Toward Data Transmission Security Based on Proxy Broadcast Re-encryption in Edge Collaboration

QINGYANG ZHANG, JIE CUI, and HONG ZHONG, Anhui University, China
LU LIU, University of Leicester, UK

With the development of IoT, more and more data is offloaded from the cloud to the edge for computing, eventually forming a collaborative computing model at the edge. However, in this model, the problem of secure data transmission has not been solved. In this model, data is transmitted and forwarded in multiple messaging systems, and existing security schemes cannot achieve end-to-end security in a multi-hop, broadcast transmission model. Therefore, in this paper, we propose a new security scheme based on proxy re-encryption and broadcast encryption techniques. Moreover, the performance and security of the scheme are further enhanced by using online-offline techniques and a trusted execution environment when integrating the scheme with edge collaboration. Finally, this paper proves the security of the scheme in theory, compares the functionality of the scheme, analyzes the theoretical performance of the scheme, and finally measures the actual performance of the scheme in the edge collaboration system.

CCS Concepts: • **Security and privacy** → **Security protocols**; **Distributed systems security;**

Additional Key Words and Phrases: Edge computing, data security protocol, proxy re-encryption, broadcast encryption

ACM Reference format:
Qingyang Zhang, Jie Cui, Hong Zhong, and Lu Liu. 2022. Toward Data Transmission Security Based on Proxy Broadcast Re-encryption in Edge Collaboration. *ACM Trans. Sen. Netw.* 18, 3, Article 48 (August 2022), 27 pages.
https://doi.org/10.1145/3529510

1 INTRODUCTION

With the development of hardware and communication technology, by connecting to the 5G network, an increasing number of Internet of Things (IoT) devices can access the Internet at any place and any time, enabling human–thing and thing–thing interconnection, also called the Internet of Everything [26]. At the same time, a mass of data is generated by IoT devices and uploaded to the

This work was supported by the National Natural Science Foundation of China (grant nos. 61872001, 6191101332, and U1936220), the Open Fund of Key Laboratory of Embedded System and Service Computing (Tongji University), Ministry of Education (grant no. ESSCKF2018-03), the Open Fund for Discipline Construction, Institute of Physical Science and Information Technology, Anhui University, and the Excellent Talent Project of Anhui University.
Authors' addresses: Q. Zhang, J. Cui (corresponding author), and H. Zhong, Anhui University, 111 Jiulong Road, Hefei, Anhui, 230601, China; email: cuijie@mail.ustc.edu.cn; L. Liu, University of Leicester, University Road, Leicester, LE1 7RH, United Kingdom.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2022 Association for Computing Machinery.
1550-4859/2022/08-ART48 $15.00
https://doi.org/10.1145/3529510

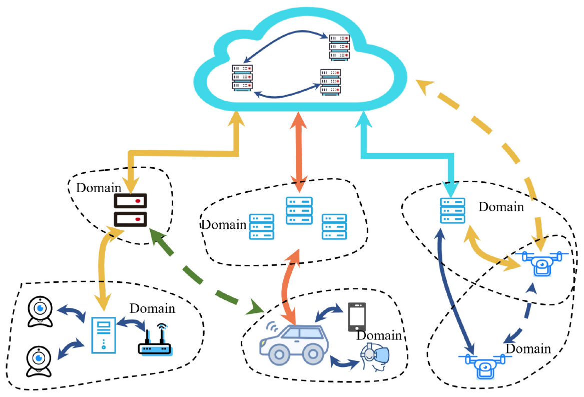


Fig. 1. Edge collaboration in the IoT.

cloud, which places a serious burden on traditional cloud computing as well as the software infrastructure layer for data forwarding and management, such as message queue systems in cloud computing. Recently, the concept of edge computing [13, 23, 25] has been proposed, which enables data to be processed by nearly edge nodes with a trade-off, that is, lower data transmission latency than cloud computing and higher computational power processing than IoT devices. Further, the concept of edge collaboration has been proposed and noticed. In the edge collaboration environment [32, 36], IoT data are no longer transmitted to a cloud computing or edge computing environment for analysis. Instead, data are processed directly through the collaboration between things and things and between edges and things, reducing the cost of data transmission and improving the quality of data services [10, 14, 35]. However, IoT data usually necessitate a large amount of user privacy, and the data transmission and processing on untrusted entities may lead to serious privacy leakage issues.

In traditional IoT scenarios, the data collected or processed at the device side are collected by various middleware systems to the cloud for final processing and distribution, such as message queue systems. However, the network environment is more complex in the edge collaboration environment. Data are no longer transmitted directly from one device to another; rather, data may go through multiple message-forwarding nodes to reach the collaboration node. In such a scenario, the data are forwarded by a third-party entity. Transmitted data maliciously tampered with will seriously affect data security issues and cause degradation of service quality [12]. As shown in Figure 1 of the edge collaboration environment, data not only collaborate within a single domain but there may also be cross-domain collaboration issues. In addition, the collaboration relationship might be dynamic. For example, a connected and autonomous vehicle (CAV) pushes computing tasks to different collaborating edge nodes and CAVs as it moves. Therefore, it is critical to ensure the security of data transmission between the collaborating parties in such a scenario.

The proxy re-encryption technique could transform one ciphertext to another ciphertext encrypted by different public keys without data decryption [3]. Thus, it could be used to enable end-to-end security in IoT software infrastructure. At the same time, the messaging system cannot know the plaintext of transmitted data. Some works have been proposed recently in cloud storage and email systems [31]. In addition, the proxy re-encryption technique is often combined with other techniques to address security issues in different scenarios. For example, by combining proxy re-encryption and attribute-based access control, the efficiency of data sharing in cloud storage is improved [15]. To solve the problem of this article's scenario, the combination of proxy re-encryption and broadcast encryption techniques is a potential solution. However, the current schemes still have some issues in terms of efficiency and functionality. For example, the number of participants has been limited in the initialization process in the traditional identity-based broadcast encryption scheme [7], which poses certain problems for the implementation of security protection in large-scale IoT scenarios, especially in the case of the explosion of the number of IoT nodes in the Internet of Everything scenario.

This article aims to design an end-to-end security scheme for dynamic edge collaboration environments. In the following, we present the application scenarios that motivate us, summarize the contributions of this article, and introduce its organization.

1.1 Motivation Applications

The edge collaboration scenario is the focus of this article, in which multi-hop data processing is often adopted. The multi-hop collaborative model described in this scenario is actually an instance of the computation path proposed by the early edge computing model [25, 34]. Here, we will introduce the edge collaboration applications motivating us.

Edge video analytics. As a killer application, edge video analytics [34] is used to propose the promotion of edge computing. As shown in the AMBER alert assistant [35], the video data will not only be transmitted to the local edge but also may be transmitted to the peripheral edge nodes because of the limited computing power of the local edge, and certain task scheduling will be performed based on the processing results.

Connected and autonomous vehicle. In fact, the CAV can be seen as a specific typical application in edge collaboration. We consider the CAV to solve its own data processing problem by collaborating with passengers' devices in the vehicle, as discussed in [28]. At the same time, the excess computing power can also be shared with other CAVs, thus, forming collaborative computing that requires multiple layers and multiple hops through the infrastructure among numerous vehicles and passengers' devices.

Unmanned aerial vehicles. Unmanned aerial vehicles (UAVs), commonly known as *drones*, are often used in disaster rescue [27]. However, the network infrastructure may be damaged in a disaster and cannot provide a reliable network. Therefore, drones build ad-hoc networks to transmit data. Network robustness can be greatly improved if drones from different organizations collaborate with each other. However, this leads to multi-hop data transmission across drones from different organizations, which raises data security issues.

1.2 Our Contribution

The main contributions are summarized as follows.

- We introduce the data transmission model and security issues in the edge collaboration environment and analyze the security requirements.
- In response to the security requirements, we propose a scheme, called *ME2E-PBRE*, based on proxy re-encryption and broadcast encryption techniques and integrate the proposed scheme into an edge collaboration framework with optimizations, supported by a trusted execution environment (TEE). This provides bidirectional, broadcast end-to-end security in multi-hop data transmission for edge collaboration.
- We prove the security and analyze the performance of the scheme in theory. In addition, we experimentally evaluate the actual performance of the scheme. The experimental results show that the data-forwarding nodes have less computation burden in our scheme and that the optimized scheme can significantly improve performance.

1.3 Organization

The remainder of this article is organized as follows. We first review related works in Section 2. We introduce the system model and security design goal in Section 3, followed by techniques used to construct our scheme and system in Section 4. Section 5 presents the framework of our scheme and the security model. The designed scheme is presented in Section 6, and the integration with optimizations is introduced in Section 7. In Section 8, we prove and analyze the security of the proposed scheme and system. We analyze the performance of the proposed scheme and evaluate it at the system level in Section 9. We present our conclusions in Section 10.

2 RELATED WORK

Proxy re-encryption. The concept of proxy re-encryption was first proposed by Blaze et al. [3] in 1998 at the European Cryptology Conference. Proxy re-encryption introduces a proxy role to the traditional public-key cryptosystem, which can transform ciphertext from one user's ciphertext to another's using the re-encryption key without decryption, ensuring data confidentiality from beginning to end. Many proxy re-encryption schemes followed. For example, Canetti et al. [5] proposed a two-way, multi-hop proxy re-encryption scheme that can resist selective ciphertext attacks. However, it is not efficient due to its bilinear pairwise operation. Due to the need for data confidentiality for the proxy, proxy re-encryption is used in a large number of applications, such as distributed file storage [2] and email systems [31]. Ateniese et al. [2] proposed a one-way proxy re-encryption scheme and applied it to distributed file storage systems for the first time.

By combining different public key cryptography techniques, proxy re-encryption schemes with other functionalities have been proposed competitively. For example, Weng et al. [30] proposed a conditional proxy re-encryption scheme. Shao et al. [24] proposed a proxy re-encryption scheme with keyword search. However, these schemes cannot meet the security and functionality requirements of this article, which requires a bidirectional, multi-hop proxy re-encryption scheme and the ability to transform a single user's ciphertext into a group's ciphertext.

Therefore, it is feasible to construct a proxy re-encryption scheme in conjunction with broadcast encryption techniques. Chu et al. [7] proposed a conditional proxy broadcast re-encryption scheme. In this scheme, only ciphertexts that meet certain conditions can be re-encrypted by the proxy; thus, they do not meet the requirements of the work in this article. Similarly, Xu et al. [31] proposed an identity-based conditional broadcast proxy re-encryption scheme, which also cannot be directly applied to the scenario of this article. In addition, Ge et al. [8] proposed an identity-based broadcast proxy re-encryption for data sharing in clouds. However, these identity-based schemes cannot simultaneously meet the requirements for function identity and node identity in this article.

Security in systems. In this part, we focus on security systems. First, we introduce the security of the publish/subscribe system, which is the common messaging system used in the IoT. Ion et al. [11] proposed a secure data transmission scheme for publish-subscribe systems based on attribute-based encryption and searchable encryption. Here, attribute-based encryption (ABE) protects the confidentiality of data in publish/subscribe systems. Similarly, Pal et al. [17] used a ciphertext policy attribute-based encryption scheme to secure data in a secure message queueing middleware system P3S, which makes messages decrypted only at the receivers that hold a set of attributes that satisfy the decryption policy. However, the ABE-based data security transmission scheme requires the data sender to encrypt against the attributes of the data receiver, which poses a high requirement for key management. Pallickara et al. [18] proposed an end-to-end encrypted topic-based message queuing system with an identity-based key management scheme to manage the keys of the nodes and symmetric encryption techniques to ensure the confidentiality of the

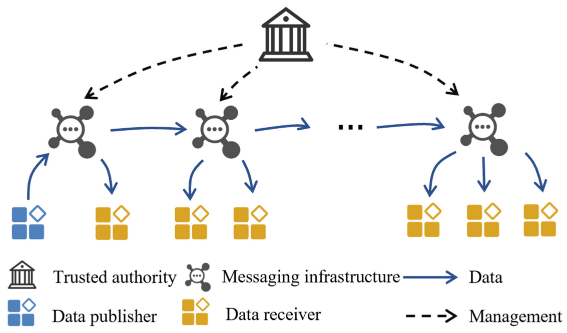


Fig. 2. System model of edge collaboration.

transmitted message. Similarly, Rajan et al. [20] also proposed an identity-based end-to-end encryption scheme to guarantee the confidentiality of data. However, this requires the data sender to know the identity of the data receiver, which does not make sense in this article.

Some studies have introduced the proxy re-encryption technique to publish-subscribe systems. However, there are still some problems for edge collaboration in the IoT. Borcea et al. [4] proposed an end-to-end secure publish-subscribe system PICADOR utilizing a lattice-based proxy re-encryption scheme proposed by Polyakov et al. [19]. In the PICADOR system, the publisher encrypts the published data using the public key, while the publish-subscribe system receives the message and re-encrypts the ciphertext data according to its topic. Although the proxy re-encryption algorithm of Polyakov et al. [19] has higher security based on the lattice difficulty problem construction, the algorithm does not have bidirectional and broadcast characteristics. This leads to the PICADOR system needing to encrypt with different re-encryption keys for different recipients during the re-encryption process, thus affecting efficiency. Zhang et al. [33] proposed a hardware-enhanced proxy re-encryption scheme using a secure TEE enclave. In fact, the proposed scheme is not a proxy re-encryption since the ciphertext will be decrypted in the secure enclave. Similarly, it does not support broadcast encryption.

3 SYSTEM MODEL

In this section, we will introduce the system model first. Then, the threat model is presented, followed by the analysis of the security objectives.

3.1 System Model

In an edge collaboration scenario, the data may be multi-hop transmitted with distributed nodes, as shown in Figure 2. An edge collaboration system consists of four entities: data publisher, data receiver, messaging infrastructure, and trusted authority. The functionalities of these four entities are as follows.

(1) *Data publisher.* The data publisher is the entity of publishing collaborative tasks. One data publisher senses data and sends data to its messaging infrastructure, which can communicate with other systems, forwarding collaborative tasks to other edge collaborative systems or finding a data receiver within the collaborative system to collaborate with.

(2) *Data receiver.* The data receiver is the entity of a collaborative task processor, which accepts the collaborative task forwarded from the collaborative system infrastructure and returns the result. A data receiver can also act as a data publisher and collaborate with other nodes. By concatenating multiple publishers and subscribers, a computation path is constructed.

(3) *Messaging infrastructure.* The infrastructure mainly includes various orchestrators and messaging systems. In the scenario of this article, it mainly accomplishes the forwarding of collaborative tasks and the transforming of ciphertexts according to the forwarding rules.
(4) *Trusted authority.* A trusted authority is mainly responsible for the key distribution of all nodes, including public system parameters, private keys, and re-encryption keys.

3.2 Threat Model

In an edge collaboration system, four different entities are included; we focus on the security of the messaging infrastructure. Here, we divide the attacks into internal attackers and external attackers. In this article, an external attacker is an attacker outside the four entities, which may be a wireless monitor or network switch and has access to all of the transmitted information. Thus, they monitor the forwarded messages, try to decrypt the data, or forward the data to other unpermitted nodes to obtain the plaintext. An internal attacker in our edge collaboration system is mainly an untrusted messaging infrastructure. Similar to an external attacker, an internal attacker tries to decrypt the data, but with additional information, such as re-encryption keys or private keys.

In addition, we assume that the data publisher, data receiver, and untrusted messaging infrastructure are equipped with a TEE. The TEE can guarantee the execution of the program on the untrusted messaging infrastructure and ensure that the internal data is not known to the messaging infrastructure. We take advantage of this property to enhance the proposed scheme.

3.3 Security Objectives

To protect the data in edge collaboration, a scheme should have the following objectives.

(1) **End-to-end.** End-to-end security means that the communication data between collaborating nodes will not be known to forwarding infrastructures. Here, we distinguish between local security and cross-domain security. We consider a solution as locally secure when it can satisfy end-to-end security for data senders and receivers under the same infrastructure. Also, when data is transmitted through multiple infrastructures, end-to-end security can still be guaranteed, which we consider to be cross-domain security. The goal of this paper is to accomplish cross-domain security. The collaborative nodes might not be connected on the same infrastructure; rather, they are distributed across multiple infrastructures, that is, in a computation path or due to network limitations. Therefore, there is a requirement that a re-encrypted ciphertext could be re-encrypted again, ensuring that data in infrastructures are not decrypted.
(2) **Broadcast.** The data or results in the system might be forwarded to multiple nodes simultaneously, including data receivers and other messaging infrastructures. The different ciphertexts are obtained from encrypting with different keys for different receivers, increasing computation and data transmission. Therefore, the broadcast encryption feature is needed to reduce the overhead, especially the communication overhead.
(3) **Isolated.** Different messages are tagged by different topics for communication in the messaging infrastructure. Even if they are the same receiver and publisher, the topics used for communication are different as long as the types of collaborative tasks are different. Thus, in terms of data security, isolation should also be done for security, that is, the ciphertext should be function related or topic related.
(4) **Bidirectional.** The transmission of data is bidirectional. For example, the data to be processed for the collaborative task and the results of the computation need to be transferred in collaboration. If different keys are used to encrypt the data, the number of keys will double, which is a key management burden for a trusted authority. Therefore, the bidirectionality of transforming ciphertext can preferably be supported as well.

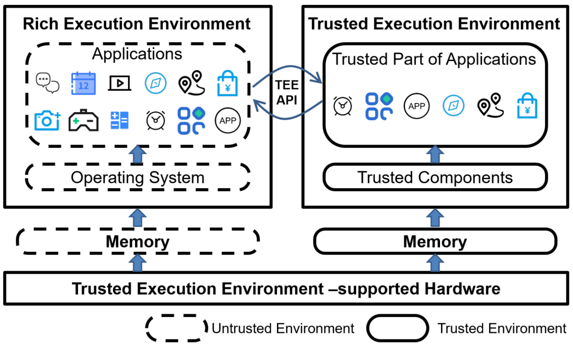

Rich Execution Environment
Applications
TEE API
Trusted Execution Environment
Trusted Part of Applications
Operating System
Trusted Components
Memory
Memory
Trusted Execution Environment –supported Hardware
Untrusted Environment
Trusted Environment

Fig. 3. The architecture of the hardware-assistant trusted execution environment.

4 BLOCKS

In this section, we will introduce the preliminary knowledge of this article.

4.1 Pairing and Related Computational Assumption

Let $\mathbb{G}$ be an additive group and $\mathbb{G}_T$ a multiplicative group with the same prime order p, while g is a generator of $\mathbb{G}_1$. Then, a bilinear pairing map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is constructed with the following properties:

(1) Bilinearity: $e(g^a, g^b) = e(g^b, g^a) = e(g, g)^{ab}$ for all $(a, b) \in \mathbb{Z}_p^{*\,2}$.
(2) Non-degeneracy: $e(g, g) \neq 1$.
(3) The result of $e(g, h)$ can be calculated in polynomial time for all $g \in \mathbb{G}$ and $h \in \mathbb{G}$.

Our scheme is based on a complexity assumption that has been used in some works, such as [21], the decisional truncated bilinear Diffie-Hellman exponent assumption (q-TBDHE), which is described as follows. Given a vector of $q + 3$ elements $(g', g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^q}, P) \in \mathbb{G}^{q+2} \times \mathbb{G}_T$, no adversary can decide whether $P = e(g', g)^{\alpha^{q+1}}$ in probability polynomial time (PPT), where $g, g' in \mathbb{G}$, $\alpha \in \mathbb{Z}_p^*$ and $P \in \mathbb{G}_T$ are random. Formally, for any PPT adversary, the probability is negligible as follows.

$$|Pr[\mathcal{A}(g', g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^q}, e(g', g)^{\alpha^{q+1}})] - Pr[\mathcal{A}(g', g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^q}, P)]| \tag{1}$$

4.2 Trusted Execution Environment

As shown in Figure 3, a TEE is a secure area in a processor for applications [22], the functionality of which is typically provided through hardware and software collaboration mechanisms. Currently, TEEs have been widely adopted in the industrial community, and mainstream processor vendors have implemented TEE in their products, such as Intel Software Guard eXtensions (SGX) [16], ARM TrustZone technology [9], and the AMD Platform Secure Processor. For example, the ARM TrustZone technology usually supports biometric authentication in mobile payment, such as facial and fingerprint authentication.

An important security feature of TEE is isolation. The application must be verified without any modification, and TEE applications could be isolated from rich execution environments (REEs). The other applications, rich operating systems, and even hypervisors cannot access, modify, and control one isolated application in TEE as well as application data inside TEE. Thus, the integrity and isolation of applications are guaranteed along with the confidentiality of their assets.

In this case, we will use a TEE to enable an isolated and secure environment in messaging infrastructures, to improve the performance of the proposed scheme, and to provide confidential computing for data publishers and data receivers.

5 DEFINITION

In this section, we present the definition of our proposed scheme, *ME2E-PBRE*. We also briefly introduce the security model used in the security proof of *ME2E-PBRE*.

5.1 Definition of *ME2E-PBRE*

As mentioned before, we focus on proposing a multi-hop proxy broadcast re-encryption scheme in this article, which consists of the following seven algorithms.

- *Setup*$(\lambda)\rightarrow(msk, mpk)$: The *Setup* algorithm is run by a trusted party. The input of this algorithm is a security parameter λ.The algorithm outputs the master public parameters *mpk* and the master secret key *msk*.
- *KeyGen*$(mpk, msk, ID_i, fun_f, S)\rightarrow sk_{i,f}$: The *KeyGen* algorithm is run by a trusted party, as is the *Setup* algorithm, to generate private keys for entities. This algorithm takes the master public parameters *mpk*, master secret key *msk*, entity identity ID_i, requesting function identity fun_f, and the set of allowed decryptions S as inputs and outputs an entity private key $sk_{i,f}$ of entity i for function fun_f.
- *MsgEnc*$(mpk, M, fun_f, S, \{U_j\}_{j\in S})\rightarrow CT$: The *MsgEnc* algorithm is run by a sender entity. The inputs of this algorithm consist of the master public parameters *mpk*, message M, private key $sk_{i,f}$, function identity fun_f, and allowed decryption set S with public information $\{U_j\}_{j\in S}$. This algorithm outputs the ciphertext CT.
- *ReKeyGen*$(mpk, sk_{i,f}, S', fun_f)\rightarrow rk_{i,S\rightarrow S',r}$: The *ReKeyGen* algorithm is run by a trusted party to generate a re-encryption key $rk_{i,S\rightarrow S',r}$, which could transform a ciphertext under the set S to set S'. This algorithm is fed by the master public parameters *mpk*, private key $sk_{i,f}$, encrypted set S, and function identity fun_f.
- *ReEnc*$(CT_S, rk_{i,S\rightarrow S',r})\rightarrow CT_{S'}$: The *ReEnc* algorithm is run by a proxy to transform ciphertext from the set S to set S'. This algorithm takes as input the ciphertext CT_S and re-encryption key $rk_{i,S\rightarrow S',r}$ and outputs the re-encrypted ciphertext $CT_{S'}$.
- *Dec-I*$(mpk, CT_S, sk_{i,f})\rightarrow M$: The *Dec-I* algorithm is run by a receiver entity. The inputs of this algorithm consist of the master public parameters *mpk*, private key $sk_{i,f}$ and ciphertext CT_S, which is the output of *MsgEnc*. The algorithm outputs decrypted plaintext M.
- *Dec-II*$(mpk, CT_S, sk_{i,f})\rightarrow M$: Similar to the *Dec-I* algorithm, *Dec-II* is run by a receiver entity. The inputs of this algorithm consist of the master public parameters *mpk*, private key $sk_{i,f}$, and ciphertext CT_S, which is the output of *ReEnc*. This algorithm outputs decrypted plaintext M.

Consistency: The consistency of an *ME2E-PBRE* scheme means that any correct ciphertext CT_S can be decrypted by the *Dec-I* or *Dec-II* algorithm with a valid private key. For multi-hop features, we have that

$$Dec - I(MsgEnc(M, S'), sk_{id,r}, \ldots) = M \tag{2}$$

$$Dec - II(\underbrace{\underline{ReEnc}(\underline{ReEnc}(\ldots(\underline{ReEnc}(\underline{ReEnc}}_{n}(MsgEnc(M), \ldots), \ldots), \ldots)), \ldots), rk_{i,S\rightarrow S',r}, S', \ldots), sk_{id,r}, \ldots) = M, \tag{3}$$

where $n \geq 1$, $id \in S'$, and all re-encryption keys have rights to transform ciphertext.

5.2 Security Model for *ME2E-PBRE*

Now, we define the security game between an adversary $\mathcal{A}$ and a challenger C in the security model for *ME2E-PBRE*. We should note that the proxy will not collude with the receiver. For simplicity of description, we omitted some parameters, such as function identity fun_f and master public parameters mpk.

Definition 2 (IND-CCA). A *ME2E-PBRE* scheme is indistinguishable chosen-ciphertext attack secure if no adversary $\mathcal{A}$ can win the following game $Exp^{\mathcal{A},IND-CCA}$ with a non-negligible advantage in probability polynomial time.

(1) **Setup.** The challenger C runs the *Setup* algorithm to get the master public parameters mpk and then gives mpk to the adversary $\mathcal{A}$.
(2) **Phase 1.** The adversary $\mathcal{A}$ is allowed to adaptively issue the following queries.
  - Key generation query (ID_i,S). The challenger C executes a key generation algorithm to generate a private key for entity identity ID_i and function identity fun_f under the set S. Then, the challenger C forwards this information to the adversary $\mathcal{A}$.
  - Re-encryption key generation query (S_1,S_2). The challenger C executes the re-encryption key generation algorithm to generate a re-encryption key with the ability to transform ciphertext under S_1 to ciphertext under S_2, where the input private key sk of this algorithm is generated by the *KeyGen* algorithm with a random identity in the set S_1, and the adversary $\mathcal{A}$ should never query any private key in S_2. Then, the challenger C returns the re-encryption key to the adversary $\mathcal{A}$.
  - Re-encryption query ($sk_{i,f}, CT_{S_1}, S_1, S_2$). If $ID_i \in S_1$, the challenger C executes the re-encryption algorithm to transform ciphertext CT_{S_1} and returns transformed ciphertext, where a re-encryption key is generated. Here, the adversary $\mathcal{A}$ never queries any private key in S_2.
  - Decryption query (ID_i, CT_S). The challenge C executes a decryption algorithm to decrypt ciphertext CT_S with a stored private key sk_i by the *KeyGen* algorithm before and included in the set S.
(3) **Challenger.** The adversary $\mathcal{A}$ outputs two equal length messages $\{M_0, M_1\}$ and the challenger C chooses a random bit $w \in \{0,1\}$. Then, the challenger C makes the challenge ciphertext to be $CT^*_{S^*} = MsgEnc(M_w, S^*, \ldots)$ and returns ciphertext $CT^*_{S^*}$ to the adversary $\mathcal{A}$. Here, the adversary $\mathcal{A}$ should never query any private key in $S^* \cup S_f$ and make any re-encryption key generation queries, which can transform CT_{S^*} to CT_{S_f}.
(4) **Phase 2.** The adversary $\mathcal{A}$ is allowed to continue making queries as in **Phase 1**, except for the following.
  - Key generation query (ID). The adversary $\mathcal{A}$ cannot query any private key sk_{ID}, where the corresponding identity is included in any decryption set queried before.
  - Re-encryption key generation query (S^*, S_f) and key generation query (ID). The adversary $\mathcal{A}$ cannot make any re-encryption key generation query and key generation query if any identity in key generation queries is included in the set S_f.
  - Re-encryption key generation query (S^*, S_f) and decryption query (sk, CT_{S_f}). The adversary $\mathcal{A}$ cannot make re-encryption key generation queries, which could transform ciphertext $CT^*_{S^*}$ to ciphertext CT_{S_f} and then query decryption to ciphertext CT_{S_f}.
  - Re-encryption query ($S^*, S_f, CT^*_{S^*}$) and key generation query (ID). The adversary $\mathcal{A}$ cannot make a re-encryption query that transforms ciphertext $CT^*_{S^*}$ to ciphertext CT_{S_f} and then query any private key, where $ID \in S_f$.
  - Decryption query ($sk_i, CT^*_{S^*}$). The adversary $\mathcal{A}$ cannot query decryption on ciphertext $CT^*_{S^*}$.

Table 1. Notations Table

Notations	Definitions
$\mathbb{G}$	A multiplicative group generated by generater g with order p
$e()$	A bilinear map $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$
H_1, H_2	Secure one-way collision-resistant hash functions
msk/mpk	The master private key and public key of the trusted party
ID_i	The identity of node i
fun_f	The identity of function f
$sk_{i,f}$	The private key of node i for function f
S	The set authenticated for decryption
CT	Ciphertext
$CT_S/CT_{S'}$	Ciphertexts that could be decrypted by any node in the set S and the set S', respectively
$rk_{i,S\rightarrow S',r}$	The re-encryption key of node i for function f to transform decryption set from S to S'

(5) **Guess**. The adversary $\mathcal{A}$ outputs the guess $w' \in \{0, 1\}$.

If $w' = w$, the adversary $\mathcal{A}$ wins the game. Thus, the advantage that the adversary $\mathcal{A}$ can break the scheme in this game can be defined as follows:

$$Adv_{\mathcal{A}}^{IND-CCA} = Pr[w' = w)] - \frac{1}{2} \tag{4}$$

If for all PPT algorithm adversary $\mathcal{A}$ and negligible function ϵ, $Adv_{\mathcal{A}}^{IND-CCA} \leq \epsilon(k)$, so that a *ME2E-PBRE* scheme is IND-CCA secure.

6 PROPOSED SCHEME

In this section, we first give an overview of the proposed scheme. Then, a detailed introduction of the proposed scheme is presented, followed by a consistency analysis. The definition of notations in our scheme is summarized in Table 1.

6.1 Overview

Inspired by Ren and Gu's identity-based broadcast encryption scheme [21], as well as Chu and Tzeng's identity-based proxy re-encryption scheme [6], we propose a multi-hop proxy broadcast re-encryption scheme. However, Wang et al. [29] have proved that the scheme [21] is insecure, where a private key holder could calculate any private key in the system. In this case, we first fix this problem by adding one random element in the key generation phase, and we use identity ID_i and function identity fun_r in keys to ensure that the used keys are related to corresponding functions. Similar to [6], a random element R is embedded into re-encrypted ciphertext and encrypted as the re-encryption key to implement multi-hop proxy re-encryption. Figure 4 is a flowchart of the proposed scheme.

6.2 Construction

Our proposed *ME2E-PBRE* scheme consists of the following algorithms.

6.2.1 $Setup(\lambda) \rightarrow (msk, mpk)$. Given a security parameter λ, this algorithm first constructs a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative groups with prime order p ($|p| = \lambda$). Let g be the generater for $\mathbb{G}$. Then, the algorithm randomly chooses $\alpha \in \mathbb{Z}_p^*$ and calculates $g_1 = g^\alpha$. Also, $(g_2, g_3, h_0, h_1, h_2) \in \mathbb{G}^5$ are randomly selected. A polynomial $f(x) = ax + b$ is randomly generated, where $(a, b) \in \mathbb{Z}_p^{*\,2}$. Note that these two numbers should be re-generated for security concerns when $g_2 = g_3^{-a}$ or $h_0 = g_3^{-b}$. Finally, two cryptographic hash functions $H_1 : \mathbb{G}^2 \times$

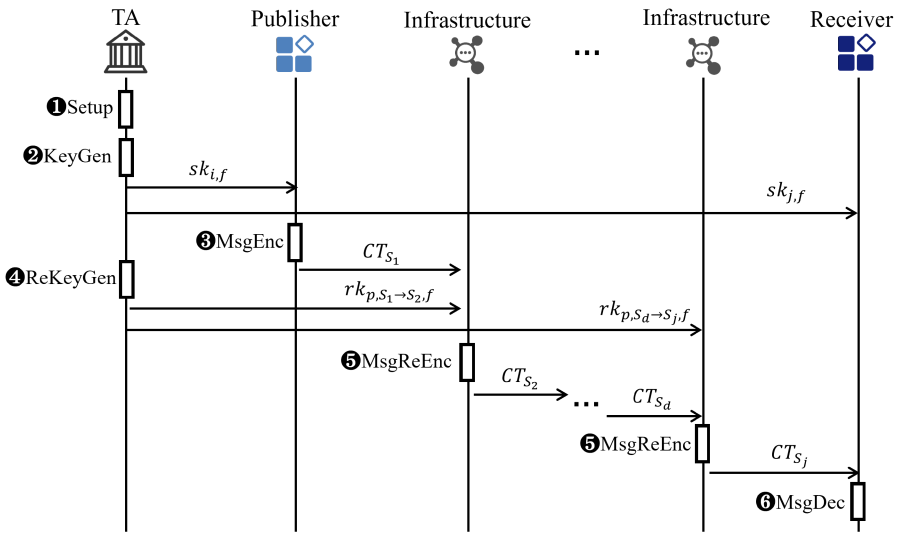


Fig. 4. The flowchart for proposed scheme.

$\mathbb{G}_T{}^* \to Z_p^*$, $H_2 : \mathbb{Z}_p^* \times \{0,1\}^* \to \mathbb{Z}_p^*$ and two functions $F_1 : \{0,1\}^l \leftrightarrow \mathbb{G}_T$, $F_2 : \mathbb{G}_T \to \mathbb{G}$ are chosen, where l is the length of a symmetric encryption key. Hence, the algorithm outputs the master private key $msk = \alpha$ and the master public key $mpk = \{g, g_1, g_2, g_3, h_0, h_1, h_2, f, H_1, H_2, F_1, F_2\}$.

6.2.2 $KeyGen(mpk, msk, ID_i, fun_f, S) \to sk_{i,f}$. When a node with $ID_i \in \mathbb{Z}_p^*$ requests a private key for the special function fun_f and set S, this algorithm will be launched. Two random large numbers $(r_{i,f,1}, r_{i,f,2}) \in \mathbb{Z}_p^{*\,2}$ are chosen, and $(r_{i,f,1}, r_{i,f,2})$ should be re-chosen if $h_0 g_2^{r_{i,f,2}} g_3^{f(r_{i,f,2})} = 1$. Thus, $u_{i,f,-1} = r_{i,f,2}$ and $u_{i,f,0} = g^{r_{i,f,1}}$. The algorithm randomly chooses a $U_i \in \mathbb{G}$ and inserts U_i in mpk as public. Then, the algorithm calculates $u_{i,f,i}$ and set $\{u_{i,f,j}\}_{j\in S\backslash i}$ as follows:

$$u_{i,f,i} = \left(h_0 g_2^{r_{i,f,2}} g_3^{f(r_{i,f,2})}\right)^{\alpha} \left(h_2^{H_2(ID_i, fun_f)} h_1^{ID_i} U_i\right)^{r_{i,f,1}} \tag{5}$$

$$u_{i,f,j} = \left(h_2^{H_2(ID_j, fun_f)} h_1^{ID_j} U_j\right)^{r_{i,f,1}}, j \in S\backslash i \tag{6}$$

Finally, the algorithm outputs the private key $sk_{i,f} = \{u_{i,f,-1}, u_{i,f,0}, \{u_{i,f,j}\}_{j\in S}\}$.

6.2.3 $MsgEnc(mpk, M, fun_f, S, \{U_j\}_{j\in S}) \to CT$. This algorithm could encrypt message $M \in \{0,1\}^*$ utilizing hybrid encryption. First, the algorithm generates a random symmetric encryption key key and encrypts the message M using the corresponding symmetric encryption algorithm formalized as $SymEnc_{key}(M)$. Then, an element $k \in \mathbb{G}_T$ is transformed by the function F_1 with the input of key.

Second, the algorithm encrypts the element k utilizing asymmetric encryption. This step could be defined as $AsymEnc(mpk, k, fun_f, S, \{U_j\}_{j\in S})$. A random large number $t \in \mathbb{Z}_p^*$ is selected, and the values $(c_1, c_2, c_3, c_4, c_5)$ are calculated as follows:

$$\begin{cases} c_1 = \prod_{j \in S} (h_2^{H_2(ID_j, fun_f)} h_1^{ID_j} U_j)^t \\ c_2 = g^t \\ c_3 = e(g_2, g_1)^t \\ c_4 = e(g_3, g_1)^t \\ c_5 = k \cdot e(h_0, g_1)^{t+s} \end{cases}, \tag{7}$$

where $s = H_1(c_1, c_2, c_3, c_4, e(h_0, g_3)^t)$. Then, the validating element $\beta = H_1(c_1, c_2, c_3, c_4, c_5, k \cdot e(h_0, g_1)^t)$ is calculated. The sub-algorithm *AsymEnc* outputs the set $\{c_1, c_2, c_3, c_4, c_5, \beta\}$.

Finally, the algorithm finishes by encrypting the message M and outputting the ciphertext CT as follows:

$$CT = \{c_1, c_2, c_3, c_4, c_5, \beta, S, SymEnc_{key}(M)\}. \tag{8}$$

6.2.4 $ReKeyGen(mpk, sk_{i,f}, S', fun_f) \rightarrow rk_{i,S \rightarrow S',f}$. This algorithm first checks the private key $sk_{i,f}$ and function identity $func_f$ to determine whether the private key is assigned for the function. If not, the algorithm will return $\perp$ or generate a random $\gamma \in \mathbb{G}_T$. Then, $R \in \mathbb{G}$ and its multiplicative inverse R^{-1} are obtained, where $R = F_2(\gamma)$. Next, the algorithm will encrypt the element γ by calling the sub-algorithm *AsymEnc* with the inputs of (mpk, S', γ, fun_f), and the output is defined as rct. Based on the private key $sk_{i,f}$, the algorithm calculates a new $u'_{i,f,0} = u_{i,f,0}R^{-1}$. Thus, it outputs the values of $RK_{i,S \rightarrow S',f} = \{u_{i,f,-1}, u'_{i,f,0}, \{u_{i,f,j}\}_{\{j \in S\}}, rct\}$.

6.2.5 $ReEnc(CT_S, rk_{i,S \rightarrow S',f}) \rightarrow CT_{S'}$. This algorithm is executed to re-encrypt the ciphertext CT_S to another ciphertext $CT_{S'}$ so that the node in the set S' could decrypt the ciphertext. First, the algorithm will check whether the re-encryption key $rk_{i,S \rightarrow S',r}$ could make the transform by checking whether the set S in CT_S is a sub-set of the set S in $rk_{i,S \rightarrow S',r}$. Once the relationship does not hold, the algorithm will return $\perp$ since the re-encryption key does have enough information to re-encrypt the message. Note that the ciphertext CT_S has two cases: the output of the algorithm *MsgEnc* and the output of the algorithm *ReEnc*.

While the ciphertext CT_S has the format$\{c_1, c_2, c_3, c_4, c_5, \beta, S_c, SymEnc_{key}(M)\}$ as original ciphertext, the algorithm will calculate the value of c_6 as follows:

$$c_6 = \frac{e(\prod_{j \in S} u_{i,f,j}, c_2)}{c_3^{u_{i,f,-1}} c_4^{f(u_{i,f,-1})} e(c_1, u'_{i,f,0})}. \tag{9}$$

The algorithm then completes the message re-encryption with the output of $CT_{S'}$ as follows:

$$CT_{S'} = \left\{c_1, c_2, c_3, c_4, c_5, \beta, c_6, c'_1, c'_2, c'_3, c'_4, c'_5, \beta', S', SymEnc_{key}(M)\right\}, \tag{10}$$

where the values of $\{c'_1, c'_2, c'_3, c'_4, c'_5, \beta'\}$ are obtained from the rct in $rk_{i,S \rightarrow S',r}$.

When the ciphertext CT_S is a re-encrypted ciphertext, its format is as follows:

$$CT_{S_c} = \left\{ c_1^1, c_2^1, \ldots, c_5^1, \beta^1, \underbrace{\underline{c_6^1, c_1^2, \ldots, c_5^2, \beta^2}, \ldots, \underline{c_6^{d-1}, c_1^d, \ldots, c_5^d, \beta^d}}_{d-1}, S^d, SymEnc_{key}(M) \right\}, \tag{11}$$

where this ciphertext has been re-encrypted by $d-1$ times, and $S^d = S$. To re-encrypt such ciphertexts, the algorithm first extracts the last set of $\{c_1^d, c_2^d, c_3^d, c_4^d, c_5^d, \beta^d\}$ and then calculates c_6^{d-1} as

Equation (9). Finally, the algorithm outputs the re-encrypted ciphertext $CT_{S'}$ by removing S^d and appending $(c_6^d, c_1^{d+1}, \ldots, c_5^{d+1}, \beta^{d+1})$ and S^{d+1} as follows:

$$CT_{S_c} = \left\{ c_1^1, c_2^1, \ldots, c_5^1, \beta^1, \underbrace{\underline{c_6^1, c_1^2, \ldots, c_5^2, \beta^2}, \ldots, \underline{c_6^d, c_1^{d+1}, \ldots, c_5^{d+1}, \beta^{d+1}}}_{d}, S^{d+1}, SymEnc_{key}(M) \right\}, \tag{12}$$

where $S^{d+1} = S'$.

6.2.6 $Dec{-}I(mpk, CT_S, sk_{i,f}) \rightarrow M$. This algorithm is utilized to decrypt the original ciphertext CT_S with the format of $\{c_1, c_2, c_3, c_4, c_5, \beta, S_c, SymEnc_{key}(M)\}$, outputted by the algorithm *MsgEnc*. First, the algorithm will check whether the set S in CT_S is a subset of the one in $sk_{i,f}$. Once the relationship does not hold, the algorithm will return $\perp$ since the private key has enough information to decrypt the message. Then, the algorithm will calculate the value of $e(h_0, g_1)^t$ as follows:

$$e(h_0, g_1)^t = \frac{e\Big(\prod_{j \in S} u_{i,f,j}, c_2\Big)}{c_3^{u_{i,f,-1}} c_4^{f(u_{i,f,-1})} e(c_1, u_{i,f,0})} \tag{13}$$

and calculate the element $\hat{s}$ as

$$\hat{s} = H_1(c_1, c_2, c_3, c_4, e(h_0, g_1)^t) \tag{14}$$

Based on c_5 and $\hat{s}$, the algorithm obtains the information $k \cdot e(h_0, g_1)^t = c_5 / e(h_0, g_1)^{\hat{s}}$ to validate the value of β in CT_S. If the calculated β' is not equal to the β in CT_S, the algorithm will be terminated and return $\perp$. Otherwise, the key information k' could be calculated by the following equation:

$$k' = \frac{k \cdot e(h_0, g_1)^t}{e(h_0, g_1)^t}. \tag{15}$$

Hence, the algorithm could calculate the symmetric key $key' = F_1^{-1}(k')$, and also the plaintext M utilizing the corresponding symmetric encryption algorithm with key', where $key' = key$.

6.2.7 $Dec - II(mpk, CT_S, sk_{i,f}) \rightarrow M$. This algorithm is utilized to recursively decrypt the re-encrypted ciphertext CT_S with the format of Equation (11) outputted by the algorithm *ReEnc*. First, the algorithm will check whether the set S in CT_S is a subset of the one in $sk_{i,f}$. Once the relationship does not hold, the algorithm will return $\perp$ since the private key has enough information to decrypt the message.

Then, the algorithm extracts the last subset of CT_S with the form $\{c_1^d, c_2^d, c_3^d, c_4^d, c_5^d, \beta^d\}$. Thus, the algorithm could obtain and validate the random element r_d following the operations for the original ciphertext, which is encrypted by *AsymEnc* in the algorithm *ReKeyGen*. Further, the inverse element R_d of r_d can be calculated by $R_d = F_2(r_d)$, and the information for decrypting previous ciphertext can be obtained as follows:

$$e(h_0, g_3)^{t_{d-1}} = \frac{c_6^{d-1}}{e(c_1^{d-1}, R_d^{-1})}. \tag{16}$$

Hence, the previous ciphertext $\{c_1^{d-1}, c_2^{d-1}, c_3^{d-1}, c_4^{d-1}, c_5^{d-1}, \beta^{d-1}\}$ can be decrypted. In this way, the original ciphertext $\{c_1^1, c_2^1, c_3^1, c_4^1, c_5^1, \beta^1\}$ can be decrypted step by step with a decrypted key information k'. Second, the symmetric encryption key key' can be calculated as $key' = F_1^{-1}(k')$, and the message is decrypted by the symmetric encryption algorithm with the key key'. Finally, the plaintext M is outputted.

6.3 Consistency

In this subsection, we explain the consistency of our *ME2E-PBRE* scheme.

For an original ciphertext encrypted by the *MsgEnc* algorithm, the key step is described in Equation (13). In this case, we have that

$$
\begin{aligned}
&\frac{e(\prod_{j\in S} u_{i,f,j}, c_2)}{c_3^{u_{i,f,-1}} c_4^{f(u_{i,f,-1})} e(c_1, u_{i,f,0})} \\
&= \frac{e\Big(\Big(h_0 g_2^{r_{i,f,2}} g_3^{f(r_{i,f,2})}\Big)^{\alpha} \prod_{j\in S}\Big(h_2^{H_2(ID_j,fun_f)} h_1^{ID_j} U_j\Big)^{r_{i,f,1}}, g^t\Big)}{c_3^{u_{i,f,-1}} c_4^{f(u_{i,f,-1})} e(c_1, u_{i,f,0})} \\
&= \frac{e\Big(\Big(h_0 g_2^{r_{i,f,2}} g_3^{f(r_{i,f,2})}\Big)^{\alpha}, g^t\Big) e\Big(\prod_{j\in S}\Big(h_2^{H_2(ID_j,fun_f)} h_1^{ID_j} U_j\Big)^{r_{i,f,1}}, g^t\Big)}{c_3^{u_{i,f,-1}} c_4^{f(u_{i,f,-1})} e\Big(\prod_{j\in S}\Big(h_2^{H_2(ID_j,fun_f)} h_1^{ID_j} U_j\Big)^{t}, g^{r_{i,f,1}}\Big)} \\
&= \frac{e\Big(\Big(h_0 g_2^{r_{i,f,2}} g_3^{f(r_{i,f,2})}\Big)^{\alpha}, g^t\Big)}{c_3^{u_{i,f,-1}} c_4^{f(u_{i,f,-1})}} \\
&= \frac{e\big(h_0^{\alpha}, g^t\big) e\Big(\Big(g_2^{r_{i,f,2}}\Big)^{\alpha}, g^t\Big) e\Big(\Big(g_3^{f(r_{i,f,2})}\Big)^{\alpha}, g^t\Big)}{e(g_2, g_1)^{t r_{i,f,2}} e(g_3, g_1)^{t f(r_{i,f,2})}} \\
&= e(h_0, g_1)^t
\end{aligned}
. \qquad (17)
$$

Thus, the consistency of an original ciphertext can be verified.

For a re-encrypted ciphertext, as shown in Equation (12), the last sub-ciphertext set is a *rct* set, and the consistency for calculating the element R_d has been verified as Equation (17). Thus, following the decryption processes, the previous R_{d-1} could be obtained as well as R_1. Finally, the symmetric encryption key *key* is computed. Hence, the key step is the consistency of Equation (16). In this way, we have that

$$
\begin{aligned}
&\frac{c_6}{e(c_1, R^{-1})} \\
&\quad= \frac{e(\prod_{j\in S} u_{i,f,j}, c_2)}{c_3^{u_{i,f,-1}} c_4^{f(u_{i,f,-1})} e(c_1, u_{i,f,0}) e(c_1, R^{-1})} \\
&\quad= \frac{e\Big(\Big(h_0 g_2^{r_{i,f,2}} g_3^{f(r_{i,f,2})}\Big)^{\alpha} \prod_{j\in S}\Big(h_2^{H_2(ID_j,fun_f) h_1^{ID_j}} U_j\Big)^{r_{i,f,1}}, g^t\Big)}{c_3^{u_{i,f,-1}} c_4^{f(u_{i,f,-1})} e(c_1, R g^{i,f,1}) e(c_1, R^{-1})}. \qquad (18) \\
&\quad= \frac{e\Big(\Big(h_0 g_2^{r_{i,f,2}} g_3^{f(r_{i,f,2})}\Big)^{\alpha} \prod_{j\in S}\Big(h_2^{H_2(ID_j,fun_f) h_1^{ID_j}} U_j\Big)^{r_{i,f,1}}, g^t\Big)}{c_3^{u_{i,f,-1}} c_4^{f(u_{i,f,-1})} e(c_1, g^{i,f,1})} \\
&\quad= e(h_0, g_1)^t
\end{aligned}
$$

Thus, the consistency of a re-encrypted ciphertext is verified.

7 INTEGRATING SCHEME IN THE EDGE-ASSISTED COLLABORATION SYSTEM

In this section, we integrate our *ME2E-PBRE* with an edge-assisted collaboration system, named TCFDL [36]. TCFDL is an edge collaboration infrastructure with trusted execution environment support.

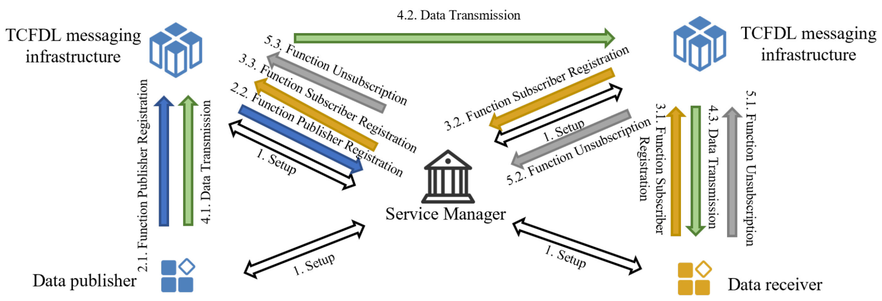

4.2. Data Transmission
TCFDL messaging infrastructure
TCFDL messaging infrastructure
5.3. Function Unsubscription
3.3. Function Subscriber Registration
2.2. Function Publisher Registration
1. Setup
3.2. Function Subscriber Registration
1. Setup
5.2. Function Unsubscription
Service Manager
2.1. Function Publisher Registration
4.1. Data Transmission
3.1. Function Subscriber Registration
4.3. Data Transmission
5.1. Function Unsubscription
Data publisher
Data receiver
1. Setup
1. Setup

Fig. 5. Architecture of the security-enhanced TCFDL system.

7.1 Overview of the Secure Edge-Assisted Collaboration System

The TCFDL system is of serverless architecture and uses message queues to transmit data and associate different collaborative entities, where data publishers and data receivers are function instances. The collaborative instances, that is, data publisher and data receiver, first need to register with the service manager (SMger), serving as the trusted authority and obtain the keys, such as private keys after performing remote attestation. Note that the communication is protected by the Transport Layer Security (TLS) protocol. When data publishers and data subscribers need to construct a collaborative relationship, they must register with the local messaging infrastructure, which will register with the SMger. The communication between collaborative instances will be automatically established by corresponding messaging infrastructures once the collaborative relationship is established on the SMger.

To integrate our scheme with the edge-assisted collaboration system, five functions should be modified with security enhancement, including Setup, Function Publisher Registration, Function Subscriber Registration, Data Transmission, and Function Unsubscription, as shown in Figure 5.

However, there are still several problems with the current scheme when applied to real systems. First, in the previous scheme, which enables multi-hop proxy broadcast re-encryption, the re-encryption key should be generated in the trusted authority. This is because the proxy could decrypt forwarding messages if it holds a private key. In this case, the proposed scheme is inefficient on re-encryption key generation when the decryption set updates frequently. Second, encryption is also computation intensive, which will increase the latency of end-to-end data transmission. In this case, we adopt *ME2E-PBRE* to the system with two improvements as follows.

- To deal with inefficient re-encryption generation in our original scheme, we migrate the re-encryption key generation to the messaging infrastructures based on TEE technology, which could protect private keys from the software, operating system, and even hypervisor.
- To minimize the impact of encryption on end-to-end data transmission latency, part of the operations are processed offline and we keep only plaintext/ciphertext-related operations online.

7.2 Setup

The service manager runs the *Setup* algorithm of Section 6 to generate the master public parameters *mpk* and master private key *msk*, then stores these values in the database. Then, a messaging infrastructure, data publisher, or data receiver can connect to the SMgr to obtain the master public parameters via a public channel. Still, it should verify identities using security technology, such as certificate-based authentication.

7.3 Function Publisher Registration

Once a messaging infrastructure of ID_{sproxy} as the edge node receives the function publisher registration fun_f from a function instance, that is, a data publisher, with identity ID_i, it will first check the identity of function instance. Then, it forwards this registration information to the SMgr using tunnelling technique so that the function instance can obtain the private key $sk_{i,f}$ with the set $S_s = \{ID_{sproxy}\}$ and element $U_{ID_{sproxy}}$. Note that the private key $sk_{ID_{sproxy},f}$ has been generated and also sent to the messaging infrastructure of ID_{sproxy}, which will be directly decrypted and stored in TEE so that the messaging infrastructure cannot obtain any information about the private key. Here, we assume that instance ID_i is a data publisher in function fun_f.

7.4 Function Subscriber Registration

If an instance ID_j is a data receiver registered in function fun_f, it sends the function subscriber registration to its local messaging infrastructure of ID_{rproxy}. Then, the SMger first generates the private key $sk_{ID_{rproxy},f}$ and sends it to the messaging infrastructure of ID_{sproxy}. Also, it will return a private key $sk_{j,f}$ with the set $S_r = \{ID_{rproxy}\}$ and element $U_{ID_{rproxy}}$. Finally, the SMgr should send the messaging infrastructure of ID_{sproxy} that another messaging infrastructure of ID_{rproxy} has subscribed the function fun_f, so that the messaging infrastructures of ID_{sproxy} and ID_{rproxy} can generate re-encryption keys $rk_{sproxy,S_s \to S_r,f}$ and $rk_{rproxy,S_r \to S',f}$, respectively, where S' includes local function instances of subscribed function fun_f and $S' = \{ID_j\}$ here. To this end, a collaborative relationship is established.

7.5 Data Transmission

Simplified, we assume only two messaging infrastructures and two function instances. When a message is transmitted from a data publisher instance of ID_i to several data receiver function instances, the data publisher instance of ID_i first encrypts a message using the *Enc* algorithm with the set S_s, which includes only one element, that is, the local messaging infrastructure of ID_{sproxy}. Then, the messaging infrastructure of ID_{sproxy} transforms CT_{S_s} to CT_{S_r} using the *ReEnc* algorithm and sends CT_{S_r} to the messaging infrastructure of ID_{rproxy}. The messaging infrastructure of ID_{rproxy} transforms the ciphertext CT_{S_r} to $CT_{S'}$, where data receiver instance ID_j is in set S'. Note that sets S_r and S' will change according to the function subscription relationship.

To reduce the latency of end-to-end data transmission, a part of the operations in this step could be online/offline as follows:

- In the *Enc* algorithm, a symmetric key key will be randomly chosen offline and element k will be calculated. Then, element k is encrypted to $\{c_1, c_2, c_3, c_4, c_5, \beta\}$. Thus, only symmetric encryption on messages needs to be performed online. phase.
- Due to most operations in the *ReEnc*, *Dec-I*, and *Dec-II* algorithms being related to the ciphertext, only small operations can be performed offline, such as $f(u_{i,f,-1})$ and $\sum_{j \in S} u_{i,f,j}$.

7.6 Function Unsubscription

To unsubscribe a function, the data receiver instance ID_j should send a request to its local messaging infrastructure of ID_{rproxy}. Then, the application in the TEE of the messaging infrastructure will update the re-encryption key, removing element ID_j from the decryption set. In addition, if there is no instance in the messaging infrastructure of ID_{rproxy}, it will notify the SMgr so that the messaging infrastructure connecting to the data publisher instance could generate a new re-encryption key and remove ID_{rproxy} from the subscription list.

8 SECURITY PROOF

In this section, we prove that the proposed scheme is theoretically secure using security reduction. We also determine whether the scheme meets our design goals. To prove the security of our *ME2E-PBRE*, we first prove that the subset of the proposed scheme is IND-CCA secure, named by the basic broadcast encryption (BBE) scheme, which consists of the *Setup*, *KeyGen*, *MsgEnc*, and *Dec-I* algorithms. Then, we can prove that the proposed scheme is IND-CCA secure.

8.1 Security Proof for the Subset of the Proposed Scheme

To prove the subset of the proposed scheme BBE, we first define the security model.

Definition 3 (BBE-IND-CCA). A BBE scheme is indistinguishable chosen-ciphertext attack secure if no adversary $\mathcal{A}$ can win the following game $Exp^{\mathcal{A},BBE-IND-CCA}$ with a non-negligible advantage in probability polynomial time.

(1) **Setup.** This is the same as in Definition 2.
(2) **Phase 1.** The adversary $\mathcal{A}$ is allowed to adaptively issue the following queries.
   - Key generation query (ID_i,S). The challenger C executes a key generation algorithm to generate a private key for entity identity ID_i and function identity fun_r under the set S. Then, the challenger C forwards this information to the adversary $\mathcal{A}$.
   - Decryption query (ID_i, CT_S). The challenge C executes a decryption algorithm to decrypt ciphertext CT_S with a private key sk_i stored earlier by the $KeyGen$ algorithm and included in the set S.
(3) **Challenger.** The adversary $\mathcal{A}$ outputs two equal-length messages $\{M_0, M_1\}$ and the challenger C chooses a random bit $w \in \{0,1\}$. Then, the challenger C makes the challenge ciphertext to be $CT^*_{S^*} = MsgEnc(M_w, S^*, \ldots)$ and returns ciphertext $CT^*_{S^*}$ to the adversary $\mathcal{A}$. Here, the adversary $\mathcal{A}$ should never query any private key in $S^* \cup S_f$ and make any re-encryption key generation queries, which can transform CT_{S^*} to CT_{S_f}.
(4) **Phase 2.** The adversary $\mathcal{A}$ is allowed to continue making queries as in **Phase 1**, except for the following.
   - Key generation query (ID). The adversary $\mathcal{A}$ cannot query any private key sk_{ID}, where the corresponding identity is included in any decryption set queried before.
   - Decryption query $(sk_i, CT^*_{S^*})$. The adversary $\mathcal{A}$ cannot query decryption on ciphertext $CT^*_{S^*}$.
(5) **Guess.** The adversary $\mathcal{A}$ outputs the guess $w' \in \{0,1\}$.

If $w' = w$, the adversary $\mathcal{A}$ wins the game. Thus, the advantage that the adversary $\mathcal{A}$ can break the scheme in this game can be defined as follows:

$$Adv_{\mathcal{A}}^{BBE-IND-CCA} = Pr[w' = w)] - \frac{1}{2}. \tag{19}$$

If for all PPT algorithm adversaries $\mathcal{A}$ and negligible function ϵ, $Adv_{\mathcal{A}}^{BBE-IND-CCA} \leq \epsilon(k)$; thus, a BBE scheme is IND-CCA secure.

Theorem 1. *Assume that the (t', ε', q)-TBDHE assumption holds in $\mathbb{G}, \mathbb{G}_T$, then the BBE scheme is $(t, \varepsilon, q_k, q_d)$ IND-CCA secure for $t = t' - O(t_{exp} \cdot qn) - O(t_{pair} \cdot q)$, $\varepsilon = \varepsilon' + 1/(p-1)$, $q_k + q_d \leq q - 1$, where t_{exp}, and t_{pair} are the average times required to exponentiate and pair in $\mathbb{G}_1, \mathbb{G}_T$, respectively.*

Proof. Assume that there is an adversary $\mathcal{A}$ breaking our BBE scheme. We construct a simulator $\mathcal{B}$ to solve the q-TBDHE problem. At the end of the game, the simulator $\mathcal{B}$ is given a vector $(g', g, g^{\alpha}, \ldots, g^{\alpha^q}, P) \in \mathbb{G}^{q+2} \times \mathbb{G}_T$, and decides whether $P = e(g', g)^{\alpha^{q+1}}$.

Setup. The simulator $\mathcal{B}$ randomly chooses three functions $f_a(x) = \sum_{i=0}^{q} a_i x^i$, $f_b(x) = \sum_{i=0}^{q} b_i x^i$, and $f_c(x) = \sum_{i=0}^{q} c_i x^i$ with the degree q. Let $g_1 = g^{\alpha}$, $h_0 = g^{f_a(\alpha)}$, $g_2 = g^{f_b(\alpha)}$, $g_3 = g^{f_c(\alpha)}$, $f(x) = -\frac{b_q}{c_q}x - \frac{a_q}{c_q}$, $h_1 = g^{r_1}$, and $h_2 = g^{r_2}$, where $(\alpha, r_1, r_2) \in \mathbb{Z}_p^{*3}$ is randomly chosen. Similar to proposed scheme, three functions $f_a(x)$, $f_b(x)$, and $f_c(x)$ should be randomly chosen again, if $g_2 = g_3^{b_q/c_q}$ or $h_0 = g_3^{a_q/c_q}$. Then, the simulator $\mathcal{B}$ sends the master public parameters mpk to the adversary $\mathcal{A}$, where $mpk = \{g, g_1, g_2, g_3, h_0, h_1, h_2, f(x)\}$.

Phase 1. For different queries, the simulator $\mathcal{B}$ responds as follows.

- Key generation query (ID_i,fun_f,S). The simulator $\mathcal{B}$ receives the identity ID_i, function identity fun_f, and decryption set S. If $ID_i = \alpha$ or any $ID_j \in S$ is equal to α, the simulator solves the q-TBDHE problem. Otherwise, the simulator $\mathcal{B}$ randomly chooses two large numbers $(r_{i,f,1}, r_{i,f,2}) \in \mathbb{Z}_p^{*2}$ and $U_i = g^{d_i} \in \mathbb{G}$, where $h_0 g_2^{r_{i,f,2}} g_3^{f(r_{i,f,2})} \neq 1$, then calculates related private keys $u_{i,f,-1}$, $u_{i,f,0}$, and $\{u_{i,r,j} = (h_2^{H_2(ID_j,fun_r)} h_1^{ID_j} U_j)^{r_{i,r,1}}\}_{j\in S\backslash i}$ in the same way as in the real scheme. Then, it computes $u_{i,f,i}$ as follows, and returns the private key $sk_{i,r} = \{u_{i,r,-1}, u_{i,r,0}, \{u_{i,r,j}\}_{j\in S}\}$.

$$u_{i,f,i} = \left(g^{\sum_{i=0}^{q-1}(a_i + r_{i,f,2}b_i + f(r_{i,f,2})c_i)\alpha^{i+1}}\right)\left(h_2^{H_2(ID_i,fun_f)} h_1^{ID_i} U_i\right)^{r_{i,f,1}} \tag{20}$$

Here, the outputted $sk_{i,r}$ is a valid private key, since we have the following equations:

$$\begin{aligned} g^{\sum_{i=0}^{q-1}(a_i + r_{i,f,2}b_i + f(r_{i,f,2})c_i)\alpha^{i+1}} &= g^{\sum_{i=0}^{q}(a_i + r_{i,f,2}b_i + f(r_{i,f,2})c_i)\alpha^{i+1}} \\ &= \left(g^{f_a(\alpha)} \cdot g^{r_{i,f,2}f_b(\alpha)} \cdot g^{f(r_{i,f,2})f_c(\alpha)}\right)^{\alpha} \\ &= \left(h_0 g_2^{r_{i,f,2}} g_3^{f(r_{i,f,2})}\right)^{\alpha} \end{aligned} \tag{21}$$

$$u_{i,f,i} = \left(h_0 g_2^{r_{i,f,2}} g_3^{f(r_{i,f,2})}\right)^{\alpha} \cdot \left(h_2^{H_2(ID_i,fun_f)} h_1^{ID_i} U_i\right)^{r_{i,f,1}}, \tag{22}$$

where $f(r_{i,f,2}) = -(b_q/c_q)r_{i,f,2} - (a_q/c_q)$ and $a_q + r_{i,f,2}b_q + f(r_{i,f,2})c_q = 0$. Hence, $sk_{i,f}$ is randomly distributed due to the randomness of $r_{i,f,1}$, $r_{i,f,2}$, and $U_{j\,j\in S\cup i}$.

- Decryption query (ID_i, CT_S). The simulator $\mathcal{B}$ receives the identity ID_i and a ciphertext CT_S from the adversary $\mathcal{A}$. The simulator $\mathcal{B}$ first checks whether $ID_i \in S$ or not. If yes, the simulator $\mathcal{B}$ obtains $sk_{i,f}$ from stored private keys, then decrypts CT_S and returns a decrypted message. Otherwise, the simulator $\mathcal{B}$ returns a $\perp$.

Challenge. start here The adversary $\mathcal{A}$ randomly generates two messages M_1 and M_2, and one decryption set S^*, where the identities have never been queried in the key generation query in Phase 1. Once a message is received from $\mathcal{A}$, the simulator $\mathcal{B}$ randomly chooses a bit $w \in \{0,1\}$, and symmetric key element k^*, then calculates the ciphertext $(c_1^*, c_2^*, c_3^*, c_4^*, c_5^*, \beta^*)$ as follows:

$$\begin{cases} c_1^* = (g')^{\sum_{i\in S^*} r_2 H_2(ID_i,fun_f) + r_1 ID_i + d_i} \\ c_2^* = g' \\ c_3^* = P^{b_q} \cdot e(g', g)^{\sum_{i=0}^{q-1} b_i \alpha^{i+1}} \\ c_4^* = P^{c_q} \cdot e(g', g)^{\sum_{i=0}^{q-1} c_i \alpha^{i+1}} \\ c_5^* = k^* \cdot \dfrac{e(c_2^*, u_{i,f,i}^* \cdot \prod_{j\in S^*\backslash i} u_{i,f,j})}{(c_4^*)^{f(r_{i,f,2}^*)} \cdot (c_3^*)^{r_{i,f,2}^*} \cdot e(u_{i,r,0}^*, c_1^*)} \cdot e(h_0, g_1)^{s^*} \\ \beta^* = H_1(c_1^*, c_2^*, c_3^*, c_4^*, c_5^*, c_5^*/e(h_0, g_1)^{s^*}) \end{cases}, \tag{23}$$

where $sk^*_{i,f}$ is a private key of identity $ID^*_i \in S^*$ and s^* as follows.

$$s^* = H_1\left(c^*_1, c^*_2, c^*_3, c^*_4, \frac{e(c^*_2, u^*_{i,f,i} \cdot \prod_{j\in S^*\setminus i} u_{i,f,j})}{(c^*_4)^{f(r^*_{i,f,2})} \cdot (c^*_3)^{r^*_{i,f,2}} \cdot e(u^*_{i,r,0}, c^*_1)}\right), \tag{24}$$

Phase 2. The adversary $\mathcal{A}$ makes additional queries as defined in the security model and the simulator $\mathcal{B}$ responds as in Phase 1.

Guess. The adversary $\mathcal{A}$ guesses $w' \in \{0, 1\}$ and submits it to the simulator $\mathcal{B}$. If $w = w'$, the adversary $\mathcal{A}$ wins the game.

The simulation is completed. In a challenge, for any private key $sk^*_{i,f}$ in S^*, we have that

$$\frac{e(c^*_2, u^*_{i,f,i} \cdot \prod_{j\in S^*\setminus i} u_{i,f,j})}{(c^*_4)^{f(r^*_{i,f,2})} \cdot (c^*_3)^{r^*_{i,f,2}} \cdot e(u^*_{i,r,0}, c^*_1)} = P^{a_q} \cdot e(g', g)^{\sum_{i=0}^{q-1} a_i \alpha^{i+1}}. \tag{25}$$

From this view, even though the simulator $\mathcal{B}$ can generate multiple private keys for ID^*_i, it still cannot decide whether $P = e(g', g)^{\alpha^{q+1}}$. Thus, the simulator $\mathcal{B}$ sends CT^*_S to the adversary $\mathcal{A}$. In $\mathcal{A}$'s view, let $t^* = log_g g'$. Thus, we have that

$$\begin{cases} c^*_1 = (g')^{\sum_{i\in S^*} r_2 H_2(ID_i, fun_f) + r_1 ID_i + d_i} = \prod\limits_{i\in S^*} (h_2^{H_2(ID_i, fun_f)} h_1^{ID_i} U_i)^{t^*} \\ c^*_2 = g' = g^{t^*} \\ c^*_3 = P^{b_q} \cdot e(g', g)^{\sum_{i=0}^{q-1} b_i \alpha^{i+1}} = e(g', g^{f_b(\alpha)})^\alpha = e(g_1, g_2)^{t^*} \\ c^*_4 = P^{c_q} \cdot e(g', g)^{\sum_{i=0}^{q-1} c_i \alpha^{i+1}} = e(g', g^{f_c(\alpha)})^\alpha = e(g_1, g_3)^{t^*} \\ c^*_5 = k^* \cdot \dfrac{e(c^*_2, u^*_{i,f,i} \cdot \prod_{j\in S^*\setminus i} u_{i,f,j})}{(c^*_4)^{f(r^*_{i,f,2})} \cdot (c^*_3)^{r^*_{i,f,2}} \cdot e(u^*_{i,r,0}, c^*_1)} \cdot e(h_0, g_1)^{s^*} = k^* \cdot e(h_0, g_1)^{t^*+s^*} \\ \beta^* = H_1(c^*_1, c^*_2, c^*_3, c^*_4, c^*_5, c^*_5/e(h_0, g_1)^{s^*}) = H_1(c^*_1, c^*_2, c^*_3, c^*_4, c^*_5, k^* \cdot e(h_0, g_1)^{t^*}) \end{cases}, \tag{26}$$

where $s^* = H_1(c^*_1, c^*_2, c^*_3, c^*_4, e(h_0, g_1)^{t^*})$ and the following equation holds.

$$\frac{e(c^*_2, u^*_{i,f,i} \prod_{j\in S^*\setminus i} u_{i,f,j})}{(c^*_4)^{f(r^*_{i,f,2})} \cdot (c^*_3)^{r^*_{i,f,2}} \cdot e(u^*_{i,r,0}, c^*_1)} = e(h_0, g_1)^{t^*} \tag{27}$$

Hence, CT^*_S is a valid ciphertext with the randomness of t^* and is indistinguishable for the adversary $\mathcal{A}$. In addition, a ciphertext can be accepted and decrypted only if it is valid, and all invalid ciphertext will be returned by $\perp$. Due to the uniform randomness of t^*, s^* is also uniformly random; the adversary $\mathcal{A}$ cannot calculate a valid c_1 or c_5 from CT^*_S to forge a valid ciphertext. Then, in c^*_5, since t^* and s^* are random, $t^* + s^*$ is random and $t^* + s^* = 0$ is with the probability of $1/(p-1)$. Also, c^*_5/k^* is random for the adversary $\mathcal{A}$ and it can win the game with a probability of at most $1/2 + 1/(p-1)$. In the simulation, the runtime of the simulator $\mathcal{B}$ is bound by $t \le t + O(qnt_{exp}) + O(qt_{pairing})$ where n is the size of the set S.

8.2 Security Proof for Proposed Scheme

Here, we are proving that the proposed scheme is IND-CCA secure.

Theorem 2. *Assume that there is an adversary $\mathcal{A}$ that has advantage ϵ against the game $Exp^{\mathcal{A}, BBE-IND-CCA}$. Then, there is a simulator $\mathcal{B}$ that could break the BBE scheme with at least* *the advantage as follows:*

$$Adv_{\mathcal{B}}^{IND-CCA} \geq \epsilon/e(1+q_m), \tag{28}$$

where q_m is the maximum number of queries to KeyGen by the adversary $\mathcal{A}$ and e is the base of the natural logarithm. The runtime of the simulator is $O(time(\mathcal{A}))$.

Proof. Assume that there is an adversary $\mathcal{A}$ breaking our *ME2E-PBRE* scheme. We construct a simulator $\mathcal{B}$ to break the BBE scheme in the following steps. Note that the simulator $\mathcal{B}$ maintains a table T_1 with tuples (η, ID, S, S', rk) and a table T_2 with (η, ID, S, sk), where $*$ is the wildcard.

Setup. The **Setup** of BBE is performed, and the output is forwarded by simulator $\mathcal{B}$ to the adversary $\mathcal{A}$.

Phase 1. For different queries, the simulator $\mathcal{B}$ responds as follows.

- Key generation query (ID_i,fun_f,S). The simulator $\mathcal{B}$ first generates a random coin η where $Pr[\eta = 1] = \rho$ for some rho that will be determined later. Then, if $\eta = 0$ or $(0, ID, S, *)$ already exists on the table T_2, the simulator $\mathcal{B}$ outputs randomly and aborts. Otherwise, a key generation query is sent by the simulator $\mathcal{B}$ with same parameters to the BBE scheme, and the returned private key $sk_{i,f}$ is forwarded to the adversary $\mathcal{A}$. Then, the simulator $\mathcal{B}$ inserts tuple $(1, ID, S, S, \perp)$ into the table T_1 and tuple $1, ID, S, \perp$ into the table T_2.
- Re-encryption key generation query (S_1,S_2). The simulator $\mathcal{B}$ first generates a random coin η with $Pr[\eta = 1] = \rho$. If $\eta = 1$, $(1, ID, S_1, S_1, \perp)$, or $(1, *, S_2, S_2, \perp)$ already exists in T_1, and $(1, ID, S_1, \perp)$ or $(1, ID, S_2, \perp)$ already exists in T_2, the simulator $\mathcal{B}$ makes the key generation query to the BBE, and then calculates the re-encryption key as the real scheme in Section 6, which is returned to the adversary $\mathcal{A}$. Otherwise, the simulator $\mathcal{B}$ generates a re-encryption key $rk_{i,S_1 \to S_2,f} = (x, y, MsgEnc(mpk, z, S_2))$, where $\{x, y, z\} \in \mathbb{Z}_p^* \times \{0,1\}^l$. Finally, tuple $(\eta, ID, S_1, S_2, rk_{i,S_1 \to S_2,f})$ is inserted into the table.
- Re-encryption query $(sk_{i,f}, CT_{S_1}, S_1, S_2)$. If $ID_i \in S_1$, the simulator $\mathcal{B}$ performs re-encryption key generation query with $sk_{i,f}, S_1, S_2$ to obtain $rk_{i,S_1 \to S_2,f}$, then the simulator $\mathcal{B}$ re-encrypts the ciphertext as a real scheme utilizing $rk_{i,S_1 \to S_2,f}$.
- Decryption query (ID_i, CT_S). If CT_S is an original ciphertext, the simulator $\mathcal{B}$ forwards to the decryption of the BBE scheme and outputs of the BBE scheme to the adversary $\mathcal{A}$. If CT_S is a re-encrypted ciphertext, assume that the last ciphertext set rct is encrypted under the set S, and the previous ciphertext is encrypted under the set S'. If $(0, ID, S', S, rct')$ exists in T_2, the simulator $\mathcal{B}$ first checks whether $rct = rct'$. If passed, the simulator $\mathcal{B}$ makes a decryption query to the BBE scheme with rct, and then computes plaintext as a real scheme. Finally, The simulator $\mathcal{B}$ returns the plaintext to the adversary $\mathcal{A}$.

Challenge. The adversary $\mathcal{A}$ selects (ID^*, S^*, m_0, m_1) to the simulator $\mathcal{B}$. If tuple $(1, ID^*, S^*, *)$ exists on the table T_1, the simulator $\mathcal{B}$ aborts the game. Otherwise, the simulator $\mathcal{B}$ sends the challenge to the one of the BBE scheme, and forwards ciphertext CT_{S^*} to the adversary $\mathcal{A}$.

Phase 2. The same as **Phase 2** with limitations described in the definition.

Guess. The adversary $\mathcal{A}$ guesses $w' \in \{0,1\}$ and submits it to the simulator $\mathcal{B}$. The simulator will forward the guess to the **Guess** of the BBE. If $w = w'$, the adversary $\mathcal{A}$ wins the game.

The simulation is completed. The simulation $\mathcal{B}$ in the proof almost acts the same as the real scheme except for the incorrect form of re-encryption keys for $\eta = 0$. Hence, only two cases should be considered with indistinguishability, values $(x, y, MsgEnc(mpk, z, S_2))$ and the real re-encryption key. Since (x, y) must be a valid element of $(u_{i,f,-1}, u_{i,f,0}R^{-1})$ of $sk_{i,f}$, the indistinguishability of the simulator could be proved if the encryption on R is indistinguishable. Hence, the simulator is indistinguishable based on Theorem 1.

Assume that the adversary makes q_m key generation queries. The probabilities that the simulator $\mathcal{B}$ will not abort in **Phase 1** and **Phase 2** and in **Challenge** are ρ^{q_c} and $1-\rho$, respectively. Thus, the probability of not aborting the game is $\rho^{q_m}(1-\rho)$, which is maximized at $\rho' = 1 - 1/(q_c + 1)$. Therefore, the advantage of the simulator $\mathcal{B}$ is at least $\epsilon/e(1 + q_m)$.

8.3 Security Analysis

As mentioned in Section 3, our goal of the proposed scheme and integrated system should ensure the security objectives.

(1) **End-to-end.** First, the re-encryption key owned by the infrastructure has specified the decryption group transformation rules. For example, with the re-encryption key $rk_{i,S\to S',f}$, the infrastructure can only transform the decryption sets from S to S'. Second, the infrastructure does not have access to its own private key. Therefore, the infrastructure cannot decrypt or transform ciphertexts to other decryption sets as shown in Theorem 2. Third, the scheme allows the re-encrypted ciphertext to be re-encrypted again. Therefore, the scheme satisfies cross-domain end-to-end security.

(2) **Broadcast.** The proposed scheme uses broadcast encryption, which allows the encrypted data to be decrypted by any node in the decryption set. Also, as described in Theorem 2, decryption cannot be performed by a node without a private key. Therefore, the scheme in this article satisfies broadcastability and is secure.

(3) **Isolated.** Function-related information is embedded in the data encryption process in the proposed scheme. Meanwhile, both the private and re-encryption keys are embedded with a function identity. When the data is decrypted/re-encrypted using the private key/re-encryption key with different embedded function information, the function-related information cannot be eliminated correctly, which leads to operation failure. Therefore, the proposed scheme satisfies the function isolation.

(4) **Bidirectional.** In the proposed scheme, the forwarding infrastructure can transform the ciphertext using a re-encryption key $rk_{i,S\to S',f}$. However, when the computation result is transmitted back, the forwarding infrastructure needs another re-encryption key $rk_{i,S'\to S,f}$ to complete the ciphertext transformation. Therefore, we say that the proposed scheme cannot satisfy bidirectionality. However, in the integrated system, since the TEE of the infrastructure holds the private key and it can dynamically generate re-encryption keys, we say that the integrated system satisfies bidirectionality.

9 PERFORMANCE

In this section, we will give a detailed theoretical analysis and comparison in term of functionality, storage, and communication. Then, we evaluate the performance in the integrated system. All experiments are evaluated in a Ubuntu PC with Intel i5-7400 CPU @3.0 GHz (performance mode), and the TEE is implemented by Intel SGX,

9.1 Theoretical Analysis and Comparison

9.1.1 Functionality. We compare our scheme and system with several schemes on functionality, including PICADOR [4], Xu et al. [31], hPRESS [33] and TCFDL [36]. Compared results are shown as Table 2. Here, bidirectionality is the ability that a ciphertext can be transformed from S to S' as well as from S' to S when one proxy holds one key. Broadcasting is the ability to allow one-to-many encryption, such as broadcast encryption. Local security means that a message is secure when publisher and receiver are communicated in the same messaging infrastructure, where the message is transformed by only one messaging infrastructure. It is also the feature of one-hop proxy

Table 2. Functionality Comparison

Scheme	Bidirectional	Broadcast	Local Security	Cross-Domain End-to-End Security
PICADOR [4]	×	×	√	×
Xu et al. [31]	×	√	√	×
TCFDL [36]	×	×	√	√
hPRESS [33]	×	×	√	√
Ours (Section 6)	×	√	√	√
Ours (Section 7)	√	√	√	√

re-encryption. Cross-domain end-to-end security means that message publishers and receivers are distributed across different messaging infrastructures and that the data remainsecure end-to-end regardless of how many messaging infrastructures they are forwarded through in transmission. Also, this is the feature of multi-hop proxy re-encryption.

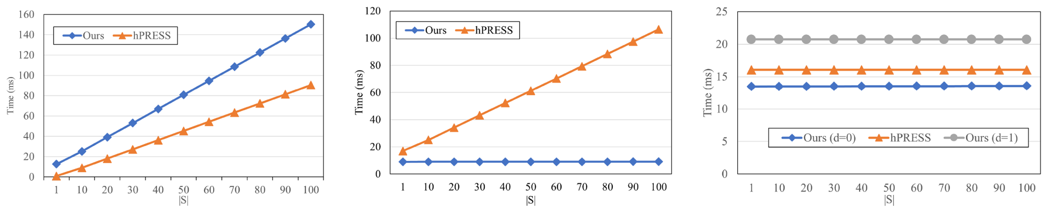
PICADOR [4] is a secure publish/subscribe system that utilizes proxy re-encryption technique to ensure that data are confidential to the messaging infrastructure. However, it can ensure this feature only locally and cannot support cross-domain security. Xu et al. proposed an identity-based proxy broadcast encryption scheme and applied it to an email system. However, the message could be transformed only once; thus, it cannot support cross-domain security. In TCFDL [36], two secure schemes are supported in the system. However, the messaging infrastructure needs to use different keys for transforming ciphertext between different decryption sets as well as different keys for different elements in the set. Thus, it is without the features of bidirectionality and broadcast. Similarly, hPRESS is also without the features of bidirectionality and broadcast and it must decrypt and re-encrypt ciphertext in TEE with different keys. In our scheme in Section 6, the private key of the messaging infrastructure will be issued, and the messaging infrastructure has to use different re-encryption keys to transform ciphertext. In this case, the bidirectionality is not supported. However, when integrated with the system, we make private keys held in the TEE of a messaging infrastructure. Thus, the messaging infrastructure could derive re-encryption keys by itself. In this case, bidirectionality is achieved.

9.1.2 Computation. Here, we compared our scheme with hPRESS [33]. The reason that we do not compare ours with the other schemes is that the others cannot achieve cross-domain security and the security of the scheme used in TCFDL is weak. To compare our scheme with hPRESS, we counted the types and numbers of cryptographic operations used by different algorithms in the scheme of *ME2E-PBRE* and hPRESS [33]. We implemented and measured the time of these basic operations using a cryptographic library, MIRACL core [1], so that we can derive the time spent by different algorithms. The security level we evaluated is 128 bit. The chosen pairing is BLS12381, and the length of RSA is 3,072. The order of $\mathbb{G}_1$ and $\mathbb{G}_2$ is 255-bit length. We should note that BLS12381 is an asymmetric pairing curve and our scheme is also secure on top of asymmetric pairing without modification. The time consumption of cryptographic operations is shown in Table 3, and time consumption in algorithm level is illustrated as Figure 6. Note that the decryption time depends on the algorithm invoked, *Dec-I* or *Dec-II*, based on the number of re-encryption operations.

For the *Enc* algorithm, the time consumption of our scheme is $2|S|t_{gm} + (4|S|+1)t_{gp} + 4t_{pairing} + 2t_{tm}$, and that of hPRESS is t_{rsaenc}, where RSA encryption and decryption have been implemented based on the standard. For the *ReEnc* algorithm, the time consumptions of our scheme and hPRESS are $(|S|-1)t_{gm} + 2t_{pairing} + 3t_{tm} + 2t_{tp}$ and $t_{rsadec} + t_{rsaenc}$, respectively. For *Dec*, our scheme and hPRESS are $(|S|-1)t_{gm} + 3t_{pairing} + 5t_{tm} + 3t_{tp} + (d-1)(2t_{pairing} + 2t_{tm} + t_{tp})$ and t_{rsadec}, respectively, where d is the number of re-encryptions for ciphertext. From Figure 6, our scheme has advantages over the *ReEnc* and *Dec* algorithms, where the ciphertext is without re-encrypting.

Table 3. Cryptography Basic Operation Time

Notation	Description	Value (μs)
t_{gm}	Multiplication operation time on $\mathbb{G}$	1
t_{gp}	Power operation time on $\mathbb{G}$	347
$t_{pairing}$	Pairing operation of e	2,726
t_{tm}	Multiplication operation time on $\mathbb{G}_T$	8
t_{tp}	Power operation time on $\mathbb{G}_T$	1,758
t_{rsaenc}	Encryption operation on RSA	904.09
t_{rsadec}	Decryption operation on RSA	16,065.7



(a) Enc (b) ReEnc (c) Dec

Fig. 6. Computation time consumption of different algorithms.

Table 4. Storage and Communication Cost Comparison

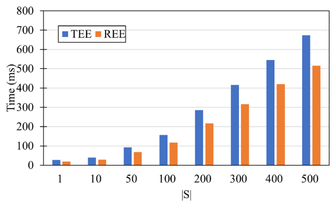
Type	Size
Parameters in trusted authority	$(2+3\|ID\|)l_{\mathbb{Z}_p} + (7+\|ID\|)l_{\mathbb{G}}$
$sk_{i,f}$	$(\|S\|+1)l_{\mathbb{Z}_p} + \|S\|l_{\mathbb{G}}$
$rk_{i,S\to S',f}$	$(\|S\|+\|S'\|+1)l_{\mathbb{Z}_p} + (\|S\|+2)l_{\mathbb{G}} + 3l_{\mathbb{G}_T} + l_{hash}$
Original CT_S	$\|S\|l_{\mathbb{Z}_p} + 2l_{\mathbb{G}} + 3l_{\mathbb{G}_T} + l_{hash}$
Increased value from CT_S to $CT_{S'}$	$(\|S'\|-\|S\|)l_{\mathbb{Z}_p} + 2l_{\mathbb{G}} + 4l_{\mathbb{G}_T} + l_{hash}$

For 5G-enabled infrastructure, this is a huge advantage where many messages will be forwarded by infrastructure, that is, messaging infrastructures in our article. Our scheme could be improved by online/offline technique, which we will show in experiments.

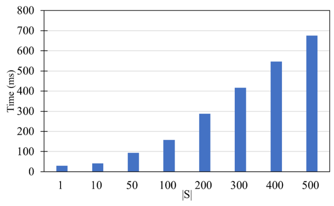
9.1.3 Storage and Communication. Here, we focus on the impact of our scheme on the amount of data communicated and stored. Table 4 demonstrates these effects, where ID is the count of all identities in system, $|S|$ and $|S'|$ are the size of the sets, $l_{\mathbb{Z}_p}$, $l_{\mathbb{G}}$, and $l_{\mathbb{G}_T}$ are the size of one element in a corresponding group, and l_{hash} is the size of the hash function output used in system. As a trusted authority, it must store msk, mpk, and all information for the node private key, which could be used to generate real private keys and re-encryption keys. The relationships between identities, private keys, and elements U_i should also be stored so that additional $|ID|$ elements in $\mathbb{Z}_p$ are stored. Moreover, for re-encryption, the identities for the set S could be removed from CT_S and the identities for new set S' should be added.

9.2 Experiment Analysis

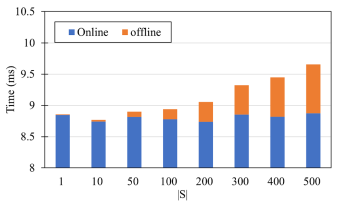
To evaluate the performance of the proposed scheme in the system, we implement our scheme as in Section 7 in TCFDL using the C++ programming language and MIRACL core library with the BLS12381 pairing-friend curve. As shown in Section 7, we make messaging infrastructures keeping the private key in TEE, and preprocess some ciphertext-independent operations in the



(a) *Enc* (offline)



(b) *ReKeyGen*



(c) *ReEnc*

Fig. 7. Time consumption of the *Enc*, *ReKeyGen*, and *ReEnc* algorithms in the system.

offline phase. The results are shown as follows. Here, the time on operations in a rich execution environment is labelled as REE, and the time on operations in a trusted execution environment is labelled as TEE.

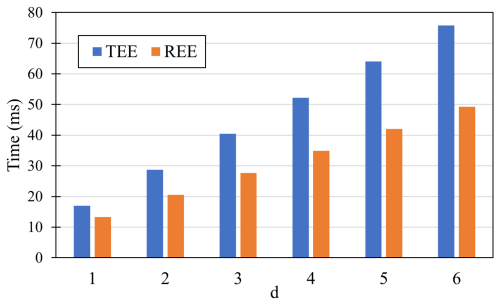
All operations of the *Enc* algorithm could be run in TEE or REE depending on the function instance. Thus, we evaluated these two cases. Figure 7(a) illustrates the time consumption of our *Enc* algorithm as the size of decryption set S varies. Encrypting data in TEE consumes more time due to the additional time required for switching the CPU mode from REE to TEE and for transmitting data between different modes. Further, in most cases, the *Enc* algorithm is performed by the data publisher, which means that the decryption set contains only one messaging infrastructure. Therefore, the encryption cost is 19.65 ms and 28.39 ms in REE and TEE, respectively. Almost all schemes encrypt the data using hybrid encryption, in which the symmetric encryption algorithm encrypts the data and the asymmetric algorithm encrypts the key of the symmetric encryption. In this article, we count the cost of asymmetric encryption only. As for the online phase, there is no difference between this article and the traditional schemes, such as the TLS scheme commonly used on the web. The asymmetric encryption stage is also the part that can be operated offline, as mentioned in Section 7.

Figure 7(b) shows the time consumption of the *ReKeyGen* algorithm, which is executed in TEE. Due to re-encryption, keys are generated by messaging infrastructures with TEE support. We also evaluated the *ReKeyGen* algorithm in the TEE. In fact, the re-encryption key generation is an encryption operation on encryption key with new decryption set. Therefore, the results show that the re-encryption key generation time increases as the decryption set size increases and is similar to the data encryption time. In addition, this algorithm is executed offline.

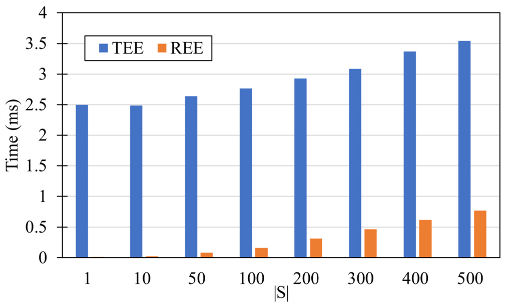
Figure 7(c) shows the time consumption of the *ReEnc* algorithm. The experimental results show that the time of online operation is independent of the set size, while the time of offline operation is linearly related to the set size. Moreover, we have proved the security of the algorithm, so that the re-encryption algorithm is actually executed directly under the REE. By optimizing online and offline, a lot of time consumption can be reduced.

As described in the Section 7, some operations of the *Dec-I* and *Dec-II* algorithms can be performed offline. Figure 8(a) illustrates the time consumption of this part of the computation. Since this part of the computation calculates the information associated with the set S, the time consumption is also linearly related to the set S size. Again, we have evaluated in different environments, that is, the TEE and REE. In general, performance under the REE is significantly higher than that under the TEE, mainly due to the fact that there is a fixed time consumption for the TEE mode switch. In addition, this part of the computation does not involve sensitive information; thus, it is practically enabled to compute in REE.

Figure 8(b) illustrates the relationship between the online time and the number of re-encryptions. The symmetric decryption time for obtaining plaintexts is not shown here. The time to decrypt



(a) *Dec* time vs *ReEnc*



(b) *Dec* time vs decryption set sizes

Fig. 8. Time consumption of the *Dec-I* and *Dec-II* algorithms in the system.

the message is proportional to the number of re-encryptions and the time to decrypt in the TEE is significantly higher than the time to decrypt in the REE.

10 CONCLUSION AND FUTURE WORK

In this article, we analyze the security requirements of data transmission in an edge collaboration environment and propose an end-to-end security scheme based on proxy re-encryption and broadcast encryption. Then, it is integrated into an edge collaboration system, and the functionality and performance of the scheme are further optimized. Through theoretical and experimental analysis, the proposed scheme can achieve end-to-end security in a multi-hop environment and support the features of broadcast encryption, thus addressing the proposed environment's requirements. The performance of the proposed scheme is analyzed theoretically and compared with an existing scheme, hPRESS. The results show that the proposed scheme can effectively relieve the burden of the infrastructure in the process of re-encryption. Finally, the scheme of this article is implemented in the system and the performance is measured.

In the future, we will mainly consider improving performance in two ways. On the one hand, we intend to design more lightweight broadcast encryption and proxy re-encryption and combine them to obtain an efficient scheme. On the other hand, we will still consider optimization from the system level. We found that although broadcastability has some advantages in terms of bandwidth in wireless communication and multicast scenarios, the advantage is not obvious in unicast scenarios. Therefore, we are considering designing unicast-oriented end-to-end security schemes and a secure transmission system to dynamically select the optimal end-to-end scheme, such as the proposed scheme or unicast-oriented end-to-end scheme, by comprehensively evaluating the network environment, latency, and computational performance.

ACKNOWLEDGMENTS

The authors are very grateful to the anonymous referees for their detailed comments and suggestions regarding this paper.

REFERENCES

[1] MIRACL Core. 2022. Retrieved April 22, 2022 from https://github.com/miracl/core.
[2] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. 2006. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security* 9, 1 (Feb. 2006), 1–30. https://doi.org/10.1145/1127345.1127346
[3] Matt Blaze, Gerrit Bleumer, and Martin Strauss. 1998. Divertible protocols and atomic proxy cryptography. In *Advances in Cryptology — (EUROCRYPT'98)*, Kaisa Nyberg (Ed.). Springer, Berlin, 127–144.

[4] Cristian Borcea, Arnab “Bobby” Deb Gupta, Yuriy Polyakov, Kurt Rohloff, and Gerard Ryan. 2017. PICADOR: End-to-end encrypted Publish-Subscribe information distribution with proxy re-encryption. *Future Generation Computer Systems* 71 (2017), 177–191. https://doi.org/10.1016/j.future.2016.10.013

[5] Ran Canetti and Susan Hohenberger. 2007. Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*. ACM, New York, NY, 185–194. https://doi.org/10.1145/1315245.1315269

[6] Cheng-Kang Chu and Wen-Guey Tzeng. Identity-based proxy re-encryption without random oracles. In *International Conference on Information Security* (2007). Springer, 189–202.

[7] Cheng-Kang Chu, Jian Weng, Sherman S. M. Chow, Jianying Zhou, and Robert H. Deng. Conditional proxy broadcast re-encryption. In *Australasian Conference on Information Security and Privacy* (2009). Springer, 327–342.

[8] Chunpeng Ge, Zhe Liu, Jinyue Xia, and Liming Fang. 2021. Revocable identity-based broadcast proxy re-encryption for data sharing in clouds. *IEEE Transactions on Dependable and Secure Computing* 18, 3 (2021), 1214–1226. https://doi.org/10.1109/TDSC.2019.2899300

[9] Le Guan, Peng Liu, Xinyu Xing, Xinyang Ge, Shengzhi Zhang, Meng Yu, and Trent Jaeger. 2017. TrustShadow: Secure execution of unmodified applications with ARM trustzone. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys'17)*. ACM, New York, NY, 488–501. https://doi.org/10.1145/3081333.3081349

[10] Qiang He, Cheng Wang, Guangming Cui, Bo Li, Rui Zhou, Qingguo Zhou, Yang Xiang, Hai Jin, and Yun Yang. 2021. A game-theoretical approach for mitigating edge DDoS attack. *IEEE Transactions on Dependable and Secure Computing* (2021), 1–1. https://doi.org/10.1109/TDSC.2021.3055559

[11] Mihaela Ion, Giovanni Russello, and Bruno Crispo. 2010. Supporting publication and subscription confidentiality in pub/sub networks. In *Security and Privacy in Communication Networks*, Sushil Jajodia and Jianying Zhou (Eds.). Springer, Berlin, 272–289.

[12] Bo Li, Qiang He, Feifei Chen, Haipeng Dai, Hai Jin, Yang Xiang, and Yun Yang. 2021. Cooperative assurance of cache data integrity for mobile edge computing. *IEEE Transactions on Information Forensics and Security* 16 (2021), 4648–4662. https://doi.org/10.1109/TIFS.2021.3111747

[13] Bo Li, Qiang He, Feifei Chen, Hai Jin, Yang Xiang, and Yun Yang. 2021. Auditing cache data integrity in the edge computing environment. *IEEE Transactions on Parallel and Distributed Systems* 32, 5 (2021), 1210–1223. https://doi.org/10.1109/TPDS.2020.3043755

[14] Fang Liu, Guoming Tang, Youhuizi Li, Zhiping Cai, Xingzhou Zhang, and Tongqing Zhou. 2019. A survey on edge computing systems and tools. *Proceedings of IEEE* 107, 8 (2019), 1537–1562. https://doi.org/10.1109/JPROC.2019.2920341

[15] Qin Liu, Guojun Wang, and Jie Wu. 2014. Time-based proxy re-encryption scheme for secure data sharing in a cloud environment. *Information Sciences* 258 (2014), 355–370. https://doi.org/10.1016/j.ins.2012.09.034

[16] Frank McKeen, Ilya Alexandrovich, Ittai Anati, Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, and Carlos Rozas. 2016. Intel®software guard extensions (Intel®SGX) support for dynamic memory management inside an enclave. In *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016 (HASP'16)*. ACM, New York, NY, Article 10, 9 pages. https://doi.org/10.1145/2948618.2954331

[17] Partha Pal, Greg Lauer, Joud Khoury, Nick Hoff, and Joe Loyall. 2012. P3S: A privacy preserving publish-subscribe middleware. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2012), Vol. 7662. Springer, Berlin, 476–495. https://doi.org/10.1007/978-3-642-35170-9_24

[18] Shrideep Pallickara, Marlon Pierce, Harshawardhan Gadgil, Geoffrey Fox, Yan Yan, and Yi Huang. A framework for secure end-to-end delivery of messages in publish/subscribe systems. In *Proceedings of the IEEE/ACM International Workshop on Grid Computing* (2006). 215–222. https://doi.org/10.1109/ICGRID.2006.311018

[19] Yuriy Polyakov, Kurt Rohloff, Gyana Sahu, and Vinod Vaikuntanathan. 2017. Fast proxy re-encryption for publish/subscribe systems. 20, 4, Article 14 (2017), 31 pages. https://dl.acm.org/doi/10.1145/3128607.

[20] M. A. Rajan, Ashley Varghese, N. Narendra, Meena Singh, V. L. Shivraj, Girish Chandra, and P. Balamuralidhar. 2016. Security and privacy for real time video streaming using hierarchical inner product encryption based publish-subscribe architecture. In *30th International Conference on Advanced Information Networking and Applications Workshops (WAINA'16)*. IEEE, 373–380. https://doi.org/10.1109/WAINA.2016.101

[21] Yanli Ren and Dawu Gu. 2009. Fully CCA2 secure identity based broadcast encryption without random oracles. *Inform. Process. Lett.* 109, 11 (2009), 527–533. https://doi.org/10.1016/j.ipl.2009.01.017

[22] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. 2015. Trusted execution environment: What it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/ISPA*, Vol. 1. IEEE, 57–64.

[23] Mahadev Satyanarayanan. 2017. The emergence of edge computing. *Computer* 50, 1 (2017), 30–39. https://doi.org/10.1109/MC.2017.9

[24] Jun Shao, Zhenfu Cao, Xiaohui Liang, and Huang Lin. 2010. Proxy re-encryption with keyword search. *Information Sciences* 180, 13 (2010), 2576–2587. https://doi.org/10.1016/j.ins.2010.03.026

[25] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge computing: Vision and challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

[26] Maria Stoyanova, Yannis Nikoloudakis, Spyridon Panagiotakis, Evangelos Pallis, and Evangelos K. Markakis. 2020. A survey on the Internet of Things (IoT) forensics: Challenges, approaches, and open issues. *IEEE Communications Surveys Tutorials* 22, 2 (2020), 1191–1221. https://doi.org/10.1109/COMST.2019.2962586

[27] Bowen Wang, Yanjing Sun, Dianxiong Liu, Hien M. Nguyen, and Trung Q. Duong. 2020. Social-aware UAV-assisted mobile crowd sensing in stochastic and dynamic environments for disaster relief networks. *IEEE Transactions on Vehicular Technology* 69, 1 (2020), 1070–1074. https://doi.org/10.1109/TVT.2019.2949634

[28] L. Wang, Q. Zhang, Y. Li, H. Zhong, and W. Shi. MobileEdge: Enhancing on-board vehicle computing units using mobile edges for CAVs. In *IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS'19)*. 470–479.

[29] Xu An Wang, Jian Weng, Xiaoyuan Yang, and Yanjiang Yang. 2011. Cryptanalysis of an identity based broadcast encryption scheme without random oracles. *Information Processing Letters* 111, 10 (2011), 461–464. https://doi.org/10.1016/j.ipl.2011.02.007

[30] Jian Weng, Yanjiang Yang, Qiang Tang, Robert H. Deng, and Feng Bao. Efficient conditional proxy re-encryption with chosen-ciphertext security. In *International Conference on Information Security* (2009). Springer, 151–166.

[31] Peng Xu, Tengfei Jiao, Qianhong Wu, Wei Wang, and Hai Jin. 2016. Conditional identity-based broadcast proxy re-encryption and its application to cloud email. *IEEE Trans. Comput.* 65, 1 (2016), 66–79. https://doi.org/10.1109/TC.2015.2417544

[32] Liang Yuan, Qiang He, Siyu Tan, Bo Li, Jiangshan Yu, Feifei Chen, Hai Jin, and Yun Yang. 2021. CoopEdge: A decentralized blockchain-based platform for cooperative edge computing. In *Proceedings of the Web Conference 2021 (WWW'21)*. ACM, New York, NY, 2245–2257. https://doi.org/10.1145/3442381.3449994

[33] Fan Zhang, Ziyuan Liang, Cong Zuo, Jun Shao, Jianting Ning, Jun Sun, Joseph K. Liu, and Yibao Bao. 2021. hPRESS: A hardware-enhanced proxy re-encryption scheme using secure enclave. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 6 (2021), 1144–1157. https://doi.org/10.1109/TCAD.2020.3022841

[34] Qingyang Zhang, Hui Sun, Xiaopei Wu, and Hong Zhong. 2019. Edge video analytics for public safety: A review. *Proceedings of the IEEE* 107, 8 (2019), 1675–1696. https://doi.org/10.1109/JPROC.2019.2925910

[35] Qingyang Zhang, Quan Zhang, Weisong Shi, and Hong Zhong. 2018. Distributed collaborative execution on the edges and its application to AMBER alerts. *IEEE Internet of Things Journal* 5, 5 (2018), 3580–3593. https://doi.org/10.1109/JIOT.2018.2845898

[36] Qingyang Zhang, Hong Zhong, Weisong Shi, and Lu Liu. 2021. A trusted and collaborative framework for deep learning in IoT. *Computer Networks* 193 (2021), 108055. https://doi.org/10.1016/j.comnet.2021.108055

Received October 2021; revised January 2022; accepted March 2022