cannot make $Extract(i)$ for any $i \in S^*$; second $\mathcal{A}$ cannot make $ReKGen(i, S', W')$ and $Extract(j)$, if $i \in S^*$, $j \in S'$, and $|W' \cap W^*| \geq d$.

- (4) Challenge. As soon as $\mathcal{A}$ finishes Query phase 1, $\mathcal{A}$ computes two equal length messages (m_0, m_1) . $\mathcal{C}$ selects a bit $b \in \{0, 1\}$ and sets the challenge ciphertext to be $C^* = Encrypt(PK, m_b, S^*, W^*)$. C^* is then sent to $\mathcal{A}$.
- (5) Query phase 2. In addition to the restrictions in phase 1, $\mathcal{A}$ continues making queries with the following extra restrictions: $\mathcal{A}$ is not allowed to run $REnc(i, S^*, S', C^*)$ and $Extract(j)$ if $i \in S^*$ and $j \in S'$;
 - (a) $\mathcal{A}$ is not allowed to run $DecryptL2(i, S^*, C^*)$ for any $i \in S^*$;
 - (b) $\mathcal{A}$ is not allowed to run $DecryptL1(i, j, S^*, S', C_R)$ if $i \in S^*$, $j \in S'$ and $C_R = REnc(i, S^*, S', C^*)$.
- (6) Guess. $\mathcal{A}$ makes the guess for b' . The adversary wins the game if $b' = b$.

The above adversary $\mathcal{A}$ is referred to as an IND-OR-CCA adversary. The advantage is defined as

$$Adv_{\mathcal{A},n}^{Game_1} = \left| Pr[b' = b] - \frac{1}{2} \right|. \quad (3)$$

Game 2. IND-Re-CCA game is used for the indistinguishability of the re-encrypted ciphertext.

- (1) Init. To begin the game, an adversary $\mathcal{A}$ selects a target users set $S^* \subseteq \{1, 2, \dots, n\}$ and a condition set $W^* = \{\omega_1^*, \omega_2^*, \dots, \omega_n^*\}$.
- (2) Setup. The challenger $\mathcal{C}$ generates the public key PK and master key MK by running $Setup(n)$. Then $\mathcal{C}$ gives PK to $\mathcal{A}$.
- (3) Query phase 1. The adversary $\mathcal{A}$ makes the following queries:

- (a) *Extract*(i): $\mathcal{C}$ gets sk_i from $KeyGen(PK, MK, i)$. Then it returns it to $\mathcal{A}$. Note that $\mathcal{A}$ cannot make *Extract*(i) for any user $i \in S^*$;
 - (b) *ReKGen*(i, S', W'): $\mathcal{C}$ runs $rkey_{i,S',W'} = ReKGen(PK, sk_i, S', W')$, where $sk_i = KeyGen(PK, MK, i)$, and returns $rkey_{i,S',W'}$ to $\mathcal{A}$.
 - (c) *DecryptL1*(i, j, S, S', C_R): $\mathcal{C}$ runs $DecryptL1(PK, sk_j, i, j, S, S', C_R)$, where $sk_j = KeyGen(PK, msk, j)$, and returns the result to $\mathcal{A}$.
- (4) Challenge. Once $\mathcal{A}$ finishes Query phase 1, $\mathcal{A}$ outputs two equal length messages (m_0, m_1) . Challenger $\mathcal{C}$ chooses a bit $b \in \{0, 1\}$ and sets the challenge ciphertext to be $C^* = REnc(PK, rkey_{i,S^*,W^*}, i, S, S^*, C)$, where $i \in S, i \notin S^*$ and $C = Encrypt(PK, m_b, S, W^*)$. C^* is returned to the adversary $\mathcal{A}$.
- (5) Query phase 2. $\mathcal{A}$ continues making queries but is subject to the following restrictions:
- (a) $\mathcal{A}$ cannot make *Extract*(i) for any $i \in S^*$;
 - (b) $\mathcal{A}$ cannot make *DecryptL1*(i, j, S, S^*, C^*), if $i \in S$ and $j \in S^*$.
- (6) Guess. $\mathcal{A}$ makes the guess b' . The adversary wins the game if $b' = b$.

The above adversary $\mathcal{A}$ is referred as an IND-Re-CCA adversary. Its advantage is defined as

$$Adv_{\mathcal{A},n}^{Game_2} = \left| Pr[b' = b] - \frac{1}{2} \right|. \quad (4)$$

A fuzzy-conditional proxy broadcast re-encryption scheme is called IND-Set-CCA-secure if for all PPT adversary $\mathcal{A}$, $Adv_{\mathcal{A},n}^{Game_1}$, and $Adv_{\mathcal{A},n}^{Game_2}$ are negligible.

5. The Proposed FC-PBRE Scheme

5.1. *Our Construction.* A Lagrange coefficient $\Delta_{\omega,S}(x)$ is defined for $\omega \in Z_p$ and a set S , of elements in Z_p :

$$\Delta_{\omega,S}(x) = \prod_{i \in S, i \neq \omega} \frac{x - i}{\omega - i} \quad (5)$$

Our FC-PBRE scheme contains the following algorithms:

- (i) *Setup*(λ, n, d): Let $M = \{0, 1\}^k$ denote the message space. The algorithm randomly selects $\alpha, \gamma \in Z_p$ and $Z \in G$ and computes $v_i = g^{\alpha^i}$ for $i = 1, 2, \dots, n, n+2, \dots, 2n$. Let $Hash_1 : \{0, 1\}^k \times G_T \rightarrow Z_p^*, Hash_2 : G_T \rightarrow \{0, 1\}^k, Hash_3 : G_T \times G \times G \times G \times \{0, 1\}^k \rightarrow G, Hash_4 : Z_p^* \rightarrow G$, and $Hash_5 : \{0, 1\}^k \rightarrow Z_p^*$ be collision-resistant hash functions. It computes $v =$

g^γ . *Setup*(λ, n, d) outputs the public key PK and the master key MK for the cloud server as follows:

$$PK = (v, v_1, \dots, v_n, v_{n+2}, v_{2n}, v, Z, Hash_1, Hash_2, Hash_3, Hash_4), \quad (6)$$

$$MK = \gamma.$$

- (ii) *KeyGen*(PK, MK, i): The secret key for a cloud user i is generated in this phase, where

$$sk_i = v_i^\gamma. \quad (7)$$

- (iii) *Encrypt*(PK, S, m, W): To encrypt a plaintext $m \in M$ under the set $S \subseteq \{1, 2, \dots, n\}$ with condition W , it randomly picks a $R \in G_T$, computes $t = Hash_1(m, R)$, and outputs the results of the ciphertext: $C = (C_1, C_2, C_3, C_4, C_5, C_6)$.

$$C_1 = R \cdot e(v_1, v_n)^t,$$

$$C_2 = v^t,$$

$$C_3 = \left(v \cdot \prod_{j \in S} v_{n+1-j} \right)^t, \quad (8)$$

$$C_4 = (C_\omega = Hash_4(\omega)^t)_{\omega \in W},$$

$$C_5 = m \oplus Hash_2(R),$$

$$C_6 = Hash_3(C_1, C_2, C_3, C_4, C_5)^t.$$

- (iv) *ReKGen*(PK, sk_i, S', W'): On inputting $sk_i = v_i^\gamma, S' \in \{1, 2, \dots, n\}$ and W' , it randomly selects $\sigma \in \{0, 1\}^k$ and a polynomial $f(x)$ with $d-1$ degree, where $f(0) = Hash_5(\sigma)$. For each $\omega \in W'$, it selects a random value r_ω and computes

$$a = (a_\omega = sk_i \cdot Z^{q(\omega)} Hash_4(\omega)^{r_\omega})_{\omega \in W'}, \quad (9)$$

$$b = (b_\omega = v^{r_\omega})_{\omega \in W'}.$$

It chooses random value $s \in Z_p^*, R' \in G_2$, computes $t' = Hash_1(\sigma, R')$, and sets

$$rkey_1 = R' \cdot e(v_1, v_n)^{t'},$$

$$rkey_2 = v^{t'},$$

$$rkey_3 = \left(v \cdot \prod_{j \in S'} v_{n+1-j} \right)^{t'}, \quad (10)$$

$$rkey_4 = \sigma \oplus Hash_2(R'),$$

$$rkey_5 = Hash_3(rkey_1, rkey_2, rkey_3, rkey_4)^{t'}.$$

It outputs the re-encryption key $rk_{i,S',W'} = (a, b, rkey_1, rkey_2, rkey_3, rkey_4, rkey_5)$.

- (v) $REnc(PK, rkey_{i,S',W'}, i, S, S', C)$: On inputting a re-encryption key $rkey_{i,S',W'} = (a, b, rkey_1, rkey_2, rkey_3, rkey_4, rkey_5)$ and a ciphertext $C = (C_1, C_2, C_3, C_4, C_5, C_6)$, it verifies whether the following hold:

$$e\left(C_2, v \cdot \prod_{j \in S} \nu_{n+1-j}\right) \stackrel{?}{=} e(\nu, C_3), \quad (11)$$

$$e(C_2, Hash_4(\omega))_{\omega \in S} \stackrel{?}{=} e(\nu, C_\omega)_{\omega \in S}, \quad (12)$$

$$e(C_2, Hash_3(C_1, C_2, C_3, C_4, C_5)) \stackrel{?}{=} e(\nu, C_6). \quad (13)$$

If not, it outputs $\perp$. Otherwise, it randomly selects a d element set $F \subseteq W \cap W'$. If such a set F cannot be found, the algorithm outputs $\perp$, or else it computes

$$\widetilde{C}_1 = C_1 \cdot \prod_{\omega \in F} \left(\frac{e(rkey_0 \cdot \prod_{j \in S, j \neq i} \nu_{n+1-j+i}, C_2)}{e(\nu_i, C_3) \cdot e(b_\omega, C_\omega)} \right)^{\Delta_{\omega,F}(0)} \quad (14)$$

Then the re-encrypted ciphertext is $C_R = (\widetilde{C}_1, C_2, C_5, rkey_1, rkey_2, rkey_3, rkey_4, rkey_5)$.

- (vi) $DecryptL2(PK, sk_i, i, S, C)$: It takes a secret key sk_i and a ciphertext $C = (C_1, C_2, C_3, C_4, C_5, C_6)$ as the inputs, it

- (1) verifies if the three equations (11)–(13) hold. If not, it aborts and outputs the error $\perp$ or, else,
- (2) computes $R = C_1 \cdot e(sk_i \cdot \prod_{j \in S, j \neq i} \nu_{n+1-j+i}, C_2) / e(\nu_i, C_3)$, $m = C_5 \oplus Hash_2(R)$, and $t = Hash_1(m, R)$ and verifies whether

$$C_2 = \nu^t, \quad (15)$$

$$C_3 = \left(v \cdot \prod_{j \in S} \nu_{n+1-j} \right)^t, \quad (16)$$

$$C_6 = Hash_3(C_1, C_2, C_3, C_4, C_5)^t \quad (17)$$

hold. It returns m or else returns $\perp$.

- (vii) $DecryptL1(PK, sk_j, i, j, S, S', C_R)$: On inputting a secret key sk_j and a re-encrypted ciphertext $C_R = (\widetilde{C}_1, C_2, C_5, rkey_1, rkey_2, rkey_3, rkey_4, rkey_5)$, it proceeds as follows:

First verifying whether all of the equations hold:

$$e\left(rkey_2, v \cdot \prod_{j \in S} \nu_{n+1-j}\right) \stackrel{?}{=} e(\nu, rkey_3), \quad (18)$$

$$e(rkey_2, Hash_3(rkey_1, \dots, rkey_4)) \stackrel{?}{=} e(\nu, rkey_5). \quad (19)$$

If they do not hold, it outputs $\perp$. Otherwise, it computes

$$R' = rkey_1 \cdot \frac{e(sk_j \cdot \prod_{l \in S', l \neq j} \nu_{n+1-l+j}, rkey_2)}{e(g_j, rk_3)},$$

$$\sigma = rkey_5 \oplus Hash_2(R'), \quad (20)$$

$$g^s = \frac{rkey_4}{Hash_4(\sigma)},$$

$$t' = Hash_1(\sigma, R').$$

It verifies whether

$$rkey_2 = \nu^{t'},$$

$$rkey_3 = \left(v \cdot \prod_{l \in S'} \nu_{n+1-l} \right)^{t'}, \quad (21)$$

$$rkey_5 = Hash_3(rkey_1, rkey_2, rkey_3, rkey_4)^{t'}$$

hold. If not, it returns $\perp$, or else

it computes $R = \widetilde{C}_1 / e(C_2, Z^{Hash_5(\sigma)})$, $m = C_5 \oplus Hash_2(R)$, and $t = Hash_1(m, R)$. It verifies whether $C_2 = \nu^t$ holds. If it holds, it returns m or else it returns $\perp$.

5.2. Security Proof. In section, we prove that our scheme is IND-Set-CCA-secure in the ROM.

Theorem 3. *If $Hash_1, Hash_2, Hash_3, Hash_4, Hash_5$ are target collision-resistant hash functions and the Decisional nBDHE assumption holds, the above scheme then is IND-Set-CCA-secure in the ROM.*

Lemma 4. *If an IND-OR-CCA is attacker $\mathcal{A}$ that can successfully attack FC-PBRE, then we are able to construct a simulator $\mathcal{S}$ that can solve Decisional nBDHE assumption.*

Proof. The simulator $\mathcal{S}$ is provided a Decisional nBDHE instance $(\mu, \nu, \nu_1, \dots, \nu_n, \nu_{n+2}, \dots, \nu_{2n}, T)$ and has to decide whether $T = e(\nu_{n+1}, \mu)$ from a random value. The simulator $\mathcal{S}$ controls the random oracles (RO) $Hash_1, Hash_2, Hash_3, Hash_4, Hash_5$ as follows:

$\mathcal{S}$ searches $Hash_1$ for (m, R, t) . If $Hash_1$ list already exists, then $\mathcal{S}$ sends t to $\mathcal{A}$. Else, $\mathcal{S}$ randomly selects $t \in Z_p^*$, sends s to $\mathcal{A}$, and puts (m, R, t) in $Hash_1$ list.

$\mathcal{S}$ searches $Hash_2$ for (R, κ) . If $Hash_2$ list already exists, then $\mathcal{S}$ sends κ to $\mathcal{A}$. Else, $\mathcal{S}$ randomly selects $\kappa \in \{0, 1\}^k$, sends κ to $\mathcal{A}$, and puts (R, κ) to $Hash_2$.

$\mathcal{S}$ searches $Hash_3$ for $(C_1, C_2, C_3, C_4, C_5, \phi, \varphi)$. If $Hash_3$ list already exists, then $\mathcal{S}$ sends ψ to $\mathcal{A}$. Else, $\mathcal{S}$ randomly selects $\phi \in Z_p^*$, generates $\psi = \nu^\phi$, sends ψ to $\mathcal{A}$, and puts $(C_1, C_2, C_3, C_4, C_5, \phi, \varphi)$ to $Hash_3$ list.

$\mathcal{S}$ searches $Hash_4$ for (ω, τ, ω) . If $Hash_4$ list already exists, then $\mathcal{S}$ sends ω to $\mathcal{A}$. Else, $\mathcal{S}$ randomly selects $\tau \in Z_p^*$ and sets $\omega = \nu^\tau$, sends ω to $\mathcal{A}$, and puts (ω, τ, ω) to $Hash_4$ list.

$\mathcal{S}$ searches $Hash_5$ for (σ, ρ) . If $Hash_5$ list already exists, then $\mathcal{S}$ sends ρ to $\mathcal{A}$. Else, $\mathcal{S}$ randomly selects $\rho \in Z_p^*$, sends ρ to $\mathcal{A}$, and adds (σ, ρ) to $Hash_5$ list.

$\mathcal{S}$ then keeps the lists as follows:

(i) Key^{List} stores the tuples (β, i, sk_i) ;

(ii) $ReKey^{List}$ stores the tuples $(\beta_1, i, S', W', rkey_{i,S',W'}, \sigma, R, FlagOne)$, which maintains the results of query $ReKGen(sk_i, S', W')$. If $FlagOne = 1$, then it is a valid re-encryption key, or else it is random.

(iii) $REnc^{List}$ stores the tuples $(i, S, S', C, C_R, FlagTwo)$, which maintains query $REnc(i, S, S', C)$. Note that if $FlagTwo = 1$, then a re-encryption key which is used to generate the re-encrypted ciphertext should be valid, or else the re-encrypted ciphertext is invalid and randomly generated.

- (1) **Init.** The attacker selects a challenge user set $S^* \subseteq \{1, 2, \dots, n\}$ and challenge conditional set $W^* = \{\omega_1^*, \omega_2^*, \dots, \omega_n^*\}$.
- (2) **Setup.** The simulator $\mathcal{S}$ randomly selects values $\omega, \in Z_p$ and $Z \in G$ and sets

$$v = v^\omega \cdot \left(\prod_{j \in S^*} v_{n+1-j} \right)^{-1} \triangleq v^\gamma. \quad (22)$$

$\mathcal{S}$ sets the public key as

$$PK = (v, \gamma, v_1, \dots, v_n, v_{n+2}, \dots, v_{2n}, Z, Hash_1, Hash_2, Hash_3, Hash_4, Hash_5) \quad (23)$$

and $sk = \gamma$. $\mathcal{S}$ sends PK to the attacker $\mathcal{A}$.

- (3) **Query phase 1.** $\mathcal{A}$ does some of the columns of the query, and the response of $\mathcal{S}$ is as follows:

(i) $Extract(i)$: If $i \in S^*$, then $\mathcal{S}$ aborts. Else the simulator $\mathcal{S}$ first searches Key^{List} , and if (β, i, sk_i) are in Key^{List} , then it outputs sk_i . Otherwise, $\mathcal{S}$ throws a biased coin β ($Pr[\beta = 1] = \delta$ for some δ).

(a) When $\beta = 0$, $\mathcal{S}$ aborts and outputs a random bit.

(b) When $\beta = 1$, $\mathcal{S}$ calculates $sk_i = v_i^\mu \cdot \left(\prod_{j \in S^*} v_{n+1-j+i} \right)^{-1}$. Note that we have

$$sk_i = v_i^\omega \cdot \left(\prod_{j \in S^*} v_{n+1-j+i} \right)^{-1} = \left(v^\omega \cdot \left(\prod_{j \in S^*} v_{n+1-j} \right)^{-1} \right)^{\alpha^i} = v^{\alpha^i} = v_i^\gamma. \quad (24)$$

(ii) $ReKGen(i, S', W')$: the simulator $\mathcal{S}$ verifies whether tuple $(*, j, sk_j)$ is in Key^{List} ($i \in S^*$, $j \in S'$, and $|W' \cap W^*| \geq d$). If not, then $\mathcal{S}$ aborts.

Otherwise $\mathcal{S}$ searches whether there is a tuple $(*, i, S', W', rkey_{i,S',W'}, \sigma, R, *)$ in $ReKey^{List}$. If yes, $\mathcal{S}$ outputs $rkey_{i,S',W'}$. Otherwise, the simulator $\mathcal{S}$ proceeds:

- (a) If $(1, i, sk_i)$ exists in Key^{List} , as in the real algorithm of the scheme, the simulator $\mathcal{S}$ generates the re-encryption key $rkey_{i,S',W'}$ from $ReKGen$ by using the secret key sk_i . $\mathcal{S}$ returns $rkey_{i,S',W'}$ to $\mathcal{A}$. Then it puts $(*, i, S', W', rkey_{i,S',W'}, \sigma, R, 1)$ to $ReKey^{List}$, where σ, R are randomly selected.
- (b) Otherwise, $\mathcal{S}$ throws a biased coin β . If $\beta = 1$, $\mathcal{S}$ queries the $Extract(i)$ oracle to get sk_i and, then, generates $rkey_{i,S',W'}$ from $ReKGen$. $\mathcal{S}$ returns the re-encryption key $rkey_{i,S',W'}$ to $\mathcal{A}$. Then it adds $(1, i, sk_i)$ and $(*, i, S', W', rkey_{i,S',W'}, \sigma, R, 1)$ to Key^{List} and $ReKey^{List}$, respectively. If $\beta = 0$, $\mathcal{S}$ sets $\{a = (a_\omega = \rho_\omega), b = (b_\omega = \rho'_\omega) : \omega \in W'\}$ for randomly chosen $\rho_\omega, \rho'_\omega \in G$. Then $\mathcal{S}$ constructs $rkey_1, rkey_2, rkey_3, rkey_4, rkey_5$ to encrypted random σ, R as the real environment. The simulator $\mathcal{S}$ sends the re-encryption key to $\mathcal{A}$. Then it adds $(*, i, S', W', rkey_{i,S',W'}, \sigma, R, 0)$ to $ReKey^{List}$.
- (iii) $REnc(i, S, S', C)$: $\mathcal{S}$ searches $(i, S, S', C, C_R, *)$ in $REnc^{List}$. If yes, $\mathcal{S}$ returns C_R . Else, it is dealt with in the below:
 - (a) If $(*, i, S', rkey_{i,S',W'}, \sigma, R, *)$ in $ReKey^{List}$, as the real environment, $\mathcal{S}$ uses $rkey_{i,S',W'}$ to construct C_R by $REnc$ and sends C_R to $\mathcal{A}$. Then it adds $(i, S, S', C, C_R, *)$ to $REnc^{List}$. Here we need $C = Encrypt(PK, S, m, W)$ and $|W \cap W'| \geq d$.
 - (b) Otherwise, $\mathcal{S}$ first makes $ReKGen(i, S')$ query to retrieve the re-encryption key $rkey_{i,S',W'}$. Next, $\mathcal{S}$ constructs C_R . Then it puts $(i, S, S', C, C_R, *)$ to $REnc^{List}$.
- (iv) $DecryptL2(i, S, C)$: $\mathcal{S}$ tests (11)-(13). If the test does not pass, it aborts with an error $\perp$. Else
 - (a) If $(1, i, sk_i)$ already exists in Key^{List} , $\mathcal{S}$ recovers m by using sk_i .
 - (b) Otherwise, $\mathcal{S}$ queries $Extract(i)$ to obtain sk_i and $\mathcal{S}$ recovers m by using sk_i .
- (v) $DecryptL1(i, j, S, S', C_R)$: $\mathcal{S}$ checks (18)-(19). If one of the equations cannot be validated, it aborts and outputs $\perp$. Otherwise, $\mathcal{S}$ proceeds:
 - (a) If $(1, j, sk_j)$ already exists in Key^{List} , $\mathcal{S}$ recovers m by using sk_j .
 - (b) Otherwise, $\mathcal{S}$ queries $Extract(j)$ to get sk_j and $\mathcal{S}$ recovers m by using sk_j .

- (4) **Challenge.** Once the attacker $\mathcal{A}$ decides that Phase 1 is finished, it outputs two messages (m_0, m_1) . And $\mathcal{S}$

randomly selects $b \in \{0, 1\}$, $R^* \in G_T$. Let $\mu = g^{t^*}$ for some randomly chosen t^* . $\mathcal{S}$ computes

$$\begin{aligned} C_1^* &= R^* \cdot T, \\ C_2^* &= \mu = v^{t^*}, \\ C_3^* &= \mu^\omega = v^{\omega t^*} \\ &= \left(v^\omega \cdot \left(\prod_{j \in S^*} v_{n+1-j} \right)^{-1} \left(\prod_{j \in S^*} v_{n+1-j} \right) \right)^{t^*} \\ &= \left(v \prod_{j \in S^*} v_{n+1-j} \right)^{t^*}, \end{aligned} \quad (25)$$

$$C_4^* = \left(C_\omega = \mu^\tau = (v^\tau)^{t^*} = \text{Hash}_4(\omega)^{t^*} \right)_{\omega \in W^*},$$

$$C_5^* = m_b \oplus \text{Hash}_2(R^*),$$

$$\phi^* = \text{Hash}_3(C_1^*, C_2^*, C_3^*, C_4^*, C_5^*),$$

$$C_6^* = \mu^{\phi^*} = \text{Hash}_3(C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)^{t^*}$$

where $\mathcal{S}$ queries $(C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ to RO Hash_3 for entry $(C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, \phi^*, \phi^*)$. $\mathcal{S}$ sends $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, C_6^*)$ to $\mathcal{A}$. If $T = e(\gamma_{n+1}, \mu)$, we have $C_1^* = R^* \cdot T = R^* \cdot e(\gamma_{n+1}, \mu)^{t^*}$. Hence the challenge ciphertext C^* is a valid one. Otherwise, C^* is independent of b if T is a random value, in the attacker's view.

- (5) **Query phase 2.** $\mathcal{A}$ makes queries continuously.
- (6) **Guess.** $\mathcal{A}$ generates the guess bit b' , if $b' = b$, and then outputs 1, which means that $T = e(\gamma_{n+1}, \mu)$; else outputs 0. In this case, T is a random value in G_2 .

□

Probability analysis. When $\mathcal{S}'$'s aborting happens, we define the game as *Abort*. In the case $\mathcal{S}$ does not abort, $\mathcal{A}$ does not know the different between the game and the actual scheme. Let q denote the total number of all queries; we have $\Pr[\neg \text{Abort}] \geq \delta^q \cdot ((p-1)/p)^q \triangleq \xi^q \geq \xi^q(1-\xi)$, which is maximized at $\delta_{\text{opt}} = q/(1+q)$. Using δ_{opt} , the probability $\Pr[\neg \text{Abort}]$ is at least $1/\hat{e}(1+q)$, where $\hat{e}$ is the base of the nature logarithm. Therefore, we have

$$\begin{aligned} \epsilon' &\geq \frac{\epsilon}{\hat{e}(1+q)} - \text{Adv}_{\text{Hash}_1, \mathcal{A}}^{\text{TCR}} - \text{Adv}_{\text{Hash}_2, \mathcal{A}}^{\text{TCR}} \\ &\quad - \text{Adv}_{\text{Hash}_3, \mathcal{A}}^{\text{TCR}} - \text{Adv}_{\text{Hash}_4, \mathcal{A}}^{\text{TCR}} - \text{Adv}_{\text{Hash}_5, \mathcal{A}}^{\text{TCR}}. \end{aligned} \quad (26)$$

With this we complete the proof of Lemma 4.

Lemma 5. *If an IND-Re-CCA attacker can successfully attack our scheme, then a simulator $\mathcal{S}$ that can be constructed to solve the Decisional nBDHE assumption.*

Proof. We use the same game construction from the proof of Lemma 4 and modify the challenge phase to prove Lemma 5. To construct the challenge re-encrypt ciphertext, $\mathcal{S}$ randomly selects $b \in \{0, 1\}$, $R^* \in G_T$. Let $h = g^{t^*}$ for the randomly selected t^* . $\mathcal{S}$ then computes

$$\begin{aligned} \widetilde{C}_1^* &= R^* \cdot e(\mu, Z)^{\text{Hash}_5(\sigma)}, \\ C_2^* &= \mu = v^{t^*}, \\ C_5^* &= m_b \oplus \text{Hash}_2(R^*), \end{aligned} \quad (27)$$

$\mathcal{S}$ then constructs $rkey_1^*, rkey_2^*, rkey_3^*, rkey_4^*, rkey_5^*$ in the same way as in game 1.

The proofs of Lemmas 4 and 5 conclude that we have proven Theorem 3. □

6. Comparison

We use schemes from [24, 26] as baselines as [26] achieves the same security and [24] supports the same fuzzy property with our scheme. In the implementation, we selected two most efficient schemes to do experimental comparisons. Our experimental environment is as follows: Core i7 Processor (6M Cache, 3.40 GHz) with a Linux operating system. In the implementation, the proxy re-encrypted the ciphertext of the delegator to 20 different ciphertext for the delegate. The average value of the execution time of 50 experiments is used to eliminate the errors. Table 1 lists the comparison of the performance of the schemes. Although *ReKGen*, *Enc*, *REnc*, *DecryptL2*, and *DecryptL1* time is a little greater than [24, 26] as our scheme needs to support the property of broadcast, the time overhead of the re-encryption algorithm is much slower than our scheme. This is due to the fact that the proxy is only required to run the re-encryption algorithm once.

7. Application

With the proposed FC-PBRE scheme, we illustrate an example of the scheme's possible application and show how FC-PBRE scheme protects the confidentiality and privacy of multimedia data in the Internet of Things.

7.1. Application of Security in Internet of Things. With the development of wireless networks and the Internet of Things, a large number of wireless devices (WiFi cameras, wireless sensors, etc.) are deployed; they are collecting and analyzing multimedia data at all times. Many applications require these wireless devices to share data, and insecure data sharing before wireless devices can easily lead to data leakage. Usually a normal user accesses a wireless device in three ways: first, through direct connection with the wireless device; second, through the gateway; third, through the cloud platform. Because many wireless devices need to be localized before they start to work, they usually have backend configuration interfaces (WEB, PC) and are offline at some point during their work, so it is more convenient for an attacker to access multimedia data directly

TABLE 1: Comparison with Weng et al. [26] and Fang et al. [24].

Scheme	ReKGen(ms)	Enc(ms)	REnc(ms)	DecryptL2(ms)	DecryptL1(ms)
Weng et al.	9.22	5.64	40.32	5.09	6.27
Fang et al.	11.46	6.03	36.62	6.31	7.13
Our scheme	12.42	7.01	28.08	7.54	8.36

by cracking the login password of the backend configuration interface. In this case, even if the device and cloud have been authenticated and encrypted (such as using SSL) they cannot prevent the attack. How to ensure the safe storage and sharing of wireless multimedia data becomes particularly important. In this environment, we can deploy FC-PBRE to ensure the security of the system. The data of each device (including multimedia data, control data) is encrypted by the device owner's public key ($C_A = \text{Encrypt}(PK_A, S, m, W)$). Obviously, no user can decrypt the data except the data owner A. When the device owner A needs to grant data privileges to the maintenance engineer for the configurations of wireless devices (such as WiFi cameras), the maintenance engineer B has to request a re-encryption key $rk_{A,S',W'} = \text{ReKGen}(PK, sk_A, S', W')$, where $BinS'$ from the device owner A. The maintenance engineer B can use the re-encrypted key to perform the re-encryption algorithm $C_R = \text{REnc}(PK_A, rk_{i,S',W'}, i, S, S', C_A)$ to convert the device owner's ciphertext C_A into their own ciphertext C_R , and then B uses his own public key to decrypt the ciphertext by performing the re-encrypted ciphertext decryption algorithm $m = \text{DecryptL1}(PK_A, sk_B, A, B, S, S', C_R)$. Similarly, if the gateway or cloud platform wishes to share encrypted multimedia data to authorized third-party users (i.e., $E = E_1, E_2, \dots, E_n$) without decryption, the device owner A only needs to produce a re-encrypted key $rk_{A,E,W'} = \text{ReKGen}(PK_A, sk_A, E, W')$, which will be used to re-encrypt the message from the device owner A to the third-party user sets $E = E_1, E_2, \dots, E_n$. After the key $rk_{A,E,W'}$ is generated, it will be sent to the cloud platform. Therefore, the cloud platform can run the re-encryption algorithm $C_R = \text{REnc}(PK_A, rk_{i,E,W'}, A, S, S', C_A)$ to convert the device owner's ciphertext into C_R , which can be decrypted with the third party's own private key.

8. Conclusions

In the paper, a new security notion called fuzzy-conditional proxy broadcast re-encryption (FC-PBRE) is presented. In a FC-PBRE, the proxy uses a broadcast re-encryption key to re-encrypt a ciphertext which can be decrypted by a set of delegates if and only if the broadcast key's conditional set W is close to the conditional set W' of the ciphertext. Moreover, the proxy learns nothing about the plaintext from entire process. Second, we define the security notion against chosen-ciphertext attacks for FC-PBRE and propose an efficient fuzzy-conditional proxy broadcast re-encryption scheme. Finally, we prove that our FC-PBRE scheme is the chosen-ciphertext attack secure in the random oracle model under the Decisional nBDHE assumption.

Given our contributions, further research might explore constructing a CCA-secure FC-PBRE scheme in the standard model. It also can focus on the construction of the fuzzy-conditional proxy broadcast re-encryption schemes without pairings.

Data Availability

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

Disclosure

This paper is an extended version of the oral report [27] the authors made at the International Conference on Cloud Computing and Security (ICCCS 2018). However, this paper has not been published by the conference ICCCS 2018.

Conflicts of Interest

The authors declare that the funding in the Acknowledgments did not lead to any conflicts of interest regarding the publication of this manuscript. Also, there is no conflicts of interest in the manuscript.

Acknowledgments

Liming Fang is supported by the National Natural Science Foundation of China (nos. 61872181, 61702236, and 61300236), the National Natural Science Foundation of Jiangsu (under Grant no. BK20130809), the National Science Foundation for Postdoctoral Scientists of China (no. 2013M530254), the National Science Foundation for Postdoctoral Scientists of Jiangsu (no. 1302137C), China Postdoctoral Science Special Foundation (no. 2014T70518), and Changzhou Sci&Tech Program (no. CJ20179027). Jinyue Xia is partially supported by the National Natural Science Foundation of China (no. 6127208361300236).

References

- [1] Y. Ren, J. Shen, J. Wang, J. Han, and S. Lee, "Mutual verifiable provable data auditing in public cloud storage," *Journal of Internet Technology*, vol. 16, no. 2, pp. 317–323, 2015.
- [2] Y. Ren, J. Shen, D. Liu, J. Wang, and J.-U. Kim, "Evidential quality preserving of electronic record in cloud storage," *Journal of Internet Technology*, vol. 17, no. 6, pp. 1125–1132, 2016.
- [3] J. Kaur and K. Kaur, "A fuzzy approach for an IoT-based automated employee performance appraisal," *Computers, Materials and Continua*, vol. 53, no. 1, pp. 24–38, 2017.

- [4] A. Pradeep, S. Mridula, and P. Mohanan, "High security identity tags using spiral resonators," *Computers, Materials and Continua*, vol. 52, no. 3, pp. 185–195, 2016.
- [5] Z. Pan, J. Lei, Y. Zhang, and F. L. Wang, "Adaptive fractional-Pixel motion estimation skipped algorithm for efficient HEVC motion estimation," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 1, pp. 1–19, 2018.
- [6] J. G. Li, W. Yao, Y. C. Zhang, H. L. Qian, and J. G. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 785–796, 2017.
- [7] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, "User Collusion Avoidance CP-ABE With Efficient Attribute Revocation for Cloud Storage," *IEEE Systems Journal*, 2017.
- [8] J. G. Li, X. N. Lin, Y. C. Zhang, and J. G. Han, "KSF-OABE: outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 715–725, 2017.
- [9] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Advances in Cryptology—EUROCRYPT '98 (Espoo)*, vol. 1403 of *Lecture Notes in Computer Science*, pp. 127–144, Springer, Berlin, Germany, 1998.
- [10] 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.
- [11] J. Weng, R. H. Deng, X. Ding, C. Chu, and J. Lai, "Conditional proxy re-encryption secure against chosen-ciphertext attack," in *Proceedings of the 4th International Symposium*, p. 322, Sydney, Australia, March 2009.
- [12] C.-K. Chu, J. Weng, S. S. M. Chow, J. Zhou, and R. H. Deng, "Conditional proxy broadcast re-encryption," in *Information Security and Privacy*, C. Boyd and J. G. Nieto, Eds., vol. 5594 of *Lecture Notes in Computer Science*, pp. 327–342, Springer, Berlin, Germany, 2009.
- [13] Q. Tang, "Type-based proxy re-encryption and its construction," in *Progress in cryptology—INDOCRYPT 2008*, vol. 5365 of *Lecture Notes in Comput. Sci.*, pp. 130–144, Springer, Berlin, 2008.
- [14] J. Shao, Z. Cao, X. Liang, and H. Lin, "Proxy re-encryption with keyword search," *Information Sciences*, vol. 180, no. 13, pp. 2576–2587, 2010.
- [15] L. Fang, W. Susilo, C. Ge, and J. Wang, "Chosen-ciphertext secure anonymous conditional proxy re-encryption with keyword search," *Theoretical Computer Science*, vol. 462, pp. 39–58, 2012.
- [16] L. Fang, W. Susilo, and J. Wang, "Anonymous Conditional Proxy Re-encryption without Random Oracle," in *Provable Security*, vol. 5848 of *Lecture Notes in Computer Science*, pp. 47–60, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [17] C. Ge, W. Susilo, J. Wang, Z. Huang, L. Fang, and Y. Ren, "A key-policy attribute-based proxy re-encryption without random oracles," *The Computer Journal*, vol. 59, no. 7, pp. 970–982, 2016.
- [18] K. Liang, W. Susilo, and J. K. Liu, "Privacy-preserving ciphertext multi-sharing control for big data storage," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 8, pp. 1578–1589, 2015.
- [19] G. Ateniese, K. Benson, and S. Hohenberger, "Key-private proxy re-encryption," in *Topics in cryptology—CT-RSA 2009*, vol. 5473 of *Lecture Notes in Comput. Sci.*, pp. 279–294, Springer, Berlin, 2009.
- [20] J. Shao, P. Liu, G. Wei, and Y. Ling, "Anonymous proxy re-encryption," *Security and Communication Networks*, vol. 5, no. 5, pp. 439–449, 2012.
- [21] J. Shao, P. Liu, and Y. Zhou, "Achieving key privacy without losing CCA security in proxy re-encryption," *The Journal of Systems and Software*, vol. 85, no. 3, pp. 655–665, 2012.
- [22] Q. Zheng, W. Zhu, J. Zhu, and X. Zhang, "Improved anonymous proxy re-encryption with CCA security," in *Proceedings of the 9th ACM symposium*, pp. 249–258, Kyoto, Japan, June 2014.
- [23] J. Shao, "Anonymous ID-Based Proxy Re-Encryption," in *Information Security and Privacy*, vol. 7372 of *Lecture Notes in Computer Science*, pp. 364–375, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [24] L. Fang, J. Wang, C. Ge, and Y. Ren, "Fuzzy conditional proxy re-encryption," *Science China Information Sciences*, vol. 56, no. 5, 052116, 13 pages, 2013.
- [25] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Proceedings of the International Conference on Theory and Applications of Cryptographic Techniques*, pp. 440–456, 2005.
- [26] J. Weng, M. Chen, Y. Yang, R. Deng, K. Chen, and F. Bao, "CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles," *Science China Information Sciences*, vol. 53, no. 3, pp. 593–606, 2010.
- [27] L. Fang, L. Liu, J. Xia, and M. Sun, "A secure multimedia data sharing scheme for wireless network," *Oral report at ICCCS 2018*, 2018.