Certificateless Key Insulated Encryption: Cryptographic Primitive for Achieving Key-escrow free and Key-exposure Resilience

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

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

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 secret key is inevitable. In this paper, we present a new concept called an 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.
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.

Keywords: Bilinear Pairing, Certificateless cryptography, Key insulated

1 Introduction

1.1 Motivation and Related Work

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 [14] first published the first practical public key encryption RSA algorithm whose security 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 [15] in 1985. The public key cryptosystem needs Public Key Infrastructure(PKI) to offer the authentication and validation for the public key. 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 [1]. In 2001, Dan Boneh and Matthew K. Franklin [16] 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 and 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 [2] 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 [4-9] 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 a 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-secure [17], key-insulated [10-13] and intrusion resilience [18]. In the key-evolving cryptosystem, the lifetime of the system is 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 can not 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.

1.2 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). To the best of our knowledge, this is the first CL-KIE scheme up to now. We give the formal definition and security model for CL-KIE scheme. Then we construct the 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 higher security to solve the problem of key exposure compared with the CL-PKE scheme, while sacrificing a little on the cost of computing time. This is an attractive advantage which the standard CL-PKE scheme does not possess.

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 (Certificateless Key-Insulated Encryption) 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 IND-CPA (Indistinguishability of Encyption Against Adaptive Chosen Plaintext Attacker) 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 G_1 denotes a cyclic additive group of order q for some large prime q, let G_2 be a cyclic multiplicative group of the same order q, We can make use of a bilinear map:$\hat{e} : G_1 \times G_1 \to G_2$ above these two groups which must satisfy the following properties:

- **Bilinearity:** $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$, where $P, Q \in G_1$, and $a, b \in Z_q^*$.
- **Non-Degeneracy:** If P is the generator for G_1, $\hat{e}(P, P)$ is the generator for G_2.
- **Computability:** For $\forall P, Q \in G_1$, $\hat{e}(P, Q)$ can be computed through an efficient algorithm in a polynomial-time.

Bilinear Diffie-Hellman(BDH) Problem
BDHP is for $a, b, c \in Z_q$, given $P, aP, bP, cP \in G_1$, to compute abc which satisfies $\hat{e}(P, Q)^{abc} \in G_2$.

3.2 Construction

- Setup: We can randomly select a security parameters $k \in Z^+$, the Setup algorithm works as follows:
  Step1: Pick two groups $(G_1, +)$ and $(G_2, \times)$ of the same prime order q where $|q| = k$. Choose a generator P over G_1 randomly, we can get a bilinear map $\hat{e} : G_1 \times G_1 \to G_2$.
  Setp2: Choose a random $s \in Z_q$ to compute $P_{pub} = sP$, the corresponding s can be regarded as the *maste-key*: $M_{mk} = s$;
  Choose a random $w \in Z_q$ to compute $P_{hk} = wP$, the corresponding w can be regarded as the *master-helper-key*: $M_{hk} = w$.
  Setp3: For some integer $n > 0$,we can select three cryptographic hash functions:
  - $H_1 : \{0,1\}^n \to G_1$.
  - $H_2 : \{0,1\}^n \times Z^+ \to G_1$.
  - $H_3 : G_1 \times G_2 \to \{0,1\}^n$.

  The system parameters $params = (G_1, G_2, p, \hat{e}, n, P, P_{pub}, P_{hp}, H_1, H_2, H_3)$.
  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 $i \in \{0, \ldots, n-1\}$:

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

– $\mathsf{PrivateKeyExtract}(i, ID_A, HK_{A,i}, params, D_A, x_A)$: 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.
– $\mathsf{PublicKeyExtract}(params, x_A, ID_A)$: Given $params$ and x_A, the algorithm outputs $P_A = \langle X_A, Y_A \rangle = \langle x_A P, x_A sP \rangle$.
– $\mathsf{Encrypt}(i, params, ID_A, P_A, M)$: 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 $r \in Z_q$, $U = rP$.
  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: $C = \langle i, U, M \oplus H_3(U, \xi) \rangle$.
– $\mathsf{Decrypt}(i, params, S_{A,i}, C)$: Received the ciphertext $C = \langle i, U,$ $V \rangle$. at the time period $i \in \{0, \ldots, n-1\}$, the algorithm perfoms 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 $params = (G_1, G_2, p, \hat{e}, n, P, P_{pub}, P_{hp})$ 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.
    ∗ 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 $M' = V \bigoplus H_3(U_i, \xi_i')$, 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 $\mathcal{S}$ 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 $Adv(\mathcal{A}_1) = 2(Pr[b = b'] - \frac{1}{2})$.
– **Analysis**: When the games begin, $\mathcal{S}$ set $P_{pub} = aP$ 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_1^* with a time period i^*. $\mathcal{S}$ will set $H_1(ID_A^*) = bP$, $H_2(ID_A^*, i^*) = (h^*{}_{,i^*} P)$, $V^* = H_3(U^*, \xi^*) = H_3(cP, \xi^*)$. In the challenge phase, $\mathcal{S}$ returned a simulated ciphertext $C^* = (i^*, U^*, V^*)$, which implies the parameter ξ^* is defined as:

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

Above all, $\mathcal{S}$ can get the solution for BDP 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 $TypeII$ adersary 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 $\mathcal{A}_2$.
- **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.
    - 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.
    - 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 $M' = V \bigoplus H_3(U_i, \xi_i')$, 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 $b \in \{0,1\}$ 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 $Adv(\mathcal{A}_2) = 2(Pr[b = b'] - \frac{1}{2})$.
- **Analysis**: 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 BDP 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 certificateless public key cryptography proposed by Al-Riyami and Paterson[2] in Talbe 1. We assume both schemes are implemented on $\mid G_1 \mid = 160$ bits, $\mid G_2 \mid = 1024$ bits, $\mid p \mid = 160$ bits and hash value $= 160$ bits. We denote by M the point multiplication in G_1, E the exponentiation in G_2 and P the pairing computation. The other computations are trival so we omitted them.

Table 1. Performance Comparison

	CL-PKE	Our schme
PartialKeyExtract	M	$3M$
PublicKeyExtract	$2M$	$2M$
Encrypt	$M + P + E$	$4M + 3P$
Decrypt	P	P

From Table 1, we can see that in the PublicKeyExtract and Decrypt phase our scheme has the same computational cost as CL-PKE. However, in the PrivateKeyExtract and Encypt phase our scheme is less efficient on execution time

compared with CL-PKE. Because the private key consisting of three parts in our scheme is more complicated than it in CL-PKE. 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.

5 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. We also gave the IND-CCA2 security proof of our scheme under BDH problem in the random oracle model. After that, we compared our scheme with 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.

References

1. Youngblood, C.: An Introduction to Identity-Based Cryptography. CSEP 590TU, 2005.
2. Al-Riyami, S.S., Paterson, K.G.: Certificateless Public Key Cryptography. In: Proceedings of 9th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2003), pp. 452-473. Springer Berlin Heidelberg (2003)
3. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-Insulated Public Key Cryptosystems. In: Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques Amsterdam, pp. 65-82. Springer Berlin Heidelberg (2002)
4. Baek, J., Safavi-Naini, R., Susilo, W.: Certificateless public key encryption without pairing. In: Proceedings of 8th International Conference on Information Security, ISC 2005. LNCS, vol. 3650, pp. 134-148. Springer-Verlag (2005)
5. Dent, A.W., Libert, B., Paterson, K.G.: Certificateless encryption schemes strongly secure in the standard model. In: Proceedings of 11th International Workshop on Practice and Theory in Public-Key Cryptography (PKC 2008), pp. 344-359. Springer Berlin Heidelberg (2008)
6. Libert, B., Quisquater, J.: On constructing certificateless cryptosystems from identity based encryption. In: Proceedings of 9th International Conference on Theory and Practice in Public-Key Cryptography (PKC 2006), pp. 474-490. Springer Berlin Heidelberg (2006)
7. Liu, J.K., Au, M.H., Susilo, W.: 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 (ASIACCS 2007), pp. 302-311. ACM, New York (2007)
8. Sun, Y., Li, H.: Short-ciphertext and BDH-based CCA2 secure certificateless encryption. Sci. China Inf. Sci. 53(10), 2005-2015 (2010)

9. Yang, W., Zhang, F., Shen, L.: Efficient certificateless encryption withstanding attacks from malicious KGC without using random oracles. Secur. Commun. Networks. 7(2), 445-454 (2014)
10. Bellare, M., Palacio, A.: Protecting against key exposure: strongly key-insulated encryption with optimal threshold. In: Proc. AAECC 2006, pp. 379-396. Springer-Verlag (2006)
11. Hanaoka, G., Hanaoka, Y., Imai, H.: Parallel key-insulated public key encryption. In: Proceedings of 9th International Conference on Theory and Practice in Public-Key Cryptography (PKC 2006), pp. 105-122. Springer-Verlag (2006)
12. Hanaoka, Y., Hanaoka, G., Shikata, J., Imai, H.: Unconditionally secure key insulated cryptosystems: models, bounds and constructions. In: Proceedings of 4th International Conference on Information and Communications Security (ICICS 2002). pp. 85-96. Springer-Verlag (2002)
13. Qiu, W., Zhou, Y., Zhu, B., Zheng, Y., Wen, M., Gong, Z.: Key-insulated encryption based key pre-distribution scheme for WSN. In: Proc. ISA 2009. pp. 200-209. Springer-Verlag (2009)
14. Rivestm L.R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM. 21(2), 120-126 (1978)
15. ElGamal T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Proceedings of CRYPTO 84. pp. 10-18. Springer Berlin Heidelberg (1984)
16. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Advances in Cryptology-CRYPTO 2001. pp. 213-229. Springer Berlin Heidelberg (2001)