An Efficient and Provably Secure Certificateless Key Insulated Encryption with Applications to Mobile Internet

Libo He, Chen Yuan, Hu Xiong, and Zhiguang Qin

(Corresponding author: Libo He)

School of Information and Software Engineering & University of Electronic Science and Technology of China
Chengdu, Sichuan, 610054, China
(Email: libowqrs@gmail.com)

(Received Apr. 14, 2016; revised and accepted July 19 & Sept. 26, 2016)

Abstract

Certificateless encryption (CLE) alleviates the heavy certificate management in traditional public key encryption and the key escrow problem in the ID-based encryption simultaneously. Current CLE schemes assumed that the user's secret key is absolutely secure. Unfortunately, this assumption is too strong in case the CLE is deployed in the hostile setting and the leakage of the secret key is inevitable. In this paper, we present a new concept called a certificateless key insulated encryption scheme(CL-KIE). We argue that this is an important cryptographic primitive that can be used to achieve key-escrow free and key-exposure resilience. We also present an efficient CL-KIE scheme based on bilinear pairing. After that, the security of our scheme is proved under the Bilinear Diffie-Hellman assumption in the random oracle model. Further, the potential applications of CL-KIE is also briefly illustrated.

Keywords: Bilinear Pairing; Certificateless Cryptography; Key-insulated

1 Introduction

The public-key cryptography is called asymmetric key encryption, as every user owns a pair of keys: a public key and a private key. In 1978, Ron Rivest, Adi Shamir, and Leonard Adleman [15] published the first practical public key encryption RSA algorithm. The security of this algorithm is relied on practical difficulty of factoring the product of two large prime number. Another widely used public key cryptography Elgama algorithm based on the Diffie-Hellman key exchange was described by Taher Elgamal [10] in 1984.

The public key cryptosystem needs Public Key Infrastructure (PKI) to offer the authentication and validation for the public key [19]. But PKI will encounter a lot of challenges on efficiency and scalability for its complicated structure. In 1984, the Identity-based Encryption was firstly proposed by Shamir [24]. In 2001, Dan Boneh and Matthew K. Franklin [7] proposed a practical identity-based encryption system based on Weil pairing over elliptic curves and finite fields. In IBE, the public key could be any arbitrary characters related to user's identity [13, 22].

The private key is derived from the identity of an entity and the master key only known by Key Generation Center (KGC). So the certificate which is used to authenticate public key will not be necessary. However, the key escrow problem arises that the malicious authority can impersonate any users to get the corresponding private key.

To solve the problem of key escrow in Identity-based Encryption and guarantee the authenticity of public keys without the use of the certificate in public key encryption, the certificateless public key encryption (CL-PKE) has been introduced by Al-Riyami and Paterson [1] in 2003. In CL-PKE, the private key is separated into two parts: one partial private key is still generated in KGC, and the secret key is selected by the user itself. The malicious KGC only can get the partial private key, so it can not impersonate any user to attack the system. Hence, the CL-PKE solves the problems of key escrow. Since then, CL-PKE becomes a research hotspot and several other relevant certificateless encryption schemes [3, 4, 5, 8, 14, 20, 23] have been developed. Until then, current CL-PKE schemes assumed that the user's secret key is absolutely secure. However, a higher security requirement in CL-PKE is needed, for example, the case where an adversary steals the whole private key.

The exposure of private key is a devastating disaster for the cryptosystem. Key-evolving cryptosystem can alleviate the damage of key leakage. Normally, Key-evolving cryptosystem can be categorized into three groups as follows: forward-security [2, 26], key-insulation [9, 11, 16, 18, 21, 25] and intrusion-resilience [12]. In the key-evolving cryptosystem, the lifetime of the system is

Table 1: Functionality comparison

	Public key management-Free	Key-escrow Free	Key-insulation
PKE	×	×	×
IBE	√	×	×
CL-PKE	√	√	×
our scheme	√	√	√

divided into N time periods. For forward-secure scheme, the private key is updated by the user himself during every time period without any interaction with other devices. Even when an adversary compromise the private key at the current time period, the forward-secure scheme can also guarantee the security of the prior time periods. In the key-insulated scheme, a user's private key is updated by communicating with a physically-secure device for every time period. The private key is composed of two parts: one part is generated by the master key and the other is created by the helper key from the physically-secure device. Meanwhile, the public key remains fixed during the whole periods of key updating. By this approach, even an adversary who steals the private key in the present time period cannot get the private key in the former or later period. The private key in the intrusion-resilient scheme also will be updated by interaction with a physically-secure device.

The difference between the key-insulated scheme and the intrusion-resilient scheme is that the intrusion-resilient scheme refreshes the secret keys of the user and physically-secure device many times in one period. So intrusion-resilient scheme remains secure even after many arbitrary compromises of both user and physically-secure device, as long as the compromise are not simultaneous. Among the above three types of key-evolving schemes, the key-insulated scheme and the intrusion-resilient scheme can offer higher security than the forward-secure scheme. Also, the intrusion-resilient scheme gives more security and is less efficient compared with the key-insulated scheme. Therefore, the key-insulated scheme is the trade-off between security and efficiency as shown in Table 1.

1.1 Contribution

In this paper, we resolve the above problem and make the following novel contributions as follows:

- Firstly, we present a concrete paradigm called the certificateless key-insulated encryption scheme (CL-KIE). We integrate key-insulated technique into CL-PKE. Through this method, our new scheme not only can solve the problem of key escrow, but also can achieve the functionality of key-insulation, without heavy public key management.

- We give the formal definition and security model for CL-KIE scheme and the construction of a CL-KIE scheme based on bilinear pairing. We also give the security proof for the CL-KIE scheme under the Bilinear Diffie-Hellman assumption in the random oracle model.

- Finally, our scheme with key updates can give more security fuctionality to solve the problem of key exposure compared with some other CL-PKE schemes, while sacrificing a little on the cost of execution time. This is an attractive advantage which the standard CL-PKE scheme does not possess.

1.2 Organization

The rest of this paper is organized as follows: We formalize the definition and give the security model of CL-KIE schemes in Section 2. Section 3 first gives an introduction to bilinear pairings and the Bilinear Diffie-Hellman Problem, then proposes a construction of the CL-KIE scheme. We prove that our scheme under the Bilinear Diffie-Hellman assumption in the random oracle model and compare our scheme with CL-PKE scheme on efficiency and security capability in Section 4. Further, the potential applications for CL-KIE is discussed in Section 5. At last, we conclude this paper in Section 6.

2 Formal Definition and Security Model

In this section we first formalize the definition of the CL-KIE scheme by cooperating the key-insualted scheme and the CL-PKE scheme. After that, we propose the security model of the CL-KIE scheme.

2.1 Definition of CL-KIE

We denote the CL-KIE scheme, which consists of the following algorithms:

Setup: The algorithm is given a security parameter k, and generates the system parameters $params$, `master-key` and `master-helper-key`. The system parameters include a description of a finite message space $\mathcal{M}$, a description of a finite ciphertext space $\mathcal{C}$ and a randomness space $\mathcal{R}$.

SecretValExtract: The algorithm takes as input $params$ and a identity string $ID_A \in \{0,1\}^*$, and

generates a random $x_A \in Z_q$ as the secret value associated with the entity A.

PartialKeyExtract: The algorithm takes as input *params*, `master-key`, and a identity string ID_A, and return the partial private key D_A corresponding to the entity A.

HelperKeyUpdate: The algorithm takes as input *params*, a time period i, `master-helper-key`, a identity string ID_A, and return the helper key $HK_{A,i}$ for a time period i.

PrivateKeyUpdate: The algorithm takes as input *param*, a time period i, a helper key $HK_{A,i}$, an identity string ID_A, a partial private key D_A and a secret value x_A, and output the private key $S_{A,i}$ for a time period i.

PublicKeyExtract: The algorithm takes as input *params*, a secret value x_A and a identity string ID_A, and output the public key P_A of the entity A.

Encrypt: The algorithm takes as input a time period i, *params*, an identity string ID_A, a public key P_A and a plaintext $M \in \mathcal{M}$. It returns the ciphertext $C \in \mathcal{C}$.

Decrypt: The algorithm takes as input a time period i, *params*, a private key $S_{A,i}$ and a ciphertext C. It returns the corresponding plaintext $M \in \mathcal{M}$.

2.2 Security Model

In this subsection we define the security model for the CL-KIE scheme by Indistinguishability of Encyption Against Adaptive Chosen Ciphertext Attacker (IND-CPA) game which is conducted between a challenger $\mathcal{S}$ and an adversary $\mathcal{A}$. In our scheme, we define two kinds of adversaries $TypeI$ adversary ($\mathcal{A}_1$) and $TypeII$ adversary ($\mathcal{A}_2$): $\mathcal{A}_1$ represents the external attacker who can not access the *master-key* but can replace the public key for an entity with its choice; $\mathcal{A}_2$ represents the malicious KGC who can access the *master-key*. We prohibit $\mathcal{A}_2$ from replacing the public key since $\mathcal{A}_2$ can not seclect the secret value by itself. First we give a list of oracles that a general adversary in our scheme may carry out, then we define chosen ciphertext security of CL-KIE for two kinds of adversaries respectively.

The list of oracles that a general adversary in CL-KIE may carry out is the following:

- **Partial-Private-Key-Queries:** If necessary, $\mathcal{A}$ makes **Partial-Private-Key-Queries** on the identity ID_A, and $\mathcal{S}$ returns the partial private key D_A associated with ID_A to $\mathcal{A}$.
- **Helper-Key-Queries:** $\mathcal{A}$ makes **Helper-Key-Queries** on the identity ID_A at a time period i, and $\mathcal{S}$ returns the helper key $HK_{A,i}$ to $\mathcal{A}$.
- **Secret-Value-Queries:** If necessary, $\mathcal{A}$ makes **Secret-Value-Queries** on the identity ID_A, and $\mathcal{S}$ returns the secret value x_A associated with ID_A to $\mathcal{A}$.
- **Public-Key-Queries:** $\mathcal{A}$ makes **Public-Key-Queries** on the identity ID_A, and $\mathcal{S}$ returns the helper key P_A to $\mathcal{A}$.
- **Public-Key-Replace:** If necessary, $\mathcal{A}$ can repeatedly make **Public-Key-Replace** to set the public key P_A for any value of its choice.
- **Decryption-Queries:** $\mathcal{A}$ makes **Decryption-Queries** for a ciphertext C on the identity ID_A at a time period i. If the recovered redundancy in M is valid, $\mathcal{S}$ returns the associated plaintext M to $\mathcal{A}$.

Semantic security against an adaptive chosen ciphertext for a KI-CLPKE scheme can be defined via the following games between two different Advesaries ($\mathcal{A}_1$ and $\mathcal{A}_2$) and Challenger $\mathcal{S}$:

- **Chosen Plaintext Security for CL-KIE on $\mathcal{A}_1$**

  Setup: $\mathcal{S}$ takes as input a security parameter k and execute the **Setup** algorithm. It returns *params* except *master-key* to $\mathcal{A}_1$.

  Phase 1: $\mathcal{A}_1$ can access a sequence of oracles: **Partial-Private-Key-Queries**, **Helper-Key-Queries**, **Secret-Value-Queries**, **Public-Key-Replace**, **Decryption-Queries**. These queries may be requested adaptively, but restricted by the rule of adversary behavior.

  Challenge: $\mathcal{A}_1$ outputs two equal length plaintext $M_0^*, M_1^* \in \mathcal{M}$ on the challenge identity ID_A^* at a time period i^*. The challenge $\mathcal{S}$ pick a random number $b \in \{0,1\}$ and generate C^* in relation to (i^*, M_b^*, ID^*). C^* is delivered to $\mathcal{A}_1$ as a target challenge.

  Phase 2: $\mathcal{A}_1$ continues to access a sequence of oracles as in Phase 1, and $\mathcal{S}$ responds to these queries as in Phase 1.

  Guess: At the end, $\mathcal{A}_1$ outputs a guess $b' \in \{0,1\}$. The adversary wins the game if $b = b'$. We define $\mathcal{A}_1's$ advantage in this game to be $Adv(\mathcal{A}_1) = 2(Pr[b = b'] - \frac{1}{2})$.

  There are a few restrictions on the $\mathcal{A}_1$ as follows:

  - $\mathcal{A}_1$ are not allowed to extract the partial private key for ID_A^*.
  - In Phase 2, we insist that $\mathcal{A}_1$ cannot make a decryption query on the challenge ciphertext C^* in the relation to the identity ID_A^* and the public key P_A^*.

- **Chosen Plaintext Security for CL-KIE on $\mathcal{A}_2$**

Setup: $\mathcal{S}$ takes as input a security parameter k and execute the *Setup* algorithm. It returns *params* to $\mathcal{A}_2$.

Phase 1: $\mathcal{A}_2$ can access a sequence of oracles: **Helper-Key-Queries**, **Public-Key-Queries**, **Decryption-Queries**. These queries may be requested adaptively, but restricted by the rule of adversary behavior.

Challenge: $\mathcal{A}_2$ outputs two equal length plaintext $M_0^*, M_1^* \in M$ on the challenge identity ID_A^* and a time period i^*. The challenger $\mathcal{S}$ pick a random number $b \in \{0,1\}$, and generate C^* in ralation to (i^*, M_b^*, ID^*). C^* is delivered to $\mathcal{A}_2$ as a target challenge.

Phase 2: $\mathcal{A}_2$ continues to access a sequence of oracles as in Phase 1, and $\mathcal{S}$ responds to these queries as in Phase 1.

Guess: At the end, $\mathcal{A}_2$ outputs a guess $b' \in \{0,1\}$. The adversary wins the game if $b = b'$. We define $\mathcal{A}_2's$ advantage in this game to be $Adv(\mathcal{A}_2) = 2(Pr[b = b'] - \frac{1}{2})$.

There are a few restrictions on the $\mathcal{A}_2$ as follows:

- The **Secret-Value-Queries** is not allowed to access if the public key for entity has been replaced.
- $\mathcal{A}_2$ are not allowed to replace the public key for ID_A^*.
- $\mathcal{A}_2$ are not allowed to extract the secret value for ID_A^*.
- In phase 2, we insist that $\mathcal{A}_2$ cannot make a decryption query on the challenge ciphertext C^* in the relation to the identity ID_A^* and the public key P_A^*.

3 KI-CLPKE Scheme

3.1 Bilinear Pairing and Bilinear Diffie-Hellman (BDH) Problem

Bilinear Pairing

Let $\mathbb{G}_1$ denotes a cyclic additive group of order q for some large prime q, let $\mathbb{G}_2$ be a cyclic multiplicative group of the same order q, We can make use of a bilinear map:$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ above these two groups which must satisfy the following properties:

- **Bilinearity:** $\hat{e}(aP, bQ) = \hat{e}(P,Q)^{ab}$, where $P, Q \in \mathbb{G}_1$, and $a, b \in \mathbb{Z}_q^*$.
- **Non-Degeneracy:** If P is the generator for $\mathbb{G}_1$, $\hat{e}(P,P)$ is the generator for $\mathbb{G}_2$.
- **Computability:** For $\forall P, Q \in \mathbb{G}_1$, $\hat{e}(P,Q)$ can be computed through an efficient algorithm in a polynomial-time.

Bilinear Diffie-Hellman(BDH) Problem

BDH Problem is for $a, b, c \in \mathbb{Z}_q$, given $P, aP, bP, cP \in \mathbb{G}_1$, to compute abc which satisfies $\hat{e}(P,Q)^{abc} \in \mathbb{G}_2$.

3.2 Construction

Setup: We can randomly select a security parameters $k \in \mathbb{Z}^+$, the Setup algorithm works as follows:

Step 1: Pick two groups $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, \times)$ of the same prime order q where $|q| = k$. Choose a generator P over $\mathbb{G}_1$ randomly, we can get a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.

Step 2: Choose a random $s \in \mathbb{Z}_q$ to compute $P_{pub} = sP$, the corresponding s can be regarded as the *master-key*: $M_{mk} = s$;

Choose a random $w \in \mathbb{Z}_q$ to compute $P_{hk} = wP$, the corresponding w can be regarded as the *master-helper-key*: $M_{hk} = w$.

Step 3: For some integer $n > 0$,we can select three cryptographic hash functions:

- $H_1 : \{0,1\}^n \to \mathbb{G}_1$.
- $H_2 : \{0,1\}^n \times \mathbb{Z}^+ \to \mathbb{G}_1$.
- $H_3 : \mathbb{G}_1 \times \mathbb{G}_2 \to \{0,1\}^n$.

The system parameters $params = (\mathbb{G}_1, \mathbb{G}_2, p, \hat{e}, n, P, P_{pub}, P_{hp}, H_1, H_2, H_3)$. The master key $M_{mk} = s$ and the master helper key $M_{hk} = w$.

The message space is $\mathcal{M} = \{0,1\}^n$, the ciphertext space is $\mathcal{C} = \{0,1\}^n \times \{0,1\}^n$, the randomness space is $\mathcal{R} = \{0,1\}^n$.

SecretValExtract $(params, ID_A)$**:** For a given identity ID_A and *params*, the algorithm outputs a random $x_A \in Z_q$ as the secret value for entity A.

PartialKeyExtrat $(params, M_{mk}, ID_A)$**:** For a given identity $ID_A \in \{0,1\}^*$ of entity A, *params* and M_{mk}, the algorithm computes $D_A = sH_1(ID_A)$.

HelperKeyUpdate $(i, ID_A, M_{hk}, params)$**:** Given a identity string ID_A and a time period $i \in \{0, \ldots, n-1\}$, the helper generates a helper key $HK_{A,i}$ which can help the private key to be updated at the time period $i \in \{0, \ldots, n-1\}$:

$$HK_{A,i} = wH_2(ID_A, i)$$

PrivateKeyExtract $(i, ID_A, HK_{A,i}, params, D_A, x_A)$**:** Given a identity ID_A, At a time period $i \in \{0, \ldots, n-1\}$, the private key is generated as:

$$\begin{aligned} S_{A,i} &= x_A H_1(ID_A) + D_A + HK_{A,i} \\ &= x_A H_1(ID_A) + sH_1(ID_A) + wH_2(ID_A, i) \end{aligned}$$

the value $S_{A,i-1}$ will be deleted subsequently.

PublicKeyExtract $(params, x_A, ID_A)$**:** Given $params$ and x_A, the algorithm outputs $P_A = \langle X_A, Y_A \rangle = \langle x_A P, x_A sP \rangle$.

Encrypt $(i, params, ID_A, P_A, M)$**:** At a time period $i \in \{0, \ldots, n-1\}$, to encrypt a plaintext $M \in \{0,1\}^n$, the algorithm does:

1) Check the equality $\hat{e}(X_A, sP) = \hat{e}(Y_A, P)$ holds. If not, output $\perp$ and abort encryption.
2) Select a random $r \in \mathbb{Z}_q$, $U = rP$.
3) Compute $\xi = \hat{e}(X_A, rH_1(ID_A))\hat{e}(P_{pub}, rH_1(ID_A))\hat{e}(P_{hk}, rH_2(ID_A, i))$.
4) Output the ciphertext: $C = \langle i, U, M \oplus H_3(U, \xi) \rangle$.

Decrypt$(i, params, S_{A,i}, C)$**:** Received the ciphertext $C = \langle i, U, V \rangle$. at the time period $i \in \{0, \ldots, n-1\}$, the algorithm performs the following steps with the Private key $S_{A,i}$:

1) Compute $\xi' = \hat{e}(U, S_{A,i})$.
2) Compute $M' = V \oplus H_3(U, \xi')$.
3) If the recovered redundancy in M is valid, then accept M' the plaintext.

4 Analysis

4.1 Security Proof

Theorem 1. *Let hash functions H_1, H_2, H_3 be random oracles. For TypeI adversary in polynomial time, suppose further that there is no* IND-CPA *adversary $\mathcal{A}_1$ that has non-negligible advangtage against the* KI-CLPKE *scheme. Then the* KI-CLPKE *is* IND-CPA *secure.*

Proof. We first deal with the *TypeI* adversary A_1. For the first type adversary A_1 is external attacker who can not get the *master-key*, Given a BDH problem (P, aP, bP, cP), we can construct a challenger $\mathcal{S}$ to compute $\hat{e}(P,P)^{abc}$ by making use of A_1 as an adversary. Now, we begin to propose the concrete proof.

Setup: Firstly, challenger $\mathcal{S}$ sets $P_{pub} = aP$ and selects $params = (\mathbb{G}_1, \mathbb{G}_2, p, \hat{e}, n, P, P_{pub}, P_{hp})$ then sends $params$ to adversary $\mathcal{A}_1$.

Phase 1: H_1 **queries:** $\mathcal{S}$ keeps a list H_1^{list} of tuples $\langle ID_j, u_j \rangle$ which is initially empty. When $\mathcal{A}_1$ issues a query on ID_i, $\mathcal{S}$ responds as follows:

- If ID_i is on H_1^{list} in a tuple $\langle ID_i, u_i \rangle$, then $\mathcal{S}$ responds with u_i. If $ID_i = ID^*$, then $\mathcal{S}$ set $H_1(ID^*) = bP$.
- Otherwise, $\mathcal{S}$ selects a random integer $u_i \in Z_p$ and stores $\langle ID_i, u_i \rangle$ into the tuple list. $\mathcal{S}$ responds with u_i.

H_2 **queries:** $\mathcal{S}$ keeps a list H_2^{list} of tuples $\langle ID_j, u_j, w_j \rangle$ which is initially empty. When $\mathcal{A}_1$ issues a query on ID_i and u_i, $\mathcal{S}$ responds as follows:

- If ID_i and u_i is on H_2^{list} in a tuple $\langle ID_i, u_i, w_j \rangle$, then β_I responds with w_i.
- Otherwise, $\mathcal{S}$ selects a random integer $w_i \in Z_p$ and stores $\langle ID_i, u_i, w_i \rangle$ into the tuple list. $\mathcal{S}$ responds with w_i.

H_3 **queries:** $\mathcal{S}$ keeps a list H_3^{list} of tuples $\langle u_j, w_j, Str_j \rangle$ which is initially empty. When $\mathcal{A}_1$ issues a query on u_i and w_i, $\mathcal{S}$ responds as follows:

- If u_i and w_i is on H_3^{list} in a tuple $\langle u_i, w_i, Str_i \rangle$, then $\mathcal{S}$ responds with Str_i.
- Otherwise, $\mathcal{S}$ selects a random integer $Str_i \in \{0,1\}^n$ and stores $\langle u_i, w_i, Str_i \rangle$ into the tuple list. $\mathcal{S}$ responds with Str_i.

Partial-Private-Key-Queries: $\mathcal{S}$ keeps a list PP^{list} of tuples $\langle ID_j, D_{A,j} \rangle$. On receiving a query **Partial-Private-Key-Queries**(ID_i), $\mathcal{S}$ responds to the query as follows:

- If $ID_i = ID^*$, $\mathcal{S}$ aborts.
- Else, if ID_i is on the list in the tuple $\langle ID_i, D_{A,i} \rangle$, then $\mathcal{S}$ responds with $D_{A,i}$
- Else, $\mathcal{S}$ first searches H_1^{list} for the tuple with ID_i. If no such tuple is found then $H_1(ID_i)$ is queried. Then $\mathcal{S}$ compute $D_{A,i} = sH_1(ID_i)$ and output $D_{A,i}$ as the answer.

Helper-Key-Queries: $\mathcal{S}$ keeps a list HK^{list} of tuples $\langle ID_j, j, HK_{A,j} \rangle$. On receiving a query **Helper-Key-Queries**$(ID_i,\ i)$, $\mathcal{S}$ responds to the query as follows:

- If $ID_i = ID^*$, $\mathcal{S}$ aborts.
- Else, if ID_i and the time period i are on the list in the tuple $\langle ID_i, i, HK_{A,i} \rangle$, then $\mathcal{S}$ responds with $HK_{A,i}$.
- Else, $\mathcal{S}$ first searches H_2^{list} for the tuple with ID_i and the time period i. If no such tuple is found then $H_2(ID_i, i)$ is queried. Then $\mathcal{S}$ compute $HK_{A,i} = wH_2(ID_i, i)$ and then output $HK_{A,i}$ as the answer.

Secret-Value-Queries: $\mathcal{S}$ keeps a list SV^{list} of tuples $\langle ID_j, x_{A,j} \rangle$. On receiving a query **Secret-Value-Queries**(ID_i), $\mathcal{S}$ responds to the query as follows:

- If $ID_i = ID^*$, $\mathcal{S}$ aborts.
- Else, if ID_i and $x_{A,i}$ is on SV^{list} in a tuple $\langle ID_i, x_{A,i} \rangle$, then $\mathcal{S}$ responds with $x_{A,i}$.
- Eles, $\mathcal{S}$ selects a random integer $x_{A,i} \in Z_q$ and stores $\langle ID_i, x_{A,i} \rangle$ into the tuple list. $\mathcal{S}$ responds with $x_{A,i}$.

Public-Key-Queries: $\mathcal{S}$ keeps a list PK^{list} of tuples $\langle ID_j, P_{A,j}\rangle$. On receiving a query **Public-Key-Queries**(ID_i), $\mathcal{S}$ responds to the query as follows:

- If ID_i is on the list in the tuple $\langle ID_i, P_{A,i}\rangle$. Then $\mathcal{S}$ responds with $P_{A,i}$.
- Otherwise $\mathcal{S}$ first searches S^{list} for the tuple with ID_i. If no such tuple is found then Secret-Value-Queries(ID_i) is queried. Then $\mathcal{S}$ compute $X_A = x_{A,i}P, Y_A = x_{A,i}sP$ and output $P_A = \langle X_A, Y_A\rangle$ as the answer.

Public-Key-Replace: Assume a query that is to replace the public key for ID_i with value $\langle X_i', Y_i'\rangle$. If $\hat{e}(X_i', P_0) = \hat{e}(Y_i', P)$, then $P_A'(\langle X_A', Y_A'\rangle)$ is a valid public key. $\mathcal{S}$ replace the public key with new values $\langle X_i', Y_i'\rangle$.

Decryption-Queries: On receiving a query **Decryption-Queries**(ID_i, C_i) where $C_i = (i, U_i, V_i)$, $\mathcal{S}$ responds to the query as follows:

- If $ID_i = ID^*$, $\mathcal{S}$ aborts.
- Else, $\mathcal{S}$ derives the private key $S_{A,i} = x_A H_1(ID_A) + sH_1(ID_A) + wH_2(ID_A, i)$, then compute $\xi_i' = \hat{e}(U, S_{A,i})$.
- Else,$\mathcal{S}$ first searches H_3^{list} for the tuple with (U_i, ξ_i'). If no such tuple is found then $H_3(U_i, \xi_i')$ is queried. Then $\mathcal{S}$ compute $M' = V \oplus H_3(U_i, \xi_i')$, and output M' as the answer.

Challenge phase: $\mathcal{A}_1$ outputs two equal length plaintext $M_0^*, M_1^* \in \mathcal{M}$ on the challenge identity ID_A^* at a time period i^*. The challenge $\mathcal{S}$ picks a random number $b \in \{0,1\}$, sets $U^* = cP$, and generates C^* in relation to (i^*, M_b^*, ID^*). C^* is delivered to $\mathcal{A}_1$ as a target challenge.

Phase 2: $\mathcal{A}_1$ continues to access a sequence of oracles as in Phase 1, and $\mathcal{S}$ responds to these queries as in Phase 1.

Guess: At the end, $\mathcal{A}_1$ outputs a guess $b' \in \{0,1\}$. The adversary wins the game if $b = b'$. We define $\mathcal{A}_1's$ advantage in this game to be $Adv(\mathcal{A}_1) = 2(Pr[b = b'] - \frac{1}{2})$.

When the challenge games begin, $\mathcal{S}$ sets $P_{pub} = aP$ as an instance of BDH problem and simulates hash functions as random oracles. During the simulation, $\mathcal{S}$ needs to guess every bit in target plaintext M_1^* with a time period i^*. $\mathcal{S}$ will set $H_1(ID_A^*) = bP$, $H_2(ID_A^*, i^*) = (h^*{}_{,i^*}P)$, $V^* = H_3(U^*, \xi^*) = H_3(cP, \xi^*)$. After that, $\mathcal{S}$ returns a simulated ciphertext $C^* = (i^*, U^*, V^*)$, which implies the parameter ξ^* is defined as:

$$\begin{aligned}\xi^* =& \hat{e}(X_A, rH_1(ID_A^*))\hat{e}(P_{pub}, rH_1(ID_A^*))\\ & \hat{e}(P_{hk}, rH_2(ID_A^*, i^*))\\ =& \hat{e}(x_A rP, bP)\hat{e}(bP, acP)\hat{e}(wP, r(h^*{}_{,i^*}P))\\ =& \hat{e}(P,P)^{abc}\hat{e}(aP, cP)^{x_A}\hat{e}(wP, (h^*{}_{,i^*})cP)\end{aligned}$$

Above all, $\mathcal{S}$ can get the solution for BDH problem, i.e. $\hat{e}(P,P)^{abc} = \xi^*(\hat{e}(aP, cP)^{-x_A}\hat{e}(wP, (h^*{}_{,i^*})cP))^{-1}$. Thus we have proved the security of the scheme for the $TypeI$ adversary through this reduction.

□

Theorem 2. *Let hash functions H_1, H_2, H_3 be random oracles. For TypeII adversary in polynomial time, suppose further that there is no* IND-CPA *adversary $\mathcal{A}_2$ that has non-negligible advangtage against the* KI-CLPKE *scheme. Then the* KI-CLPKE *is* IND-CPA *secure.*

Proof. We secondly deal with the $TypeII$ adversary A_2. For the $TypeII$ adversary is a malicious KGC attacker who can get the *master-key*, Given a BDH problem (P, aP, bP, cP), we can construct a challenger $\mathcal{S}$ to compute $\hat{e}(P,P)^{a,b,c}$ by making use of $\mathcal{A}_2$ as an adversary. Now, we begin to propose the concrete proof.

Setup: Firstly, challenger $\mathcal{S}$ selects $params = (G_1, G_2, p, \hat{e}, n, P, P_{pub}, P_{hp})$, then sends $params$ to adversary $\mathcal{A}_2$, where P_{hp} is set as aP.

Phase 1: H_1 **queries:** $\mathcal{S}$ keeps a list H_1^{list} of tuples $\langle ID_j, u_j\rangle$ which is initially empty. When $\mathcal{A}_2$ issues a query on ID_i, $\mathcal{S}$ responds as follows:

- If ID_i is on H_1^{list} in a tuple $\langle ID_i, u_i\rangle$, then $\mathcal{S}$ responds with u_i. If $ID_i = ID^*$, then $\mathcal{S}$ set $H_1(ID^*) = bP$.
- Otherwise, $\mathcal{S}$ selects a random integer $u_i \in Z_p$ and stores $\langle ID_i, u_i\rangle$ into the tuple list. $\mathcal{S}$ responds with u_i.

H_2 **queries:** $\mathcal{S}$ keeps a list H_2^{list} of tuples $\langle ID_j, u_j, w_j\rangle$ which is initially empty. When $\mathcal{A}_2$ issues a query on ID_i and u_i, $\mathcal{S}$ responds as follows:

- If ID_i and u_i is on H_2^{list} in a tuple $\langle ID_i, u_i, w_j\rangle$, then $\mathcal{S}$ responds with w_i.
- Otherwise, $\mathcal{S}$ selects a random integer $w_i \in Z_p$ and stores $\langle ID_i, u_i, w_i\rangle$ into the tuple list. $\mathcal{S}$ responds with w_i.

H_3 **queries:** $\mathcal{S}$ keeps a list H_3^{list} of tuples $\langle u_j, w_j, Str_j\rangle$ which is initially empty. When $\mathcal{A}_2$ issues a query on u_i and w_i, $\mathcal{S}$ responds as follows:

- If u_i and w_i is on H_3^{list} in a tuple $\langle u_i, w_i, Str_i\rangle$, then $\mathcal{S}$ responds with Str_i.

Table 2: Performance comparison

	CL-PKE [1]	**CL-PKE1** [6]	**CL-PKE2** [6]	**Our Scheme**
PartialKeyExtract	M	M	$2M$	$3M$
PublicKeyExtract	$2M$	M	M	$2M$
Encrypt	$M+P+E$	$2M+P$	$4M+E$	$4M+3P$
Decrypt	P	$M+P$	$M+2P$	P
Key-Insulation	$\times$	$\times$	$\times$	$\surd$

- Otherwise, $\mathcal{S}$ selects a random integer $Str_i \in \{0,1\}^n$ and stores $\langle u_i, w_i, Str_i\rangle$ into the tuple list. $\mathcal{S}$ responds with Str_i.

Helper-Key-Queries: $\mathcal{S}$ keeps a list HK^{list} of tuples $\langle ID_j, j, HK_{A,j}\rangle$. On receiving a query **Helper-Key-Queries**$(ID_i,\ i)$, $\mathcal{S}$ responds to the query as follows:

- If $ID_i = ID^*$, $\mathcal{S}$ aborts.
- Else, if ID_i and the time period i are on the list in the tuple $\langle ID_i, i, HK_{A,i}\rangle$, then $\mathcal{S}$ responds with $HK_{A,i}$.
- Else, $\mathcal{S}$ first searches H_2^{list} for the tuple with ID_i and the time period i. If no such tuple is found then $H_2(ID_i, i)$ is queried. Then $\mathcal{S}$ compute $HK_{A,i} = wH_2(ID_i, i)$ and then output $HK_{A,i}$ as the answer.

Public-Key-Queries: $\mathcal{S}$ keeps a list PK^{list} of tuples $\langle ID_j, P_{A,j}\rangle$ where $P_{A,j} = \langle X_A, Y_A\rangle$. $\mathcal{S}$ sets $X_A = aP$. On receiving a query **Public-Key-Queries**(ID_i), $\mathcal{S}$ responds to the query as follows:

- If ID_i is on the list in the tuple $\langle ID_i, P_{A,i}\rangle$. Then $\mathcal{S}$ responds with $P_{A,i}$.
- Otherwise $\mathcal{S}$ first searches S^{list} for the tuple with ID_i. If no such tuple is found then Secret-Value-Queries(ID_i) is queried. Then $\mathcal{S}$ compute $X_A = x_{A,i}P, Y_A = x_{A,i}sP$ and output $P_A = \langle X_A, Y_A\rangle$ as the answer.

Decryption-Queries: On receiving a query **Decryption-Queries**$(ID_i,\ C_i)$ where $C_i = (i, U_i, V_i)$, $\mathcal{S}$ responds to the query as follows:

- If $ID_i = ID^*$, $\mathcal{S}$ aborts.
- Else, $\mathcal{S}$ derives the private key $S_{A,i} = x_A H_1(ID_A) + sH_1(ID_A) + wH_2(ID_A, i)$, then compute $\xi_i' = \hat{e}(U, S_{A,i})$.
- Else,$\mathcal{S}$ first searches H_3^{list} for the tuple with (U_i, ξ_i'). If no such tuple is found then $H_3(U_i, \xi_i')$ is queried. Then $\mathcal{S}$ compute $M' = V \oplus H_3(U_i, \xi_i')$, and output M' as the answer.

Challenge phase: $\mathcal{A}_{II}$ outputs two equal length plaintext $M_0^*, M_1^* \in \mathcal{M}$ on the challenge identity ID_A^* at a time period i^*. The challenge $\mathcal{S}$ pick a random number $b \in \{0,1\}$, sets $U^* = cP$, and generate C^* in relation to (i^*, M_b^*, ID^*). C^* is delivered to $\mathcal{A}_2$ as a target challenge.

Phase 2: $\mathcal{A}_2$ continues to access a sequence of oracles as in Phase 1, and $\mathcal{S}$ responds to these queries as in Phase 1.

Guess: At the end, $\mathcal{A}_2$ outputs a guess $b' \in \{0,1\}$. The adversary wins the game if $b = b'$. We define $\mathcal{A}_2's$ advantage in this game to be $Adv(\mathcal{A}_2) = 2(Pr[b = b'] - \frac{1}{2})$.

When the games begin, $\mathcal{S}$ set $X_A = aP$ as an instance of BDH problem and simulate hash functions as random oracles. During the simulation, $\mathcal{S}$ need to guess every bit in target plaintext M_2^* with a time period i^*. $\mathcal{S}$ will set $H_1(ID_A^*) = bP$, $H_2(ID_A^*, i^*) = (h^*{}_{,i^*}P)$, $V^* = H_3(U^*, \xi^*) = H_3(cP, \xi^*)$. In the challenge phase, $\mathcal{S}$ returned a simulated ciphertext $C^* = (i^*, U^*, V^*)$, which implies the parameter ξ^* is defined as:

$$\begin{aligned}\xi^* =& \hat{e}(X_A, rH_1(ID_A^*))\hat{e}(P_{pub}, rH_1(ID_A^*))\\ & \hat{e}(P_{hk}, rH_2(ID_A^*, i^*))\\ =& \hat{e}(aP, bcP)\hat{e}(bP, cP)^s\hat{e}(wP, r(h^*{}_{,i^*}P))\\ =& \hat{e}(P,P)^{abc}\hat{e}(bP, cP)^s\hat{e}(wP, (h^*{}_{,i^*})cP)\end{aligned}$$

Above all, $\mathcal{S}$ can get the solution for BDH problem, i.e. $\hat{e}(P,P)^{abc} = \xi^*(\hat{e}(bP, cP)^{-s}\hat{e}(wP, (h^*{}_{,i^*})cP))^{-1}$. Thus we have proved the security of the scheme for the $TypeII$ adversary through this reduction.

□

4.2 Performance Comparison

We compare the major computational cost of our scheme with CL-PKE proposed by Al-Riyami and Paterson [1], CL-PKE1 and CL-PKE2 proposed by Cheng *et al.* [6] in Table 2. We assume both schemes are implemented on $|\ \mathbb{G}_1\ | = 160$ bits, $|\ \mathbb{G}_2\ | = 1024$ bits, $|\ p\ | = 160$ bits and hash value $= 160$ bits. We denote by M the point multiplication in $\mathbb{G}_1$, E the exponentiation in $\mathbb{G}_2$ and P the pairing computation. The other computations are trivial so we omitted them.

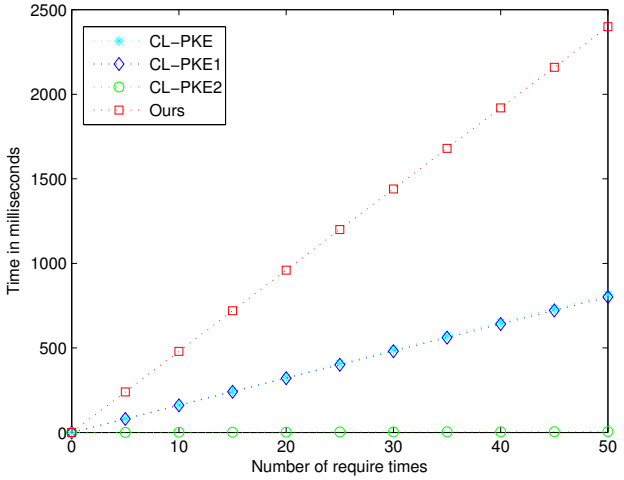

Figure 1: The comparison of the encryption computational cost

The simulation results are shown in the Figures 1, 2, 3. This experiment is executed on common desktop with 3.20GHz CPU Intel i5-4460 by using PBC Library [17]. Obviously, the ciphertext size and decryption computational cost of our scheme are comparable with the counterparts in the existing CL-PKE schemes. Indeed, our scheme is less efficient on execution time compared with CL-PKE in encryption phase, it is acceptable since the desirable key insulation function in our scheme as shown in Table 2. The additional composition of the private key in our scheme can be updated periodically, so our scheme provides extra security capability that can alleviate the problem of private key leakage. Therefore, this is a trade-off between efficiency and security capability.

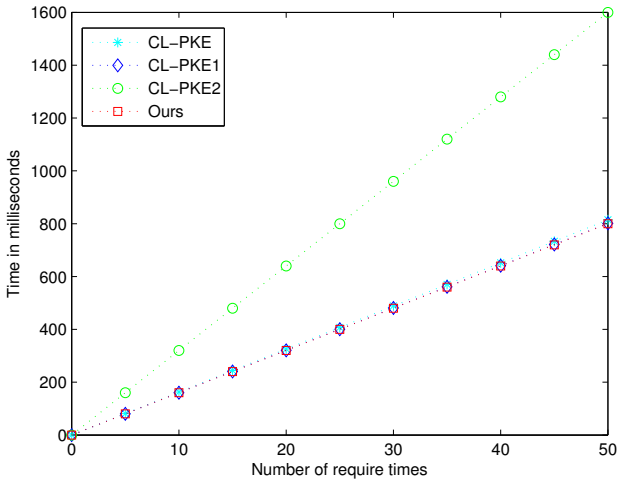

Figure 2: The comparison of the decryption computational cost

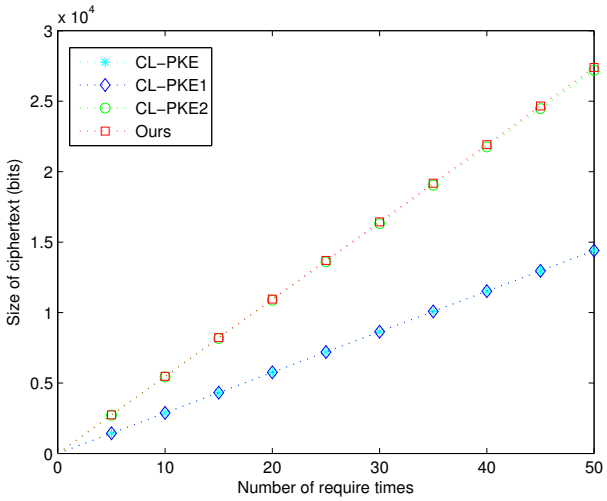

Figure 3: The comparison of the ciphertext size

5 Potential Applications

In view of the desirable merits, namely free from certificate authority and key escrow problem, and mitigating consequence of key exposure, the certificateless key insulated encryption system can be applied to a range of practical environments which are troubled by the private key exposure problem.

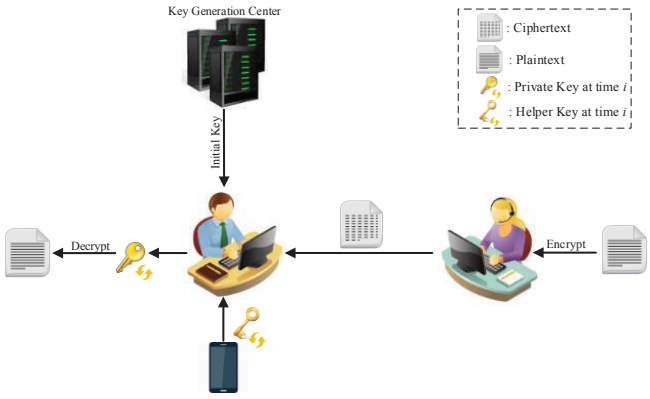

Figure 4: The system model of CL-KIE cryptosystem

In an office situation, for instance, it is common that an officer Bob might leave his seat without logging out email account. A malicious colleague could sneak into his seat and check his private mails from Alice easily even if these mails were encrypted. In contrast, the consequence of this case can be mitigated by adopting CL-KIE scheme as shown in Figure 4. To decrypt an encrypted mail, Bob should generate the latest private key at time period i with the help of Helper Key at time i that can only be produced by the helper device (his smart phone). Without

the helper, obviously, the malicious colleague can hardly decrypt these mails even if he had the access to Bob's desktop as the existing private key stored in Bob's computer has expired. Besides, other privacy-sensitive environments, such as cloud data sharing system and personal information system, are also the potentially applications for CL-KIE.

6 Conclusion

In this paper, we proposed the CL-KIE scheme by integrating the key-insulated security notion into the CL-PKE scheme in order to solve the private key exposure problem. We formalized the definition of CL-KIE scheme and proposed a concrete construction of the CL-KIE scheme. Moreover, the IND-CCA2 security proof of our scheme under BDH problem in the random oracle model was proposed. After that, we compared our scheme with three CL-PKE scheme on efficiency and security. Our scheme with key updated periodically can achieve key-escrow and key-exposure resilience which CL-PKE does not possess, while sacrifcing a little on the cost of computing time. Besides, we further extended the CL-KIE into the potential environments for the future practical application.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61003230, Grant 61370026, No. 61133016 and Grant 61272527. The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] S. S. Al-Riyami, S. Sattam, and K. G. Paterson, "Certificateless public key cryptography," in *Advances in Cryptology (ASIACRYPT'03)*, pp. 452–473, 2003.

[2] R. Anderson, *Two Remarks on Public Key Cryptology*, 1997. (`http://www.cl.cam.ac.uk/users/rja14`)

[3] J. Baek, R. Safavi-Naini, and W. Susilo, "Certificateless public key encryption without pairing," in *Information Security*, pp. 134–148, 2005.

[4] S. K. Balakrishnan and V. P. J. Raj, "Practical implementation of a secure email system using certificateless cryptography and domain name system," *International Journal of Network Security*, vol. 18, no. 1, pp. 99–107, 2016.

[5] L. Benoit and J. J. Quisquater, "On constructing certificateless cryptosystems from identity based encryption," in *Public Key Cryptography (PKC'06)*, pp. 474–490, 2006.

[6] Z. Cheng, L. Chen, L. Ling, and R. Comley, "General and efficient certificateless public key encryption constructions," in *International Conference on Pairing-Based Cryptography*, pp. 83–107, 2007.

[7] B. Dan and F. Matt, "Identity-based encryption from the weil pairing," in *Advances in Cryptology (CRYPTO'01)*, pp. 213–229, 2001.

[8] A. W. Dent, B. Libert, and K. G. Paterson, "Certificateless encryption schemes strongly secure in the standard model," in *Public Key Cryptography (PKC'08)*, pp. 344–359, 2008.

[9] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Key-insulated public key cryptosystems," in *International Conference on the Theory and Application of Cryptographic Techniques*, pp. 65–82, 2002.

[10] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances In Cryptology*, pp. 10–18, 1984.

[11] H. Goichiro, H. Yumiko, and I. Hideki, "Parallel key-insulated public key encryption," in *Public Key Cryptography (PKC'06)*, pp. 105–122, 2006.

[12] G. Itkis and L. Reyzin, "Sibir: Signer-base intrusion-resilient signatures," in *Advances in Cryptology (CRYPTO'02)*, pp. 101–116, 2002.

[13] C. Lin, Y. Li, K. Lv, and C. Chang, "Ciphertext-auditable identity-based encryption," *International Journal of Network Security*, vol. 17, no. 1, pp. 23–28, 2015.

[14] J. Liu, M. Au, and W. Susilo, "Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model," in *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, pp. 273–283, 2007.

[15] R. LRivest, S. Adi, and A. Len, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[16] B. Mihir and P. Adriana, "Protecting against key-exposure: Strongly key-insulated encryption with optimal threshold," *Applicable Algebra in Engineering, Communication and Computing*, vol. 16, no. 6, pp. 379–396, 2006.

[17] PBC Library, *The Pairing-based Cryptography Library*, Mar. 27, 2013. (`https://crypto.stanford.edu/pbc/`)

[18] W. Qiu, Y. Zhou, B. Zhu, Y. Zheng, M. Wen, and Z. Gong, "Key-insulated encryption based key predistribution scheme for wsn," in *Advances in Information Security and Assurance*, pp. 200–209, 2009.

[19] G. A. V. R. C. Rao, P. V. Lakshmi, and N. R. Shankar, "A new modular multiplication method in public key cryptosystem," *International Journal of Network Security*, vol. 15, no. 1, pp. 23–27, 2013.

[20] Y. Sun and H. Li, "Short-ciphertext and bdh-based cca2 secure certificateless encryption," *Science China Information Sciences*, vol. 53, no. 10, pp. 2005–2015, 2010.

[21] Y. Wang, X. Liu, L. Liang, W. Feng, and G. Yang, "Mitigating key escrow in attribute-based encryption," *International Journal of Network Security*, vol. 17, no. 1, pp. 94–102, Jan. 2015.

[22] Z. Wang and W. Chen, "An id-based online/offline signature scheme without random oracles for wireless sensor networks," *Personal and Ubiquitous Computing*, vol. 17, no. 5, pp. 837–841, 2013.

[23] W. Yang, F. Zhang, and L. Shen, "Efficient certificateless encryption with standing attacks from malicious kgc without using random oracles," *Security and Communication Networks*, vol. 7, no. 2, pp. 445–454, 2014.

[24] C. Youngblood, "An introduction to identity-based cryptography," *CSEP 590TU*, Mar. 2005.

[25] H. Yumiko, H. Goichiro, S. Junji, and I. Hideki, "Unconditionally secure key insulated cryptosystems: Models, bounds and constructions," in *Information and Communications Security*, pp. 85–96, 2002.

[26] X. Zhang and C. Xu, "A practical forward-secure public-key encryption scheme with untrusted update," *International Journal of Network Security*, vol. 17, no. 5, pp. 619–628, 2015.

Biography

Libo He is currently pursuing her Ph.D degree in the School of Information and Software Engineering, UESTC. Her research interests include cryptographic protocol and network security.

Chen Yuan received his B.S. degree in the School of Computer Science and Technology, Shandong University of Finance and Economics in Jun 2009. He is currently pursuing his M.S. degree in the School of Information and Software Engineering, UESTC. His research interests include cryptographic authentication protocols and network security.

Hu Xiong is an associate professor at University of Electronic Science and Technology of China (UESTC). He received his Ph.D degree from UESTC in 2009. His research interests include: cryptographic protocol, and network security.

Zhiguang Qin is the dean and professor in the School of Computer Science and Engineering, UESTC. He received his PH.D. degree from UESTC in 1996. His research interests include: information security and computer network.