AP-PRE: Autonomous Path Proxy Re-Encryption and Its Applications

Zhenfu Cao, *Senior Member, IEEE*, Hongbing Wang, and Yunlei Zhao

Abstract—In this paper, we introduce a new cryptographic primitive, called autonomous path proxy re-encryption (AP-PRE), which is motivated by several application scenarios where the delegator would like to control the whole delegation path in a multi-hop delegation process. Compared with the traditional proxy re-encryption, AP-PRE provides much better fine-grained access control to delegation path. Briefly speaking, in an AP-PRE scheme, the delegator designates a path of his preferred delegatees. The path consists of several delegatees with the privilege from high to low. If the delegatee in the path cannot complete the decryption, the decryption right is automatically delegated to the next one in the path. In this way, the delegator can ensure that the delegation has always been done among those delegatees the delegator trusts. Moreover, an AP-PRE scheme has to obey the following path rules. The delegation, for ciphertexts of a delegator i, can only be carried out on the autonomous path Pa_i designated by the delegator i, in the sense that (1) re-encrypted ciphertexts along the autonomous path Pa_i cannot branch off Pa_i with meaningful decryption, and (2) original ciphertexts generated under pk_j for $j \neq i$ (i.e., for a path Pa_j different from Pa_i) cannot be inserted into (i.e., cannot be transformed along) the autonomous path Pa_i with meaningful decryption. We give the formal definition, as well as the formal security model, for this cryptographic primitive. Under this concept, we construct an IND-CPA secure AP-PRE scheme under the decisional bilinear Diffie-Hellman (DBDH) assumption in the random oracle model. Our scheme is with the useful properties of proxy re-encryption, i.e., unidirectionality and multi-hop.

Index Terms—Autonomous path, proxy re-encryption, unidirectional, multi-hop, IND-CPA, decisional bilinear Diffie-Hellman

1 Introduction

When one is too busy to deal with all his encrypted files, he may wish to delegate his decryption rights to someone he trusts. This delegation of the power to decrypt the ciphertext [1] can be easily done if the delegator is online-simply decrypts the ciphertext and re-encrypts the plaintext with the public key of whom he trusts. However, this is not always practical, for the delegator may not be online all the time. And, it is undesirable to just disclose the secret key to some untrusted server to do the transformation of the ciphertext. To solve the above mentioned problems, at Eurocrypt'98, Blaze, Bleumer and Strauss [2] first proposed the concept of proxy re-encryption (PRE). In a PRE scheme, a semi-trusted proxy with some additional information (re-encryption key, which is computed by the delegator in advance) can convert a ciphertext computed under Alice's (delegator's) public-key into one intended to Bob (delegatee) with the same plaintext. The fundamental property of proxy re-encryption schemes is that the proxy is not fully trusted, i.e., the proxy should not know the secret keys of Alice or Bob, and should not learn the plaintext during the conversion. Blaze et al. [2] gave two methods to classify PRE schemes. One is according to the allowed times of transformation. If the ciphertext can be transformed more than one time, i.e., from Alice to Bob, then from Bob to Carol, and so on, we call the PRE scheme multi-hop; otherwise, it is single-hop. The other classification is according to the allowed direction of the transformation. If the re-encryption key can be used to transform the ciphertext from Alice to Bob, and vice versa, we call the PRE scheme bidirectional; otherwise, it is unidirectional.

A unidirectional PRE scheme is more practical than a bidirectional one. However, construction of a unidirectional PRE scheme is more difficult than that of a bidirectional one. It is easily observed that any unidirectional PRE scheme can be converted to a bidirectional one by running the former in both directions. In many cases, the unidirectional PRE scheme is especially needed. For example, the delegator delegates his decryption rights to his delegatee, but the delegatee does not always want to do the reverse delegation. Meanwhile, multi-hop is an important and practical property for a PRE scheme. For example, if a delegatee happens to be very busy, or just cannot be online when he receives a re-encrypted ciphertext from his delegator, he would like to delegate his decryption rights to other delegatees. If the PRE scheme is single-hop, then the delegatee cannot transform the re-encrypted file furthermore. In this case, a multi-hop PRE is desired. PRE has more attributes besides the above two, and the reader is referred to [3], [4] for more details.

The multi-hop property of a PRE scheme is described in Fig. 1. We note that, in Fig. 1, the delegator with public key

- *Z. Cao is with the Software Engineering Institute, East China Normal University, Shanghai, 200062, China. E-mail: zfcao@sei.ecnu.edu.cn.*
- *H. Wang is with School of Computer Science, Fudan University, Shanghai, 200433, China. E-mail: wanghongbing@fudan.edu.cn.*
- *Y. Zhao is with Shanghai Key Laboratory of Data Science, Software School, Fudan University, Shanghai, 200433, China, State Key Laboratory of Cryptology, Beijing, 100000, China. E-mail: ylzhao@fudan.edu.cn.*

Manuscript received 24 Jan. 2017; revised 9 May 2017; accepted 24 May 2017. Date of publication 9 June 2017; date of current version 30 Aug. 2019. (Corresponding author: Yunlei Zhao.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TDSC.2017.2714166

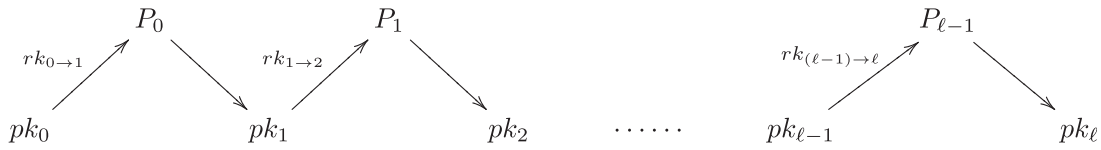

Fig. 1. Multi-hop property of PRE scheme.

pk_0 designates his delegatee pk_1. The delegatee pk_1 designates his own delegatee pk_2. There is maybe no relationship between pk_0 and pk_2, since no restrictions are specified when pk_1 selects his delegatee. In most of the practical scenarios, the delegator is reluctant about his delegatee's selection. The delegator may want to designate another delegatee by himself if his delegatee pk_1 is unable to decrypt the ciphertext. We also find that the more times the PRE scheme hops, the lower the trust degree becomes between the delegator and the delegatee. It is thus desirable to construct a flexible PRE scheme, where the delegator could control the next delegatee if the delegatee of his first choice is unable to complete the delegation.

In order to meet the above discussed applications, in this work we introduce a new cryptographic primitive, called autonomous path proxy re-encryption (AP-PRE). Briefly speaking, in an AP-PRE scheme, the delegator designates a path of his preferred delegatees. The path consists of several delegatees with the privilege from high to low. If the delegatee in the path cannot complete the decryption, the decryption right is automatically delegated to the next one in the path. In this way, the delegator can ensure that the delegation has always been done among those delegatees the delegator trusts.

Moreover, an AP-PRE scheme has to obey the following path rules. The delegation, for ciphertexts of a delegator i, can only be carried out on the autonomous path Pa_i designated by the delegator i, in the sense that (1) re-encrypted ciphertexts along the autonomous path Pa_i cannot branch off Pa_i with meaningful decryption, and (2) original ciphertexts generated under pk_j for $j \neq i$ (i.e., for a path Pa_j different from Pa_i) cannot be inserted into (i.e., cannot be transformed along) the autonomous path Pa_i with meaningful decryption.

We give the formal definition, as well as the formal security model, for this cryptographic primitive. Under this concept, we construct an IND-CPA secure AP-PRE scheme under the decisional bilinear Diffie-Hellman (DBDH) assumption in the random oracle (RO) model. Our scheme is with the useful properties of proxy re-encryption, i.e., unidirectionality and multi-hop. Generally speaking, the hardness in constructing an AP-PRE scheme lies in that, for the transformed ciphertext, the delegator should have the ability to compute a re-encryption key; The re-encryption key is used for the proxy to transform the re-encrypted ciphertext under one of his delegatee's public key to another delegatee, while the delegator actually does not know the private key of any of his delegatees.

1.1 Applications

In a traditional PRE, once the proxy holds a re-encryption key from a delegator A to a delegatee B, it can transform all the ciphertexts of A to B with the same messages. While, in some special cases, this is not desired. So, some fine-grained PREs are developed, such as conditional PREs [5], [6], [7], and tag-based PREs [8]. An AP-PRE scheme is another fine-grained PRE, which differs from the conditional PRE and the tag-based PRE in the sense that it provides autonomous path management. The following paragraphs list some applications of AP-PRE.

- *Electronic medical database:* Many countries are using national electronic medical database, and considering to put these big data in cloud. Now, we consider the following scenarios: Patient A wants to make an outpatient reservation in hospital H. A is sure to have a doctor list beginning from his most favorite doctor. Since, making an outpatient reservation needs some sensitive information of patient's, such as the patient's identity number, mobile phone, home address, etc.. In order to prevent the sensitive information from being exposed, A encrypts the information using his own public key, and generates a priority path and the corresponding re-encryption keys of his path. Finally, these encrypted data are sent to the proxies. We suppose A's choice of the doctors from high priority to low is $D_1 \to D_2 \to \ldots \to D_n$. If at A's appointment time, D_1 happens to have no time or have other important appointment, then A's reservation is automatically transferred to D_2 by the proxy, and so on.
- *Data sharing or trading in cloud:* Fig. 2 shows how an AP-PRE works in cloud. Suppose Alice has some data to share or trade. She first encrypts the data using her own public key, and transforms the resulted ciphertext to the secure cloud server along with a delegation path table which includes an ordered delegatees and the corresponding re-encryption keys. On receiving the ciphertext and path table from Alice, the cloud proxy finds out the first record from the path table, then transforms the ciphertext to Bob using the re-encryption key $rk_{a\to b}$. If Bob gives up the decryption right, then the cloud proxy finds out the second record from the path table, and produces a new ciphertext on input the ciphertext re-encrypted to Bob. This continues until a delegatee is willing to accept the decryption right for data sharing or trading.

1.2 Related Works

Since the concept of PRE was introduced by Blaze et al. [2] at Eurocrypt'98, there have been many papers [2], [3], [4], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22] that have proposed PRE schemes with different properties to meet multifarious application demands. In a traditional proxy re-encryption, once the proxy gets the re-encryption key from the delegator, it can transform all the ciphertexts of the delegator to the delegatee, and the

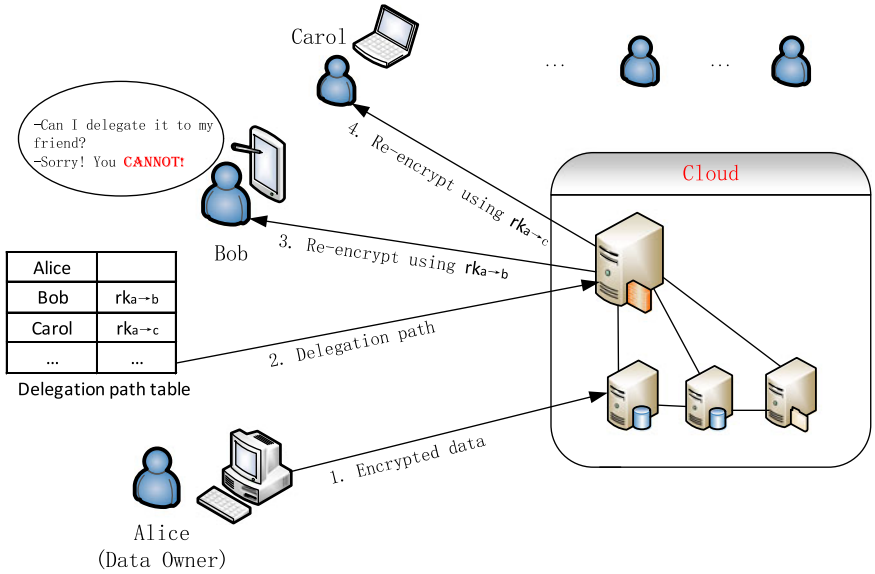

Fig. 2. Application of AP-PRE in cloud.

delegatee can decrypt all ciphertexts for the delegator after re-encryption by the proxy. This is not desired in many scenarios. In order to implement fine-grained access control policies, two variants of PREs were proposed, type-based proxy re-encryption [8] and conditional proxy re-encryption [5], [6], [7], which can provide fine-grained data access. In the following paragraph, we spend some time on describing the characters of these PREs.

Type-Based Proxy Re-Encryption(TB-PRE). In a type-based proxy re-encryption [8] scheme, the delegator categorizes his messages (ciphertexts) into different subsets, for each of these subsets, the delegator delegates the decryption rights of this subset to a specific delegatee. The ciphertexts for the delegator are generated based on the delegator's public key, and the message type is used to identify the message subset. Type-based proxy re-encryption enables the delegator to choose a particular proxy for a specific delegatee, which might be based on the sensitiveness of the delegation.

Conditional Proxy Re-Encryption(C-PRE). In a conditional proxy re-encryption [5], [6], [7], some condition is attached to a message, so the ciphertext for the delegator is the encryption of the combination of the message and the condition. With the re-encryption key generated by the delegator, the proxy can only transform the delegator's ciphertexts which satisfy the condition set by the delegator. The condition can be a single keyword, or a set of descriptive keywords [7]. Another generalization of C-PRE is conditional proxy broadcast re-encryption (CPBRE) [6]. In a CPBRE, the proxy can re-encrypt the ciphertext for a set of users at a time.

Though the above two PREs have different names, they are the same in spirit. Both of them provide fine-grained data access compared with the traditional PREs, but, neither of them can ensure that the delegation always be controlled by the delegator. In this sense, they are similar to the traditional PREs, i.e., the delegator could only control the selection of the first delegatee whom he delegates the decryption rights to. When the re-encrypted ciphertext is further transformed, the delegator has no idea of those delegatees. It is undesirable in most of the real applications, where the delegator wants to ensure that the delegation has always been transformed to those delegatees he trusts. For example, if the first delegatee a delegator designated is impossible to decrypt the ciphertext, the delegator can ensure that the delegation proceeds by the next delegatee he wishes. This ensures that the delegator's encrypted files can always be decrypted by those he trusts.

Graded encryption (GE) [23] can be applied to some scenarios that a proxy re-signature can be done. In a GE scheme, there is one central (mostly offline) authority and a number of sub-authorities holding master keys that correspond to different levels. GE works in the way, if a user succeeds in one stage, he gets an updated key for the next stage. In spirit, GE is not for ciphertext transformation, so it is different from PRE. Meanwhile, in GE, the multiple authorities are with different levels, in contrast, all proxies act in a same level in PRE. Also, as indicated in [23], GE can be elegantly used to prove that a certain path was taken in a graph that proxy re-signatures but not proxy re-encryptions are applicable.

Another cryptographic primitive for delegating the decryption power of some ciphertexts without sending the secret key to the delegatee, is key-aggregate cryptosystem (KAC) [24]. In KAC, users encrypt a message not only under a public key, but also under an identifier of ciphertext class. Using the master-secret key which is hold by the key owner, the owner extracts secret keys for different classes. KAC is a public key encryption scheme which supports flexible delegation in the sense that any subset of the ciphertexts is decryptable by a decryption key, which is generated by the owner of the master-key. Though KAC has the same meaning with PRE in delegating the decryption power, its delegation can be performed only once. In this sense, it can only be compared with a single-hop PRE.

An AP-PRE is different from a broadcast encryption. In a broadcast encryption, the data owner chooses all the recipients and uses their public keys to produce the ciphertexts for them in one algorithm. While, an encryption algorithm in an AP-PRE scheme only encrypts message for one recipient once a time. In an AP-PRE, the delegatee can obtain the ciphertext from the proxy directly without interaction with

the data owners. In an AP-PRE, once a delegatee accepts the decryption right, the delegation stops.

Moreover, an AP-PRE cannot be realized by simply running multiple single-hop PREs. According to our definition, the delegation, for ciphertexts of a delegator i, can only be carried out on the autonomous path Pa_i designated by the delegator i, in the sense that (1) re-encrypted ciphertexts along the autonomous path Pa_i cannot branch off Pa_i with meaningful decryption, and (2) original ciphertexts generated under pk_j for $j \neq i$ (i.e., for a path Pa_j different from Pa_i) cannot be inserted into (i.e., cannot be transformed along) the autonomous path Pa_i with meaningful decryption. For examples, for a re-encrypted ciphertext $c_{i_j}^i$ under user i_j on path Pa_i, if i_j generates a re-encryption key from him to another delegatee which is different from i_{j+1} (user i_{j+1} is the next delegatee of user i_j in path Pa_i), he cannot produce a meaningful ciphertext $c_{j_k}^i$ by executing the re-encryption algorithm on input $c_{i_j}^i$ and his newly generated re-encryption key, and if such ciphertext $c_{j_k}^i$ is inserted into a path which is initiated by i_j, no correct message could be recovered. Suppose that the autonomous path designated by delegator user i is denoted by $\mathsf{Pa}_i = (pk_{i_0} = pk_i, pk_{i_1}, \ldots, pk_{i_{\ell_i}})$, which is a sequence of ordered ℓ_i different public keys, where pk_i is denoted as pk_{i_0}, for any $j, 0 \leq j \leq \ell_i$, $i_j \in \{1, \ldots, n\}$, and $pk_j \neq pk_k$ for any $0 \leq j \neq k \leq \ell_i$. An AP-PRE cannot be realized by simply running the following single-hop PREs, from pk_i to $pk_{i_1}, \ldots$, from $pk_{i_{\ell_i-1}}$ to $pk_{i_{\ell_i}}$, since no users different from pk_i can produce meaningful ciphertext by executing AP-PRE algorithms (for $pk_j, j \neq i$, pk_j cannot transform the re-encrypted ciphertext produced on path Pa_i with a re-encryption key generated by him).

1.3 Paper Organization

The remainder of this paper is organized as follows. In Section 2, we introduce the preliminary knowledge of assumption which our AP-PRE scheme is based on. We then describe the concept of our autonomous path proxy re-encryption and its security model in Section 3. In Section 4, the construction of our IND-APPRE-CPA secure autonomous path proxy re-encryption scheme is presented, followed by security analysis in Section 5. Finally, we conclude this paper in Section 6.

2 Preliminaries

We use standard notations and conventions below for writing probabilistic algorithms, experiments and interactive protocols. If $\mathcal{D}$ denotes a probability distribution, $x \leftarrow \mathcal{D}$ is the operation of picking an element according to $\mathcal{D}$. If S is a finite set then $|S|$ is its cardinality, and $x \leftarrow S$ or $x \leftarrow_R S$ is the operation of picking an element uniformly at random from $\mathcal{S}$. If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement. If A is a probabilistic algorithm, then $A(x_1, x_2, \ldots; r)$ is the result of running A on inputs $x_1, x_2, \ldots$ and coins r. We let $y \leftarrow A(x_1, x_2, \ldots)$, or $A(x_1, x_2, \ldots) \rightarrow y$, denote the experiment of picking r at random and letting y be $A(x_1, x_2, \ldots; r)$. By $\Pr[R_1; \ldots; R_n : E]$ we denote the probability of event E, after the ordered execution of random processes $R_1, \ldots, R_n$.

We say that a function $f(\kappa)$ is *negligible*, if for every $c > 0$ there exists an κ_c such that $f(\kappa) < 1/\kappa^c$ for all $\kappa > \kappa_c$. Two distribution ensembles $\{X(\lambda, z)\}_{\kappa \in N, z \in \{0,1\}^*}$ and $\{Y(\lambda, z)\}_{\kappa \in N, z \in \{0,1\}^*}$ are computationally indistinguishable, if for any probabilistic polynomial-time (PPT) algorithm D, and for sufficiently large κ and any $z \in \{0,1\}^*$, it holds $|\Pr[D(\kappa, z, X) = 1] - \Pr[D(\kappa, z, Y) = 1]|$ is negligible in κ.

2.1 Bilinear Map and Computational Assumption

Definition 1 (Bilinear Map). *Let $\mathbb{G}$ and $\mathbb{G}_1$ be two multiplicative groups of the same prime order q, and g be a generator of $\mathbb{G}$. Assume that the discrete logarithm problems in both $\mathbb{G}$ and $\mathbb{G}_1$ are intractable. We say that $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ is a bilinear or pairing map if it satisfies the following properties:*

(1) Bilinear: *For all $a, b \in \mathbb{Z}_q^*$, $g \in \mathbb{G}$, $e(g^a, g^b) = e(g, g)^{ab}$.*
(2) Non-degenerate: *$e(g, g) \neq 1_{\mathbb{G}_1}$, i.e., if g generates $\mathbb{G}$, then $e(g, g)$ generates $\mathbb{G}_1$.*
(3) Computable: *The map e is efficiently computable.*

We denote $\mathsf{BSetup}(1^\kappa)$ as an algorithm that, on input the security parameter 1^κ, outputs the parameters for a bilinear map as $(q, g, \mathbb{G}, \mathbb{G}_1, e)$, where $|q| = \kappa$.

Definition 2 (Decisional Bilinear Diffie-Hellman (DBDH) Problem). *Let $\mathbb{G}$ and $\mathbb{G}_1$ be two multiplicative groups of the same prime order q, and g be a generator of $\mathbb{G}$. Suppose that $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ is a bilinear map. The decisional bilinear Diffie-Hellman (DBDH) problem is to decide, given a tuple of values $(g, g^a, g^b, g^c, T) \in \mathbb{G}^4 \times \mathbb{G}_1$ (where $a, b, c \in_R \mathbb{Z}_q^*$), whether $T = e(g, g)^{abc}$ holds.*

Let $\kappa = |q|$ be the security parameter. Formally, we say that the DBDH assumption holds in $\mathbb{G}, \mathbb{G}_1$, if for any PPT algorithms $\mathcal{A}$, the following quantity is negligible (in κ)

$$\left| \begin{array}{l} \mathsf{Pr}[a, b, c \leftarrow_R \mathbb{Z}_q^* : 1 \leftarrow \mathcal{A}(g, g^a, g^b, g^c, e(g, g)^{abc})] - \\ \mathsf{Pr}[a, b, c \leftarrow_R \mathbb{Z}_q^* : T \leftarrow_R \mathbb{G}_1; 1 \leftarrow \mathcal{A}(g, g^a, g^b, g^c, T)] \end{array} \right|.$$

3 Definition and Security Model for Autonomous Path Proxy Re-Encryption

3.1 Autonomous Path Proxy Re-Encryption (AP-PRE)

An autonomous path proxy re-encryption is a new kind of unidirectional and multi-hop proxy re-encryption, where the delegator controls the selection of all his delegatees in a delegation path. The delegator has a set of delegatees with privilege order, and generates re-encryption keys for these delegatees. The re-encryption keys are sent to the corresponding proxies via a secure channel. The delegation, for ciphertexts of a delegator i, can only be carried out on the autonomous path Pa_i designated by the delegator i, in the sense that (1) re-encrypted ciphertexts along the autonomous path Pa_i cannot branch off Pa_i with meaningful decryption; and (2) original ciphertexts generated under pk_j for $j \neq i$ (i.e., for a path Pa_j different from Pa_i) cannot be inserted into (i.e., cannot be transformed along) the autonomous path Pa_i with meaningful decryption. (See Section 1.2.)

Definition 3 (Autonomous Path Proxy Re-encryption). *An autonomous path proxy re-encryption scheme consists of the following algorithms:*

- $\mathsf{Setup}(1^\kappa) \rightarrow \mathsf{par}$: *On input the system's security parameter κ, it outputs the system's public parameters* par. *The system parameters particularly include a description of a finite message space $\mathcal{M}$ and a description of a ciphertext space $\mathcal{C}$.*
- $\mathsf{KeyGen}(\mathsf{par}, i) \rightarrow (pk_i, sk_i)$: *On input the system's public parameters* par *and a user's identity $i \in \{1, \ldots, n\}$, it outputs the public and private key pair (pk_i, sk_i) for user $i, 1 \leq i \leq n$, where n is the number of users in the system.*
- $\mathsf{CreatPath}(\mathsf{par}, pk_i) \rightarrow (\mathsf{Pa}_i, \ell_i)$: *On input the system's public parameters* par, *a public key pk_i of a delegator user i, it outputs an autonomous delegation path Pa_i of length ℓ_i represented by (Pa_i, ℓ_i). The autonomous path designated by delegator user $i, \mathsf{Pa}_i = (pk_{i_0} = pk_i, pk_{i_1}, \ldots, pk_{i_{\ell_i}})$, is a sequence of ordered ℓ_i different public keys, where pk_i is denoted to be pk_{i_0}, $i_j \in \{1, \ldots, n\}$ for any $j, 0 \leq j \leq \ell_i$, and $pk_j \neq pk_k$ for any $0 \leq j \neq k \leq \ell_i$. A sequence of ordered public-keys $\mathsf{Pa}^i_{j \rightarrow k} = (pk_{i_j}, \ldots, pk_{i_k})$ is called a* sub-path *of Pa_i, if $0 \leq j \leq k \leq \ell_i$. In this work, for presentation simplicity, we denote by $pk_{i_j} \in \mathsf{Pa}_i$ a user pk_{i_j} in the path Pa_i, and by $(pk_{i_j}, pk_{i_{j+1}}) \in \mathsf{Pa}_i$ a step from user pk_{i_j} to $pk_{i_{j+1}}$ in the path Pa_i, where $0 \leq j < \ell_i$.*
- $\mathsf{RKGen}(\mathsf{par}, sk_i, \mathsf{Pa}_i) \rightarrow rk^i = (rk^i_{0 \rightarrow 1}, rk^i_{1 \rightarrow 2}, \ldots, rk^i_{\ell_i - 1 \rightarrow \ell_i})$: *On input the system's public parameters* par, *the private key sk_i of the delegator user i, and the autonomous path $\mathsf{Pa}_i = (pk_{i_0} = pk_i, pk_{i_1}, \ldots, pk_{i_{\ell_i}})$ created by i, the algorithm outputs ℓ_i re-encryption keys and sends them to the corresponding proxies via a secure channel, where for any $j, 0 \leq j < \ell_i, rk^i_{j \rightarrow j+1}$ is the re-encryption key from user i_j to i_{j+1} in the autonomous path Pa_i.*
- $\mathsf{Enc}(\mathsf{par}, pk_i, m) \rightarrow c^i_0$, *where pk_i is denoted as pk_{i_0}: On input the system's public parameters* par, *the delegator's public key pk_i, and a message m from the message space $\mathcal{M}$, it outputs the ciphertext c^i_0 under the delegator's public key $pk_i = pk_{i_0}$.*
- $\mathsf{ReEnc}(\mathsf{par}, \mathsf{Pa}_i, pk_{i_j}, pk_{i_{j+1}}, rk^i_{j \rightarrow j+1}, c^i_j) \rightarrow c^i_{j+1}$, *where $0 \leq j < \ell_i$: On input the system's public parameters* par, *a designated path Pa_i, a re-encryption key $rk^i_{j \rightarrow j+1}$ from user i_j to i_{j+1} and a ciphertext c^i_j under the public-key pk_{i_j} of user i_j, this algorithm first checks whether $(pk_{i_j}, pk_{i_{j+1}}) \in \mathsf{Pa}_i$ and outputs "$\perp$" if not. Otherwise, it outputs the re-encrypted ciphertext c^i_{j+1} under the public-key $pk_{i_{j+1}}$ of user i_{j+1}. For presentation simplicity, for any $0 \leq \alpha < \beta \leq \ell_i$ and some ciphertext $\hat{c}$, where $\hat{c}$ denotes an original ciphertext when $\alpha = 0$ and a re-encrypted ciphertext otherwise, we denote by*

$$\mathsf{ReEnc}^i_{\alpha \rightarrow \beta}(\hat{c}) = \mathsf{ReEnc}\Big(\mathsf{par}, \mathsf{Pa}_i, rk^i_{\beta-1 \rightarrow \beta}, \mathsf{ReEnc}\big(\mathsf{par}, \mathsf{Pa}_i, rk^i_{\beta-2 \rightarrow \beta-1}, \ldots, \mathsf{ReEnc}(\mathsf{par}, \mathsf{Pa}_i, rk^i_{\alpha \rightarrow \alpha+1}, \hat{c}) \ldots\big)\Big).$$

- $\mathsf{Dec}(\mathsf{par}, c, sk_i) \rightarrow (m, \perp)$: *On input the system's public parameters* par, *a ciphertext c, and the corresponding private key sk_i of user i, it outputs a message m in the message space $\mathcal{M}$, or an error symbol $\perp$ indicating invalid ciphertext.*

Correctness. We say that an autonomous path proxy re-encryption scheme is *correct* if for any autonomous path Pa_i created by a delegator $i, 1 \leq i \leq n$, and any $j, 1 \leq j \leq \ell_i$, the following equations hold for any $m \in \mathcal{M}$

$$\mathsf{Dec}\big(\mathsf{par}, \mathsf{Enc}(\mathsf{par}, pk_i, m), sk_i\big) = m,$$

$$\mathsf{Dec}\big(\mathsf{par}, c^i_j, sk_{i_j}\big) = m, 1 \leq j \leq \ell_i,$$

where for any $k, 1 \leq k \leq j$,

$$c^i_k = \mathsf{ReEnc}^i_{0 \rightarrow k}(\mathsf{Enc}(\mathsf{par}, pk_i, m)).$$

Let $\mathcal{E}$ be an AP-PRE scheme defined as above. We consider the following game, denoted by $\mathbf{Exp}^{\mathsf{CPA}}_{\mathcal{E},\mathcal{A}}$, in which a PPT adversary $\mathcal{A}$ against an IND-APPRE-CPA secure AP-PRE is involved:

Besides the public key generation oracle, the private key generation oracle, the re-encryption key oracle, and the re-encryption oracle, which are allowed in the security model for traditional PRE, a PPT adversary $\mathcal{A}$ against an IND-APPRE-CPA secure AP-PRE can have access to an additional oracle, the path generation oracle. Next, we define the oracles which a PPT adversary against an AP-PRE scheme can make queries to a challenger.[1] In the following process, the challenger needs to maintain three tables $T_{pk}, \mathcal{RK}$, and $\mathcal{P}$. They are initially empty, and used to record the public keys, the re-encryption keys, and the autonomous paths, respectively.

- *Public key generation oracle* $\mathcal{O}_{pk}(i)$: On input an index $i \in \{1, \ldots, n\}$ by $\mathcal{A}$, the challenger responds $\mathcal{A}$ by running algorithm $\mathsf{KeyGen}(\mathsf{par}, i)$ to generate a key pair (pk_i, sk_i) for user i. The challenger returns pk_i to the adversary $\mathcal{A}$, and records (pk_i, sk_i) in the table T_{pk}.
  We assume that $\mathcal{A}$ has made appropriate $\mathcal{O}_{pk}$ queries before he makes the following queries[2].
- *Private key generation oracle* $\mathcal{O}_{sk}(pk_i)$: On input pk_i by $\mathcal{A}$, where pk_i is from $\mathcal{O}_{pk}$, $i \in \{1, \ldots, n\}$, the challenger searches pk_i in the table T_{pk} and returns the corresponding sk_i to $\mathcal{A}$.
- *Path creation oracle* $\mathcal{O}_{cp}(i, \mathsf{Pa}_i)$, where $1 \leq i \leq n$: If $\mathcal{A}$ has queried $\mathcal{O}_{cp}(i, \mathsf{Pa}'_i)$, the challenger returns "$\perp$" indicating an invalid query. Otherwise (i.e., it is the first time for $\mathcal{A}$ to query $\mathcal{O}_{cp}$ w.r.t. i), the challenger creates the path $\mathsf{Pa}_i = (pk_i = pk_{i_0}, pk_{i_1}, \ldots, pk_{i_{\ell_i}})$ by running algorithm $\mathsf{CreatPath}(\mathsf{par}, pk_i)$, and generates the re-encryption keys $rk^i = (rk^i_{0 \rightarrow 1}, rk^i_{1 \rightarrow 2}, \ldots, rk^i_{\ell_i - 1 \rightarrow \ell_i})$ for the path Pa_i by running algorithm $\mathsf{RKGen}(\mathsf{par}, sk_i, \mathsf{Pa}_i)$. The challenger records Pa_i in the path table $\mathcal{P}$, and rk^i in the re-encryption key table $\mathcal{RK}$. Finally, the challenger returns "Pa_i is created" to $\mathcal{A}$.
- *Re-encryption key generation oracle* $\mathcal{O}_{rk}(i, pk_{i_j}, pk_{i_{j+1}})$: The challenger first checks whether the path table $\mathcal{P}$ contains a path $\mathsf{Pa}_i = (\ldots, pk_{i_j}, pk_{i_{j+1}}, \ldots)$ for user i. If not, the challenger outputs "$\perp$" indicating an invalid query. Otherwise, the challenger retrieves $rk^i_{j \rightarrow j+1}$ from rk^i in the table $\mathcal{RK}$, and returns $rk^i_{j \rightarrow j+1}$ to $\mathcal{A}$.

1. For PRE in the random oracle model, the adversary also gets access to a random oracle H that is programmed by the challenger.

2. The assumption is to make sure that the adversary has grasped the public key of a user before he makes further queries about that user.

- *Re-encryption oracle* $\mathcal{O}_{reen}(i, pk_{i_j}, pk_{i_{j+1}}, c_j^i)$: The challenger first checks if the path table $\mathcal{P}$ contains the path $\mathsf{Pa}_i = (\ldots, pk_{i_j}, pk_{i_{j+1}}, \ldots)$ for user i. If not, it returns "$\perp$" to $\mathcal{A}$ indicating an invalid query. Otherwise, the challenger retrieves $rk^i_{j\to j+1}$ from rk^i in the table $\mathcal{RK}$, computes $c^i_{j+1} = \mathsf{ReEnc}(\mathsf{par}, \mathsf{Pa}_i, rk^i_{j\to j+1}, c^i_j)$, and returns c^i_{j+1} back to $\mathcal{A}$.

Next, we define a game which is run between a PPT adversary $\mathcal{A}$ and the challenger $\mathcal{C}$. The adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, who works in two stages "find" and "guess", is described in the following experiment

Experiment $Exp^{\mathsf{CPA}}_{\mathcal{E},\mathcal{A}}(1^\kappa)$

1. $\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$;
2. $(pk_i, \mathsf{Pa}_i, pk_j, m_0, m_1, st)$ $\leftarrow \mathcal{A}_1^{\mathcal{O}_{pk}(\cdot),\mathcal{O}_{sk}(\cdot),\mathcal{O}_{cp}(\cdot),\mathcal{O}_{rk}(\cdot),\mathcal{O}_{reen}(\cdot)}$ (find, par), where $|m_0| = |m_1|$, $pk_i \neq pk_j$, $pk_j = pk_{i_\mu}$ for some $\mu, 1 \le \mu \le \ell_i$, and st is some state information;
3. $d \leftarrow_R \{0,1\}$;
4. $c_0^{i*} = \mathsf{Enc}(\mathsf{par}, pk_i, m_d)$, $c_0^{j*} = \mathsf{Enc}(\mathsf{par}, pk_j, m_d)$;
5. $d' \leftarrow \mathcal{A}_2^{\mathcal{O}_{pk}(\cdot),\mathcal{O}_{sk}(\cdot),\mathcal{O}_{cp}(\cdot),\mathcal{O}_{rk}(\cdot),\mathcal{O}_{reen}(\cdot)}$ (guess, par, $c^* = (c_0^{i*}, c_0^{j*}), st$);
6. return d'.

During the above experiment, some restrictions are imposed upon the adversary $\mathcal{A}$, including:

1) $\mathcal{A}$ has not made any private key generation queries either to pk_i or pk_j.
2) Adversary $\mathcal{A}$ has to obey the path rule w.r.t. $(\mathsf{Pa}_i, c_0^{i*})$, as specified below.
3) For any path $\mathsf{Pa}_j = (pk_{j_0} = pk_j, pk_{j_1}, \ldots, pk_{j_{\ell_j}})$ created by $\mathcal{A}$ for user j, adversary $\mathcal{A}$ has to obey the path rule w.r.t. $(\mathsf{Pa}_j, c_0^{j*})$, as specified below.

The path rule w.r.t. $(\mathsf{Pa}_i, c_0^{i*})$ is defined as follows (the path rule w.r.t. $(\mathsf{Pa}_j, c_0^{j*})$ is defined similarly):

For *any* sub-path $\mathsf{Pa}^i_{0\to k} = (pk_{i_0} = pk_i, pk_{i_1}, \ldots, pk_{i_k})$ of Pa_i, where $1 \le k \le \ell_i$, if for any $v, 0 \le v \le k-1$, $\mathcal{A}$ queried $\mathcal{O}_{rk}(i, pk_{i_v}, pk_{i_{v+1}})$, or $\mathcal{O}_{reen}(i, pk_{i_v}, pk_{i_{v+1}}, c^i_v)$ such that $\mathsf{Dec}(\mathsf{par}, c^i_{v+1}, sk_{i_{v+1}}) \in \{m_0, m_1\}$, then the following restriction must be met (actually, as the path rule is specified w.r.t. *any* sub-path, it implies that the following conditions hold also for any step in the sub-path $\mathsf{Pa}^i_{0\to k}$):

- $\mathcal{A}$ should not make the $\mathcal{O}_{sk}(pk_{i_{v+1}})$ query throughout the game.

With respect to the above experiment $Exp^{\mathsf{CPA}}_{\mathcal{E},\mathcal{A}}(1^\kappa)$, we define the advantage of an adversary $\mathcal{A}$ in $Exp^{\mathsf{CPA}}_{\mathcal{E},\mathcal{A}}$ as

$$\begin{aligned}&\mathrm{Adv}^{\mathsf{IND\text{-}APPRE\text{-}CPA}}_{\mathcal{E},\mathcal{A}}(\kappa)\\ &= |\Pr[\mathbf{Exp}^{\mathsf{CPA}}_{\mathcal{E},\mathcal{A}}(\kappa) = 1 | d = 0]\\ &\quad - \Pr[\mathbf{Exp}^{\mathsf{CPA}}_{\mathcal{E},\mathcal{A}}(\kappa) = 1 | d = 1]|,\end{aligned}$$

or equivalently [25],

$$\mathrm{Adv}^{\mathsf{IND\text{-}APPRE\text{-}CPA}}_{\mathcal{E},\mathcal{A}}(\kappa) = |2\Pr[d' = d] - 1|.$$

Definition 4. *An autonomous path proxy re-encryption scheme $\mathcal{E}$ is said to be $(t, q_{pk}, q_{sk}, q_{cp}, q_{rk}, q_{reen}, \epsilon)$-*IND-APPRE-CPA *secure, if for any t-time adversary $\mathcal{A}$ who makes at most $q_{pk}, q_{sk}, q_{cp}, q_{rk}$, and q_{reen} queries to oracles $\mathcal{O}_{pk}, \mathcal{O}_{sk}, \mathcal{O}_{cp}, \mathcal{O}_{rk}$, and $\mathcal{O}_{reen}$, respectively, we have $\mathrm{Adv}^{\mathsf{IND\text{-}APPRE\text{-}CPA}}_{\mathcal{E},\mathcal{A}}(\kappa) \le \epsilon$.*

Remark 1. Note that with our IND-APPRE-CPA definition, we only require the path rule w.r.t. $(\mathsf{Pa}_i, c_0^{i*})$ and $(\mathsf{Pa}_j, c_0^{j*})$. In particular, the adversary $\mathcal{A}$ can get $c^i_\gamma = \mathsf{ReEnc}_{0\to\gamma}(c_0^{i*})$, where $\mathcal{A}$ has not made an $\mathcal{O}_{sk}$ query on the user i_γ. Meanwhile, $\mathcal{A}$ can request to create a path for user i_γ as a delegator, and then re-encrypts c^i_γ along the path of i_γ. Then, our IND-APPRE-CPA security implies that the re-encrypted ciphertext of c^i_γ is helpless for $\mathcal{A}$ to distinguish the challenge message m_d w.r.t the path rule $(\mathsf{Pa}_i, c_0^{i*})$ and $(\mathsf{Pa}_\gamma, c^i_\gamma)$; That is, for some trivial cases, if $\mathcal{A}$ obtained the private key of user γ_k from private key oracle, he cannot make re-encryption key or re-encryption queries from user i_γ to γ_k on the path Pa_γ (so does a user on the path Pa_i). This captures that an original ciphertext under pk_i cannot be re-encrypted branching off the designated path Pa_i with meaningful decryption, and also captures that any re-encrypted ciphertexts on path Pa_i cannot be inserted into Pa_γ with meaningful decryption. Similarly, an original ciphertext under pk_j cannot be re-encrypted along the designated path Pa_i for user $i \neq j$ with meaningful decryption.

Remark 2. The above IND-APPRE-CPA definition assumes that each user can designate a single autonomous path, which is the most often case in practice. But, the definition can be simply extended to the general case of multiple autonomous paths, where the path rule should hold w.r.t. c_0^{i*} (respectively, c_0^{j*}) and all autonomous paths of user i (respectively, j).

4 Construction

Generally, an autonomous path proxy re-encryption scheme consists of the following seven algorithms.

- $\mathsf{Setup}(1^\kappa) \to \mathsf{par}$: On input the security parameter 1^κ, it outputs the system's public parameters par. The system's public parameters include: Two multiplicative groups $\mathbb{G}$ and $\mathbb{G}_1$ of the same prime order q, a generator of $\mathbb{G}$, i.e., g, a random element $g_1 \in \mathbb{G}$, and a cryptographic hash function $\mathsf{H} : \mathbb{G}_1 \to \mathbb{G}$ that is modelled to be an RO. A bilinear or pairing map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$. The message space $\mathcal{M}$ is $\mathbb{G}_1$. Finally, the system's public parameters $\mathsf{par} = (\mathbb{G}, \mathbb{G}_1, g, g_1, q, e, \mathsf{H})$ is output.
- $\mathsf{KeyGen}(\mathsf{par}, i) \to (pk_i, sk_i)$: To generate a decryption key, the user$_i$ selects $sk_i = x_i$ from $\mathbb{Z}_q^*$ randomly, and sets $pk_i = g^{x_i}$.
- $\mathsf{CreatPath}(\mathsf{par}, pk_i) \to (\mathsf{Pa}_i, \ell_i)$: On input the system's public parameters par, a public key pk_i of a delegator user i, it outputs an autonomous delegation path designated by user i of length ℓ_i. The autonomous delegation path is represented by (Pa_i, ℓ_i), where $\mathsf{Pa}_i = (pk_{i_0} = pk_i, pk_{i_1}, \ldots, pk_{i_{\ell_i}})$ is a sequence of ordered ℓ_i different public keys in the system.
- $\mathsf{RKGen}(\mathsf{par}, sk_i, \mathsf{Pa}_i) \to rk^i = (rk^i_{0\to 1}, rk^i_{1\to 2}, \ldots, rk^i_{\ell_i - 1\to \ell_i})$, where $\mathsf{Pa}_i = (pk_{i_0} = pk_i, pk_{i_1}, \ldots, pk_{i_{\ell_i}})$. To generate these re-encryption keys, the delegator with public key pk_i selects $X_j \leftarrow_R \mathbb{G}_1, r_j \leftarrow_R \mathbb{Z}_q^*$, for

$j = 1, \ldots, \ell_i$. For $j = 1$, the delegator computes $rk^i_{0\to 1} = (rk^i_{(0\to 1)_1}, rk^i_{(0\to 1)_2}, rk^i_{(0\to 1)_3}) = (g^{r_1}, X_1 \cdot e(g_1, pk_{i_1}^{r_1}), \mathsf{H}(X_1) \cdot g_1^{-sk_i})$. For $j = 2, \ldots, \ell_i$, the delegator computes $rk^i_{j-1\to j} = (rk^i_{(j-1\to j)_1}, rk^i_{(j-1\to j)_2}, rk^i_{(j-1\to j)_3}) = (g^{r_j}, X_j \cdot e(g_1, pk_{i_j}^{r_j}), \frac{\mathsf{H}(X_j)}{\mathsf{H}(X_{j-1})})$. Finally, the re-encryption key $rk^i = (rk_{0\to 1}, rk_{1\to 2}, \ldots, rk_{\ell_i - 1\to \ell_i})$ is output, and each $rk_{j-1\to j}$, $1 \le j \le \ell_i$ is sent to the corresponding proxies via a secure channel.

- $\mathsf{Enc}(\mathsf{par}, pk_i, m) \to c_0^i$: To encrypt a message $m \in \mathcal{M}$ under pk_i, where pk_i is a delegator that is denoted by pk_{i_0}, the algorithm selects $r \leftarrow_R \mathbb{Z}_q^*$ and computes

$$c_1 = g^r,\ c_2 = m \cdot e(g_1, pk_{i_0})^r.$$

Finally, the ciphertext is output as $c_0^i = (c_1, c_2)$.

- $\mathsf{ReEnc}(\mathsf{par}, \mathsf{Pa}_i, pk_{i_j}, pk_{i_{j+1}}, rk^i_{j\to j+1}, c_j^i) \to c_{j+1}^i$, where $0 \le j < \ell_i$, and $\mathsf{Pa}_i = (pk_{i_0} = pk_i, pk_{i_1}, \ldots, pk_{i_{\ell_i}})$: To re-encrypt a ciphertext under the public key pk_{i_j} to the one under $pk_{i_{j+1}}$ in a delegation path Pa_i which is designated by a delegator whose public key is pk_i (denoted by pk_{i_0} in our solution), this algorithm first checks whether $(pk_{i_j}, pk_{i_{j+1}}) \in \mathsf{Pa}_i$, and outputs "$\perp$" if not. Otherwise, if $j = 0$, pk_i's proxy parses the ciphertext c_j^i as $(c_{j_1}^i, c_{j_2}^i)$, and the re-encryption key $rk^i_{j\to j+1}$ as $(rk^i_{(j\to j+1)_1}, rk^i_{(j\to j+1)_2}, rk^i_{(j\to j+1)_3})$. If $j \ge 1$, pk_i's proxy parses the ciphertext c_j^i as $(c_{j_1}^i, c_{j_2}^i, c_{j_3}^i = rk^i_{(j-1\to j)_1}, c_{j_4}^i = rk^i_{(j-1\to j)_2})$, and the re-encryption key $rk^i_{j\to j+1}$ as $(rk^i_{(j\to j+1)_1}, rk^i_{(j\to j+1)_2}, rk^i_{(j\to j+1)_3})$. The user pk_i's proxy computes

$$c_{j+1}^i = (c^i_{j+1_1}, c^i_{j+1_2}, c^i_{j+1_3}, c^i_{j+1_4}),$$

where $c^i_{j+1_1} = c^i_{j_1}$, $c^i_{j+1_2} = c^i_{j_2} \cdot e(c^i_{j_1}, rk^i_{(j\to j+1)_3})$, $c^i_{j+1_3} = rk^i_{(j\to j+1)_1}$, $c^i_{j+1_4} = rk^i_{(j\to j+1)_2}$.

- $\mathsf{Dec}(\mathsf{par}, c_j^i, sk_{i_j}) \to (m, \perp)$: On input of the system's public parameters par, a ciphertext c_j^i, and the corresponding private key sk_{i_j} of user i_j, the algorithm first checks whether c_j^i is an original ciphertext or a re-encrypted ciphertext. For an original ciphertext, the user with public key pk_{i_j} parses c_j^i as $(c_{j_1}^i, c_{j_2}^i)$, computes and outputs $m = \frac{c^i_{j_2}}{e(c^i_{j_1}, g_1^{sk_{i_j}})}$. Otherwise, if the ciphertext is a re-encrypted one, the corresponding delegatee parses c_j^i as $(c_{j_1}^i, c_{j_2}^i, rk^i_{(j-1\to j)_1}, rk^i_{(j-1\to j)_2})$. The delegatee with public key $pk_{i_j}, 1 \le j \le \ell_i$, computes $X_j = \frac{rk^i_{(j-1\to j)_2}}{e(g_1^{sk_{i_j}}, rk^i_{(j-1\to j)_1})}$. Finally, the delegatee with public key pk_{i_j} computes and outputs $m = \frac{c^i_{j_2}}{e(c^i_{j_1}, \mathsf{H}(X_j))}$.

Correctness

(1) For an original ciphertext $c_0^i = (c^i_{0_1}, c^i_{0_2})$, the decryption $m = \frac{c^i_{0_2}}{e(c^i_{0_1}, g_1^{sk_{i_0}})} = \frac{m \cdot e(g_1, pk_{i_0})^r}{e(g^r, g_1^{sk_{i_0}})} = \frac{m \cdot e(g_1^r, g^{sk_{i_0}})}{e(g_1^r, g^{sk_{i_0}})} = m$. The decryption is obviously correct.

(2) For $j = 1$, the ciphertext is like $(c^i_{1_1}, c^i_{1_2}, rk^i_{(0\to 1)_1}, rk^i_{(0\to 1)_2})$, where

$$\begin{aligned} c^i_{1_1} &= g^r, \\ c^i_{1_2} &= m \cdot e(g_1, pk_{i_0})^r \cdot e(c^i_{1_1}, rk^i_{(0\to 1)_3}) \\ &= m \cdot e(g^r, g_1^{sk_{i_0}}) \cdot e(g^r, \mathsf{H}(X_1) \cdot g_1^{-sk_{i_0}}) \\ &= m \cdot e(g^r, \mathsf{H}(X_1)), \\ rk^i_{(0\to 1)_1} &= g^{r_1}, \\ rk^i_{(0\to 1)_2} &= X_1 \cdot e(g_1, pk_{i_1})^{r_1}. \end{aligned}$$

The delegatee, whose public key is pk_{i_1}, computes

$$\begin{aligned} X_1 &= \frac{rk^i_{(0\to 1)_2}}{e(g_1^{sk_{i_1}}, rk^i_{(0\to 1)_1})} \\ &= \frac{X_1 \cdot e(g_1, pk_{i_1})^{r_1}}{e(g_1^{sk_{i_1}}, g^{r_1})} \\ &= \frac{X_1 \cdot e(g_1^{r_1}, g^{sk_{i_1}})}{e(g_1^{r_1}, g^{sk_{i_1}})} \\ &= X_1. \end{aligned}$$

Then, the delegatee computes

$$m = \frac{c^i_{1_2}}{e(c^i_{1_1}, \mathsf{H}(X_1))} = \frac{m \cdot e(g^r, \mathsf{H}(X_1))}{e(g^r, \mathsf{H}(X_1))}.$$

(3) For $j > 1$, the ciphertext is like $(c^i_{j_1}, c^i_{j_2}, rk^i_{(j-1\to j)_1}, rk^i_{(j-1\to j)_2})$, where

$$\begin{aligned} c^i_{j_1} &= g^r, \\ c^i_{j_2} &= m \cdot e(c^i_{j_1}, \mathsf{H}(X_{j-1})) \cdot e(c^i_{j_1}, rk^i_{(j-1\to j)_3}) \\ &= m \cdot e(g^r, \mathsf{H}(X_{j-1})) \cdot e(g^r, \frac{\mathsf{H}(X_j)}{\mathsf{H}(X_{j-1})}) \\ &= m \cdot e(g^r, \mathsf{H}(X_j)), \\ rk^i_{(j-1\to j)_1} &= g^{r_j}, \\ rk^i_{(j-1\to j)_2} &= X_j \cdot e(g_1, pk_{i_j})^{r_j}. \end{aligned}$$

The delegatee whose public key is pk_{i_j}, computes

$$\begin{aligned} X_j &= \frac{rk^i_{(j-1\to j)_2}}{e(g_1^{sk_{i_j}}, rk^i_{(j-1\to j)_1})} \\ &= \frac{X_j \cdot e(g_1, pk_{i_j})^{r_j}}{e(g_1^{sk_{i_j}}, g^{r_j})} \\ &= \frac{X_j \cdot e(g_1^{r_j}, g^{sk_{i_j}})}{e(g_1^{r_j}, g^{sk_{i_j}})} \\ &= X_j. \end{aligned}$$

Then, the delegatee with public key pk_i computes

$$m = \frac{c^i_{j_2}}{e(c^i_{j_1}, \mathsf{H}(X_j))} = \frac{m \cdot e(g^r, \mathsf{H}(X_j))}{e(g^r, \mathsf{H}(X_j))}.$$

The above 1 to 3 show that our AP-PRE scheme is correct.

5 Security Proof

Theorem 1. *Our AP-PRE scheme in Section 4 is* IND-APPRE-CPA *secure, under the* DBDH *assumption in the random oracle model.*

Proof. Let $\mathcal{A}$ be a PPT adversary that has non-negligible advantage ε in attacking our AP-PRE scheme in Section 4 in the sense of IND-APPRE-CPA. We construct another PPT adversary $\mathcal{B}$ using $\mathcal{A}$ to break the DBDH assumption also with non-negligible probability.

Adversary $\mathcal{B}$ accepts a properly-distributed tuple $\langle \mathbb{G} = \langle g \rangle, g^a, g^b, g^c, T \rangle \in \mathbb{G}^4 \times \mathbb{G}_1$ as input. We say the tuple $(g^a, g^b, g^c, e(g,g)^{abc})$ is a DBDH instance. If $T = e(g,g)^{abc}$, $\mathcal{B}$ tries to output 1; Otherwise, $\mathcal{B}$ tries to output 0. The interaction between the two adversaries $\mathcal{A}$ and $\mathcal{B}$ is described as following:

- SETUP. $\mathcal{B}$ generates the scheme's system parameters $\mathsf{par} = (\mathbb{G}, \mathbb{G}_1, g, g_1 = g^b, q, e, \mathsf{H})$ and gives par to $\mathcal{A}$, where q is the prime order of $\mathbb{G}$ and $\mathbb{G}_1$, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a pairing map, and $\mathsf{H} : \mathbb{G}_1 \to \mathbb{G}$ is a cryptographic hash function that is modelled to be an RO.
  $\mathcal{B}$ takes $i^*, j^* \leftarrow \{1, 2, \ldots, n\}$ independently and uniformly at random. That is, $\mathcal{B}$ tries to randomly guess the two users to be challenged by the adversary.
  During the game, $\mathcal{B}$ needs to maintain four tables T_{pk}, $\mathcal{RK}$, $\mathcal{P}$ and $\mathcal{H}$. They are initially empty and are used to record the public keys, re-encryption keys, autonomous paths, and the RO queries, respectively.
- The find stage. In this stage, $\mathcal{B}$ responds $\mathcal{A}$'s queries as following:
  - On an $\mathcal{O}_{pk}(i)$ query: $\mathcal{B}$ first selects a random $x_i \leftarrow \mathbb{Z}_q^*$. If $i = i^*$ or $i = j^*$, $\mathcal{B}$ sets $pk_i = (g^a)^{x_i}$, which means the private keys of the user is ax_i that is unknown to $\mathcal{B}$. Otherwise (i.e., $i \notin \{i^*, j^*\}$), $\mathcal{B}$ sets $pk_i = g^{x_i}$, which means that the private key of this user is x_i. Finally, $\mathcal{B}$ returns pk_i to $\mathcal{A}$, and records (pk_i, x_i) for $i \notin \{i^*, j^*\}$ in the table T_{pk}.
  - On an $\mathcal{O}_{sk}(pk_i)$ query, if $i \notin \{i^*, j^*\}$, $\mathcal{B}$ returns $sk_i = x_i$ to $\mathcal{A}$. Otherwise (i.e., $i \in \{i^*, j^*\}$), $\mathcal{B}$ aborts.
  - On an $\mathcal{O}_{cp}(i, \mathsf{Pa}_i)$ query, $\mathcal{B}$ searches the table $\mathcal{RK}$ to see if there is any entry for user i in the table. If there is, $\mathcal{B}$ returns $\perp$ which indicates that $\mathcal{A}$ has queried to create a path for user i before. Otherwise, it means that it is the first time for $\mathcal{A}$ to make an $\mathcal{O}_{cp}$ query w.r.t. user i. If $i \notin \{i^*, j^*\}$, $\mathcal{B}$ acts just as the honest user does.
    For $i \in \{i^*, j^*\}$, $\mathcal{B}$ computes the re-encryption key rk^i for path $\mathsf{Pa}_i = (pk_{i_0} = pk_i, pk_{i_1}, \ldots, pk_{i_{l_i}})$ designated by the delegator user i, as follows:
    - $\mathcal{B}$ selects $r_1, \ldots, r_{i_{\ell_i}} \leftarrow_R \mathbb{Z}_q^*$, $X_1, \ldots, X_{i_{\ell_i}} \leftarrow_R \mathbb{G}_1$.
    - To compute $rk^i_{0\to 1}$, $\mathcal{B}$ randomly selects $Z_1 \leftarrow \mathbb{G}_1$, and sets $rk^i_{0\to 1} = (rk^i_{(0\to 1)_1}, rk^i_{(0\to 1)_2}, rk^i_{(0\to 1)_3}) = (g^{r_1}, X_1 \cdot e(g_1, pk_{i_1}^{r_1}), Z_1)$. Note that, in the real view of $\mathcal{A}$, $rk^i_{0\to 1} = (rk^i_{(0\to 1)_1}, rk^i_{(0\to 1)_2}, rk^i_{(0\to 1)_3}) = (g^{r_1}, X_1 \cdot e(g_1, pk_{i_1}^{r_1}), \mathsf{H}(X_1) \cdot g_1^{-sk_i})$.
    - To compute re-encryption keys $rk^i_{j\to j+1}$ for $1 \le j \le \ell_i - 1$, $\mathcal{B}$ computes $rk^i_{j\to j+1} = (rk^i_{(j\to j+1)_1}, rk^i_{(j\to j+1)_2}, rk^i_{(j\to j+1)_3}) = (g^{r_{j+1}}, X_{j+1} \cdot e(g_1, pk_{i_{j+1}})^{r_{j+1}}, \frac{\mathsf{H}(X_{j+1})}{\mathsf{H}(X_j)})$.
    Finally, $\mathcal{B}$ records rk^i in the table $\mathcal{RK}$, and returns "Pa_i is created" to $\mathcal{A}$. Throughout the simulation, in case $\mathcal{A}$ makes the RO-query $H(X_k)$, $1 \le k \le l_i$, w.r.t. rk^i for $i \in \{i^*, j^*\}$, $\mathcal{B}$ aborts. Note that, conditioned on $\mathcal{A}$ does not make the RO-queries $H(X_k)$'s, the simulation of rk^i for $i \in \{i^*, j^*\}$ by $\mathcal{B}$ is perfect in the random oracle model.
  - On an $\mathcal{O}_{rk}(i, pk_{i_j}, pk_{i_{j+1}})$ query, $\mathcal{B}$ first checks if the path table $\mathcal{P}$ contains a path $\mathsf{Pa}_i = (\ldots, pk_{i_j}, pk_{i_{j+1}}, \ldots)$ for user i. If not, $\mathcal{B}$ outputs "$\perp$" indicating an invalid query. If yes, $\mathcal{B}$ retrieves $rk^i_{j\to j+1}$ from rk^i in the table $\mathcal{RK}$, and returns $rk^i_{j\to j+1}$ to $\mathcal{A}$.
  - On an $\mathcal{O}_{reen}(i, pk_{i_j}, pk_{i_{j+1}}, c^i_j)$ query, $\mathcal{B}$ first checks if the path table $\mathcal{P}$ contains a path $\mathsf{Pa}_i = (\ldots, pk_{i_j}, pk_{i_{j+1}}, \ldots)$ for user i. If not, $\mathcal{B}$ outputs "$\perp$" indicating an invalid query. Otherwise, $\mathcal{B}$ retrieves $rk^i_{j\to j+1}$ from rk^i in the table $\mathcal{RK}$. Then, $\mathcal{B}$ runs ReEnc(par, $\mathsf{Pa}_i, rk^i_{j\to j+1}, c^i_j$), and returns the output to $\mathcal{A}$.
- The choice − and − challenge stage. At the end of the find stage, $\mathcal{A}$ submits two messages m_0 and m_1 of equal length in the message space $\mathbb{G}_1$, a delegator user with public key pk_i and its designated autonomous path Pa_i, and a user with public key pk_j.
  First, if $i \notin \{i^*, j^*\}$ or $j \notin \{i^*, j^*\}$, i.e., the guess of (i^*, j^*) by $\mathcal{B}$ is incorrect, $\mathcal{B}$ aborts. Otherwise, to compute the challenge ciphertext, $\mathcal{B}$ randomly selects $d \leftarrow \{0, 1\}$. Finally, $\mathcal{B}$ selects r_i^*, r_j^* randomly from $\mathbb{Z}_q^*$, and computes the challenge ciphertext under pk_i in path Pa_i as $c_0^{i^*}$, and a ciphertext under pk_j as $c_0^{j^*}$

$$c_0^{i^*} = ((g^c)^{r_i^*}, m_d \cdot T^{x_i r_i^*}).$$

$$c_0^{j^*} = ((g^c)^{r_j^*}, m_d \cdot T^{x_j r_j^*}).$$

  At the end, $c^* = (c_0^{i^*}, c_0^{j^*})$ is sent to $\mathcal{A}$ as the challenge ciphertexts.

  Here, the restrictions imposed upon $\mathcal{A}$ are listed as following:
  - $\mathcal{A}$ has not made any private key generation queries either to pk_i or pk_j;
  - $\mathcal{A}$ has to obey the path rule w.r.t. $(\mathsf{Pa}_i, c_0^{i^*})$, where the path rule is defined in Section 2;
  - For any path $\mathsf{Pa}_j = (pk_{j_0} = pk_j, pk_{j_1}, \ldots, pk_{j_{\ell_i}})$ created by $\mathcal{B}$ for user j, $\mathcal{A}$ has to obey

the path rule w.r.t. $(\mathsf{Pa}_j, c_0^{j^*})$ as defined in Section 2.[3]

- The guess stage. $\mathcal{A}$ continues to make queries as in the find stage, with the same restrictions as in the choice − and − challenge stage.
  At the end of this stage, $\mathcal{A}$ outputs his guess d', where $d' \in \{0,1\}$. If $d = d'$, $\mathcal{B}$ outputs 1, in which case it indicates that $T = e(g,g)^{abc}$, and (g^a, g^b, g^c, T) is a DBDH tuple; Otherwise, $\mathcal{B}$ outputs 0 indicating that T is a random element in $\mathbb{G}_1$.

It is obvious that if the input to $\mathcal{B}$ is a DBDH tuple, then the challenge ciphertexts $c_0^{i^*}$ and $c_0^{j^*}$ are correct original encryptions of m_d under pk_i and pk_j. The ciphertext $c_0^{i^*}$ is an original ciphertext under pk_i in its autonomous delegation path Pa_i, and $c_0^{j^*}$ is an original ciphertext under pk_j in some autonomous path designated by user j with public key pk_j. Otherwise, $c_0^{i^*}$ and $c_0^{j^*}$ are the encryptions of a random element.

Next, we analyze the probability that $\mathcal{B}$ aborts in its simulation.

First, $\mathcal{B}$ may abort, in case that $\mathcal{A}$ makes an RO-query $H(X_k)$, where $1 \le k \le l_i$ and X_k was used by $\mathcal{B}$ in generating rk^i for $i \in \{i^*, j^*\}$. Note that, in case $\mathcal{B}$ does not abort in this case, its simulation of rk^i for $i \in \{i^*, j^*\}$ is perfect in the RO model. We further consider two cases.

- The first case is that $\mathcal{A}$ has queried $\mathcal{O}_{sk}(pk_{i_k})$, i.e., $\mathcal{A}$ has gotten the private key of user pk_{i_k}. In this case, according to the path rule, $\mathcal{A}$ is not allowed to get or use the re-encryption key $rk^i_{k-1\to k}$, and hence the view of $\mathcal{A}$ in this case is independent of X_k. This means that $\mathcal{A}$ can make the RO-query $H(X_k)$ in this case is negligible in the random oracle model.
- The second case is that $\mathcal{A}$ did not make the query $\mathcal{O}_{sk}(pk_{i_k})$, but got the re-encryption key $rk^i_{k-1\to k} = (g^{r_k}, X_k \cdot e(g_1, pk_{i_k}^{r_k}), \cdot)$. In this case, the probability that $\mathcal{A}$ makes the RO-query $H(X_k)$ must also be negligible, as the ability of outputting X_k implies the ability of break the DBDH assumption. Specifically, if $\mathcal{A}$ can make the RO-query $H(X_k)$ with non-negligible probability in this case, we can construct another PPT algorithm $\mathcal{C}$ to break the DBDH assumption also with non-negligible probability. In more detail, the input to $\mathcal{C}$ is $(g^{r_k}, g_1 = g^b, pk_{i_k} = g^{sk_{i_k}}, Z_k)$, where $Z_k = e(g,g)^{r_k \cdot b \cdot sk_{i_k}}$ or $Z_k \leftarrow_R \mathbb{G}_1$. $\mathcal{C}$ sets $rk^i_{k-1\to k} = (g^{r_k}, X_k \cdot Z_k, \cdot)$, and then uses the ability of $\mathcal{A}$ in outputting X_k to violate the DBDH assumption. Further details are omitted here.

Second, $\mathcal{B}$ may abort due to incorrect guess of (i^*, j^*). Conditioned on $\mathcal{A}$ did not make the RO-query $H(X_k)$ (i.e., the simulation of rk^i for $i \in \{i^*, j^*\}$ by $\mathcal{B}$ is perfect), this event occurs with probability $1 - \frac{1}{n^2}$.

It is obvious that in case $\mathcal{B}$ does not abort, which occurs with probability negligibly close to $\frac{1}{n^2}$, the view of $\mathcal{A}$ in the simulation is identical to its view in the real attack. This shows that $\mathcal{B}$'s advantage in solving the DBDH problem is negligibly close to $\frac{\varepsilon}{n^2}$, where ε is the non-negligible probability with which $\mathcal{A}$ can break our AP-PRE scheme. □

3. The path rule makes sure that $\mathcal{A}$ cannot transform the challenge ciphertexts to the ciphertexts under any entities he grasps the corresponding private keys.

6 Conclusion

In this paper, we introduce a new cryptographic primitive, called autonomous path proxy re-encryption, which is motivated by the demands in several potential applications. Not only do we first put forward the concept of delegator autonomous path proxy re-encryption, but also we give a concrete construction of an IND-CPA secure scheme under this concept. We note that such scheme combines the advantage of a single-hop PRE and a multi-hop PRE, in other words, AP-PRE provides much better fine-grained access control to the delegation path than the traditional multi-hop PRE. In the new proposed AP-PRE scheme, the delegator has the ability to fully control the selection of the delegatees as in a single-hop PRE, as well as the convenience and flexibility of a multi-hop PRE. In fact, an AP-PRE must be a multi-hop proxy re-encryption. Moreover, in our scheme, any delegatee can designate a new delegation path, while any re-encrypted ciphertext from other path cannot be branched into his new path. As discussed, AP-PRE schemes are desirable in many interesting applications.

Our major contributions are summarized as follows:

- We introduce a new cryptographic primitive, called autonomous path proxy re-encryption. This new primitive has many potential applications.
- We give the formal definition of autonomous path proxy re-encryption, and the formal security model of this cryptographic primitive.
- We construct a concrete scheme, with provable IND-CPA security under the DBDH assumption in the random oracle model. Achieving IND-CCA secure AP-PRE schemes is left as an interesting question for future exploration.

Acknowledgments

We are indebted to the anonymous referees for their great review efforts and for numerous invaluable and insightful suggestions, which have significantly improved this work. The work of the first author was supported in part by the National Natural Science Foundation of China (Grant Nos. 61371083, 61632012), and in part by Shanghai High Technology Field Project under Grant 16511101400. The work of the second and the third authors was supported by National Natural Science Foundation of China (Grant Nos. 61472084, 61272012, U1536205), Shanghai Innovation Action Project No. 16DZ1100200, and Project funded by China Postdoctoral Science Foundation.

References

[1] M. Mambo and E. Olamoto, "Proxy cryptosystem: Delegation of the power to decrypt ciphertexts," *IEICE Trans.*, vol. 80, no. 1, pp. 54–63, 1997.

[2] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. Int. Conf. Theory Appl. Cryptographic Tech.*, 1998, pp. 127–144.

[3] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," in *ACM Trans. Inf. Syst. Security*, vol. 9, pp. 1–30, 2005.
[4] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Inf. Syst. Security*, vol. 9, no. 1, pp. 1–30, 2006.
[5] J. Weng, R. H. Deng, X. Ding, C. Chu, and J. Lai, "Conditional proxy re-encryption secure against chosen-ciphertext attack," in *Proc. ACM Symp. Inf. Comput. Commun. Security*, 2009, pp. 322–332. [Online]. Available: http://doi.acm.org/10.1145/1533057.1533100
[6] C. K. Chu, J. Weng, S. S. M. Chow, J. Zhou, and R. H. Deng, "Conditional proxy broadcast re-encryption," in *Proc. Australasian Conf. Inf. Security Privacy*, 2009, pp. 327–342.
[7] L. M. Fang, J. D. Wang, C. P. Ge, and Y. J. Ren, "Fuzzy conditional proxy re-encryption," *Sci. China Inf. Sci.*, vol. 56, no. 5, pp. 1–13, 2013.
[8] Q. Tang, "Type-based proxy re-encryption and its construction," in *Proc. 9th Int. Conf. Cryptology India Progress Cryptology-INDOCRYPT*, 2008, pp. 130–144.
[9] A.-A. Ivan and Y. Dodis, "Proxy cryptography revisited," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2003.
[10] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proc. ACM Conf. Comput. Commun. Security*, 2007, pp. 185–194.
[11] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Proc. 5th Int. Conf. Appl. Cryptography Netw. Security*, 2007, pp. 288–306.
[12] S. Hohenberger, G. N. Rothblum, A. Shelat, and V. Vaikuntanathan, "Securely obfuscating re-encryption," in *Proc. 4th Conf. Theory Cryptography*, 2007, pp. 233–252.
[13] R. H. Deng, J. Weng, S. Liu, and K. Chen, "Chosen-ciphertext secure proxy re-encryption without pairings," in *Proc. Int. Conf. Cryptology Netw. Security*, 2008, pp. 1–17.
[14] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," in *Proc. Public Key Cryptography*, 2008, pp. 360–379.
[15] B. Libert and D. Vergnaud, "Tracing malicious proxies in proxy re-encryption," in *Pairing*, 2008, pp. 332–353.
[16] G. Ateniese, K. Benson, and S. Hohenberger, "Key-private proxy re-encryption," in *Proc. Cryptographers' Track RSA Conf. Topics Cryptology*, 2009, pp. 279–294.
[17] J. Shao and Z. Cao, "CCA-secure proxy re-encryption without pairings," in *Proc. Public Key Cryptography*, 2009, pp. 357–376.
[18] J. Weng, Y. Zhao, and G. Hanaoka, *On the Security of a Bidirectional Proxy Re-encryption Scheme from PKC 2010*. Berlin, Germany: Springer Berlin Heidelberg, 2010.
[19] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1786–1802, Mar. 2011.
[20] T. Isshiki, M. H. Nguyen, and K. Tanaka, "Proxy re-encryption in a stronger security model extended from CT-RSA2012," in *Proc. Cryptographers' Track RSA Conf. Topics Cryptology*, 2013, pp. 277–292.
[21] K. Singh, C. P. Rangan, and A. K. Banerjee, "Cryptanalysis of unidirectional proxy re-encryption scheme," in *Proc. Inf. Commun. Technol.-EurAsia Conf.*, 2014, pp. 564–575.
[22] E. Kirshanova, "Proxy re-encryption from lattices," in *Proc. Public Key Cryptography*, 2014, pp. 77–94.
[23] A. Kiayias, M. Osmanoglu, and Q. Tang, *Graded Encryption, or How to Play Who Wants To Be A Millionaire? Distributively*. Berlin, Germany: Springer International Publishing, 2014.
[24] C. K. Chu, S. S. M. Chow, W. G. Tzeng, J. Zhou, and R. H. Deng, "Key-aggregate cryptosystem for scalable data sharing in cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 468–477, Feb. 2014.
[25] M. Bellare and A. Sahai, "Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization," in *Proc. Annu. Int. Cryptology Conf.*, 1999, pp. 519–536.

Zhenfu Cao (SM'10) received the BSc degree in computer science and technology and the PhD degree in mathematics from Harbin Institute of Technology, China. His research interests mainly include Number Theory, Cryptography and Information Security. He has published more than 400 academic papers in leading journals or conferences. He was early promoted to associate professor in 1987, and became a full professor in 1991. He is currently a distinguished professor in East China Normal University, China. He also serves as a member of the expert panel for the National Nature Science Fund of China. He is actively involved in academic community, serving as program co-chairs or committee members for many international conferences, including the *IEEE Global Communications Conference (GLOBECOM)*, and the *IEEE International Conference on Communications (ICC)*. He is the associate editor of *Computers and Security* (Elsevier) and *Security and Communication Networks* (John Wiley), an editorial board member of *Fundamenta Informaticae* (IoS) and *Peer-to-Peer Networking and Applications* (Springer-Verlag). Also, he serves as guest editor for several journals, including *IEEE Transactions on Parallel and Distributed Systems* and *Wireless Communications and Mobile Computing* (Wiley). He has received a number of awards, including the Youth Research Fund award of the Chinese Academy of Science (1986), the Ying-Tung Fok Young Teacher award (1989), the National Outstanding Youth Fund of China (2002), the Special Allowance by the State Council of China (2004), and a co-recipient of the 2007 IEEE ICC Best Paper award. He is also the principal investigators of Asia 3 Foresight Program and the key project of National Natural Science Foundation of China. He is a senior member of the IEEE.

Hongbing Wang received the PhD degree in computer science and technology from Shanghai Jiao Tong University, China, in 2013. Her current research interests include applied cryptography, network security, cloud computing, and RFID security, etc.

Yunlei Zhao received the PhD degree in computer science from Fudan University, Shanghai, China, in 2004. He joined Hewlett-Packard European Research Center, Bristol, U.K., as a Post-Doctoral Researcher, in 2004. Since 2005, he has been with Fudan University, and is currently a professor with the School of Computer Science, Fudan University. His research interests are the theory and applications of cryptography.