Secure Certificateless Public Key Encryption without Redundancy

Yinxia Sun and Futai Zhang *

School of Mathematics and Computer Science
Nanjing Normal University, Nanjing 210097, P.R.China

Abstract. Certificateless public key cryptography was introduced to solve the key escrow problem in identity based cryptography while enjoying the most attractive *certificateless* property. In this paper, we present the first *secure* certificateless public key encryption (CLPKE) scheme *without redundancy*. Our construction provides optimal bandwidth and quite efficient decryption process compared with the existing CLPKE schemes. It is provably secure against adaptive chosen ciphertext attacks in the random oracle model under a slightly stronger assumption.

keywords: certificateless public key encryption, redundancy, short ciphertext, bilinear pairing, random oracle.

1 Introduction

Easy *key distribution* makes public key cryptography attract much attention from the research area. One of the most important topics in the implementation of public key cryptosystems is how to guarantee the authenticity of public keys. In traditional public key cryptography, the authenticity of public keys is ensured by certificates signed by a certificate authority (CA). But the issues associated with certificate management are quite complex and costly. In 1984, Shamir invented a new paradigm called "Identity-based Cryptography" (ID-PKC) [17] which eliminates the need for certificate by deriving public keys for users directly from their human-memorizable information, such as e-mail address and IP address. The private keys are fully generated by a Private Key Generator (PKG) which inevitably introduces the key escrow problem.

In 2003, Al-Riyami and Paterson brought forth the notion of "Certificateless Public Key Cryptography" (CL-PKC) [1] to solve the inherent key escrow problem in ID-PKC and meanwhile keep the attractive *certificateless* property. A subsequent work by Yum-Lee [20] considered a

* Project supported by the National Natural Science Foundation of China (Grant No. 60673070) and the Natural Science Foundation of Jiangsu Province (Grant No. BK2006217)

generic construction of certificateless encryption. However their construction was insecure against Type II adversaries as showed by Galindo et al. In 2005, Al-Riyami and Paterson gave another CLPKE scheme [2] which was independently broken and modified by Libert-Quisquater [14] and Zhang-Feng [21]. The Cheng-Comley scheme [6], which is very similar to Zhang-Feng scheme [21], was proved to be secure in the weak Type Ib* security model. Papers by Baek et al. [3] and Sun et al. [19] showed certificateless encryption schemes without pairing, which enjoys a better efficiency than pairing-based proposals. But it prevents users from generating their public keys independently from the Key Generator Center (KGC). Among other related results, we mention the paper by Libert-Quisquater [14]. They presented four provably secure CLPKE constructions by applying the improved Fujisaki-Okamoto transformation to weakly secure certificateless schemes. Their forth concrete construction is similar to the Shi-Li [18] scheme but is more efficient. Very recently, Dent [8] surveyed all known security models that had been proposed for CLPKE, and presented a consistent, new nomenclature for these models. In addition, there are some achievements on CLPKE that are provably secure in the standard model [15,12,9]. We may notice that, to resist chosen ciphertext attacks, most of the concrete CLPKE schemes use the Fujisaki-Okamoto-style technique [10,11], which introduces redundancies in ciphertexts. Since redundancy will aggravate the load of the cryptosystem, especially in communication cost, which is extremely disliked in the bandwidth limited network, our motivation focuses on removing redundancy of ciphertexts in certificateless setting.

Recently, Libert and Quisquater [13] showed a hybrid variant of the Boneh-Franklin identity based encryption (IBE) [5] without introducing redundancies in ciphertexts that are thus shorter than that in the original IBE scheme. Their construction was realized by combining an identity based Key Encapsulation Mechanism (KEM) and an IND-CCA secure Data Encapsulation Mechanism (DEM). We think their technique is useful for us to construct a certificateless encryption scheme *without redundancy*. Though this was also mentioned at the end of section 3 in [13], their proposal is fragile against the Type I adversary under adaptive chosen ciphertext attacks. It is because the certificateless encryption scheme they used is not secure as pointed out by Libert-Quisquater [14] and Zhang-Feng [21]. Note that "without redundancy" here means the expanded ciphertext size is zero beyond both the plaintext size and necessary randomness size for CCA security.

Our Contributions. In this paper, by extending the technique of Libert and Quisquater [13] to the certificateless setting, we present the first secure CLPKE scheme without redundancy. Our construction enjoys better performances both in ciphertext length and the decryption process than all the existing CLPKE schemes. We also give a formal security proof against adaptive chosen ciphertext attacks in the random oracle model.

Our paper proceeds as follows. Section 2 gives definitions of certificateless public key encryption, the security model and some computational problems. Our CLPKE construction is presented in Section 3 followed by the security discussion in Section 4. Then in Section 5, we compare our scheme with some other existing ones. The last section gives our concluding remarks.

2 Preliminaries

In this section, we review the definition and the security model of certificateless public key encryption (CLPKE), followed by some computational problems which form the basis of the security for our scheme.

2.1 Definition of Certificateless Public Key Encryption

A certificateless public key encryption scheme is defined by seven probabilistic, polynomial-time algorithms:

- Setup: Taking as input a security parameter k, the key generation center (KGC) runs this algorithm to generate a master key msk and a list of public parameters $params$.
- Partial-Private-Key-Extract: Taking as input msk, $params$ and a user's identity ID, KGC runs this algorithm to generate a partial private key D_{ID} for the user.
- Set-Secret-Value: Taking as input $params$ and ID, the user runs this algorithm to generate a secret value x_{ID}.
- Set-Private-Key: Taking as input $params$, D_{ID} and x_{ID}, the user runs this algorithm to generate a private key S_{ID}.
- Set-Public-Key: Taking as input $params$ and x_{ID}, the user runs this algorithm to generate a public key P_{ID}.
- Encrypt: Taking as input a message m, $params$, a user's identity ID and public key P_{ID}, a message sender runs this algorithm to return a ciphertext C.
- Decrypt: Taking as input the ciphertext C, $params$ and the private key S_{ID}, the user runs this algorithm to return a message m.

2.2 Security Model

In this paper, we employ the common security model developed by Al-Riyami and Paterson [1] (Dent afterwards strengthened the power of the Type II adversary and proposed a stronger security model for CLE. For detail, please refer to [8]). A general adversary $\mathcal{A}$ against a CLPKE scheme may carry out actions as follows:

- **Partial Private Key Extraction query** $\mathsf{PPK}(ID)$**:** On receiving such a query, the challenger $\mathcal{C}$ responds by running algorithm Partial-Private-Key-Extract to generate the partial private key D_{ID}.
- **Private Key Extraction query** $\mathsf{PrK}(ID)$**:** On receiving such a query, if the public key has not been replaced, then $\mathcal{C}$ can respond by running algorithm Set-Private-Key to generate the private key S_{ID}. But it is unreasonable to expect $\mathcal{C}$ to be able to respond to such a query if the public key has been replaced.
- **Public Key request query** $\mathsf{PK}(ID)$**:** On receiving such a query, $\mathcal{C}$ responds by running algorithm Set-Public-Key to generate the public key P_{ID} (first running Set-Secret-Value if necessary).
- **Public Key Replacement** $\mathsf{PKR}(ID)$**:** Such a query allows the adversary $\mathcal{A}$ to replace the public key of a user ID with any value of its choice. The new value will be recorded and will be used by $\mathcal{C}$ in the coming computations or responses to the adversary's queries.
- **Decryption query** $\mathsf{D}(ID, P_{ID}, C)$**:** On receiving such a query, the challenger returns the correct decryption of ciphertext C under identity ID and public key P_{ID}, even if the corresponding public key for the user ID has been replaced. This is a rather strong property for the security model (after all, the challenger may no longer know the correct private key). However, this capability gives the adversary more power in breaking the scheme. For further discussion of this feature, see [1].

The security model distinguishes two types of adversaries.

A CLPKE Type I Adversary: Such an adversary $\mathcal{A}_I$ does not have access to the master key, but may replace public keys of users with values of its choice, as well as request public keys, extract partial private and private keys, and make decryption queries. There are some restrictions on this type of adversary:

- $\mathcal{A}_I$ cannot extract the partial private or the private key for the challenge identity ID^* at any point.
- $\mathcal{A}_I$ cannot request the private key for any identity if the corresponding public key has been replaced.

– In Phase 2, $\mathcal{A}_I$ cannot make a decryption query on the challenge ciphertext C^* for the combination (ID^*, P_{ID^*}) that was used to encrypt one of the challenge plaintexts.

A CLPKE Type II Adversary: Such an adversary $\mathcal{A}_{II}$ does have access to the master key msk, but may not replace public keys of users. $\mathcal{A}_{II}$ can compute partial private keys for all users thanks to its knowledge of the master key msk. It can also request public keys, make private key extraction queries and decryption queries. The restrictions on this type of adversary are:

– $\mathcal{A}_{II}$ cannot replace public keys at any point.
– $\mathcal{A}_{II}$ cannot extract the private key for the challenge identity ID^* at any point.
– In Phase 2, $\mathcal{A}_{II}$ cannot make a decryption query on the challenge ciphertext C^* for the challenge identity ID^*.

Definition 1. *A CLPKE scheme is said to be secure against adaptive chosen ciphertext attack (IND-CCA secure) if no polynomially bounded adversary $\mathcal{A}$ of Type I or Type II has a non-negligible advantage in the following game:*

Setup: The challenger $\mathcal{C}$ takes a security parameter k as input and runs the Setup algorithm. It gives $\mathcal{A}$ the resulting public parameters *params*. If $\mathcal{A}$ is of Type I, then $\mathcal{C}$ keeps the master key msk to himself. Otherwise, he gives msk to $\mathcal{A}$.
Phase 1: $\mathcal{A}$ issues a sequence of requests described above. These queries may be asked adaptively, but are subject to the restrictions on adversary behavior defined above.
Challenge Phase: Once $\mathcal{A}$ decides that Phase 1 is over, it outputs a challenge identity ID^* with public key P_{ID^*} and two plaintexts (m_0, m_1). Note that $\mathcal{A}$ is not allowed to know the private key of ID^* in any way. $\mathcal{C}$ now picks a random bit $\beta \in \{0, 1\}$ and computes C^*, the encryption of m_β under the current public key P_{ID^*} for ID^*. Then C^* is delivered to $\mathcal{A}$.
Phase 2: Now $\mathcal{A}$ issues a second sequence of queries as in Phase 1, again subject to the constraints defined above on the adversary behavior.
Guess: Finally, $\mathcal{A}$ outputs a guess β' for β. The adversary wins the game if $\beta' = \beta$. We define $\mathcal{A}$'s advantage in this game to be $\mathsf{Adv}(\mathcal{A}) = 2(\Pr[\beta' = \beta] - \frac{1}{2})$.

Our CLPKE scheme makes use of a symmetric cipher that is chosen-ciphertext secure. A symmetric encryption scheme consists of three algorithms (K, E, D). The key generation algorithm K generates a key

$k \rightarrow \{0,1\}^{\lambda}$ for a security parameter λ. The encryption algorithm E takes a key k and a plaintext m to produce a ciphertext $c = E_k(m)$ while the decryption algorithm takes a key k and a ciphertext c to return $m/\perp = D_k(c)$.

For easy reading, we review the notion of chosen ciphertext security for a symmetric encryption scheme described by Libert and Quisquater in [13] as follows:

Definition 2. *A symmetric encryption scheme is said to be chosen-ciphertext secure if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage in the following game:*

- The challenger chooses a key $k \in \{0,1\}^{\lambda}$.
- $\mathcal{A}$ queries the encryption oracle $E_k()$ and the decryption oracle $D_k()$.
- $\mathcal{A}$ outputs two challenge messages (m_0, m_1) that were not submitted to the encryption oracle $E_k()$ or obtained from the decryption oracle $D_k()$. The challenger computes $c^* = E_k(m_\beta), \beta \in \{0,1\}$ and sends it to $\mathcal{A}$.
- $\mathcal{A}$ issues more queries as in the second step, but is not allowed to ask for the decryption of c^* and the encryption of m_0 or m_1.
- $\mathcal{A}$ outputs a guess β' for β. We define $\mathcal{A}$'s advantage in this game to be $\mathsf{Adv}(\mathcal{A}) = 2(\Pr[\beta' = \beta] - \frac{1}{2})$.

2.3 Computational Problems

Let $\mathbb{G}_1$ denote an additive group of prime order q and $\mathbb{G}_2$ a multiplicative group of the same order. We let P denote a generator of $\mathbb{G}_1$. For us, a pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties:

1. Bilinear: given $Q, W, Z \in \mathbb{G}_1$, we have $e(Q, W+Z) = e(Q, W) \cdot e(Q, Z)$ and $e(Q+W, Z) = e(Q, Z) \cdot e(W, Z)$.
2. Non-degenerate: $e(P, P) \neq 1_{\mathbb{G}_2}$.
3. Computable: for any $Q, W \in \mathbb{G}_1$, $e(Q, W)$ can be computed efficiently.

Next, we introduce some computational problems that form the basis of the security for our CLPKE scheme.

Definition 3 (Computational Diffie-Hellman (CDH) problem). *Let G_1 be as above. The CDH problem in G_1 is that given $\langle P, aP, bP \rangle$ with uniformly random choices of $a, b \in \mathbb{Z}_q^*$, compute $abP \in \mathbb{G}_1$.*

Let $\mathcal{B}$ be a CDH attacker. $\mathcal{B}$'s advantage to solve the CDH problem is defined as $\mathsf{Adv}(\mathcal{B}) = \Pr[\mathcal{B}(P, aP, bP) = abP]$. Here the probability is measured over the random choices of $a, b \in \mathbb{Z}_q^*$ and the random bits of $\mathcal{B}$.

Definition 4 (Bilinear Diffie-Hellman (BDH) problem). *Let G_1, G_2, e be as above. The BDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ is that given $\langle P, aP, bP, cP \rangle$ with uniformly random choices of $a, b, c \in \mathbb{Z}_q^*$, compute $e(P,P)^{abc} \in \mathbb{G}_2$.*

Definition 5 (Decision Bilinear Diffie-Hellman (DBDH) problem). *Let G_1, G_2, e be as above. The DBDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ is that given $\langle P, aP, bP, cP, w \rangle$ with uniformly random choices of $a, b, c \in \mathbb{Z}_q^*$ and a random element $w \in \mathbb{G}_2$, decide whether $e(P,P)^{abc} = w$.*

Definition 6 (Gap Bilinear Diffie-Hellman (Gap-BDH) problem). *Let G_1, G_2, e be as above. The Gap-BDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ is that given (P, aP, bP, cP), compute $e(P,P)^{abc}$ with the help of a DBDH oracle, which, given (P, aP, bP, cP, w), output 1 if $w = e(P,P)^{abc}$ and 0 otherwise.*

Let $\mathcal{B}$ be a Gap-BDH attacker. $\mathcal{B}$'s advantage to solve the Gap-BDH problem is defined as $\mathsf{Adv}(\mathcal{B}) = \Pr[\mathcal{B}^{\mathcal{O}_{DBDH}}(P, aP, bP, cP) = e(P,P)^{abc}]$. Here the probability is measured over the random choices of $a, b, c \in \mathbb{Z}_q^*$ and the random bits of $\mathcal{B}$.

3 Our scheme

In this section, we present a redundancy-free certificateless public key encryption scheme that provides optimal bandwidth and more efficient decryption process than all known certificateless encryption schemes.

- Setup:
  1. On input a security parameter k, this algorithm outputs a pair of bilinear groups $(\mathbb{G}_1, \mathbb{G}_2)$ of prime order q and a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
  2. Choose a generater $P \in \mathbb{G}_1$.
  3. Select a master key $s \in_R \mathbb{Z}_q^*$ and set $P_0 = sP$.
  4. Choose cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1^*, H_2 : \mathbb{G}_1^3 \times \mathbb{G}_2 \to \{0,1\}^\lambda$, as well as a chosen-ciphertext secure cipher (K, E, D) of keylength λ, where λ is polynomial in k.

  The public parameters are $params = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_0, H_1, H_2, \lambda, E, D, n \rangle$ and the master key msk is s. The plaintext space is $\mathcal{M} = \{0,1\}^n$, and the ciphertext space is $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$
- Partial-Private-Key-Extract: This algorithm takes as input $params$, msk and ID. It computes $Q_{ID} = H_1(ID)$ and outputs $D_{ID} = sQ_{ID}$ as the partial private key for the user with identity ID.

- Set-Secret-Value: This algorithm takes as input $params$ and ID. It selects $x_{ID} \in_R \mathbb{Z}_q^*$ and outputs x_{ID} as the secret value.
- Set-Private-Key: This algorithm takes as input $params$, the partial private key D_{ID} and secret value x_{ID}. It returns $S_{ID} = (x_{ID}, D_{ID})$ as the private key.
- Set-Public-Key: This algorithm takes as input $params$ and the secret value x_{ID} and returns $P_{ID} = x_{ID}P$ as the public key.
- Encrypt: To encrypt $m \in \mathcal{M}$ for the user with identity ID and public key P_{ID}, this algorithm performs as follows:
  1. Choose $r \in_R \mathbb{Z}_q^*$ and compute $U = rP$.
  2. Compute $Q_{ID} = H_1(ID)$.
  3. Compute $SK = H_2(Q_{ID}, U, rP_{ID}, e(Q_{ID}, P_0)^r)$ and $V = E_{SK}(m)$. Output $C = \langle U, V \rangle$ as the ciphertext.
- Decrypt: To decrypt the ciphertext $C = \langle U, V \rangle$ using private key $S_{ID} = (x_{ID}, D_{ID})$, this algorithm computes $SK = H_2(Q_{ID}, U, x_{ID}U, e(D_{ID}, U))$ and returns $m = D_{SK}(V)$.

4 The Security

Theorem 1. *Our CLPKE scheme is IND-CCA secure in the random oracle model, assuming that the Gap-BDH problem and the CDH problem are intractable.*

This theorem follows from two lemmas below depending on the type of adversaries.

Lemma 1. *Suppose H_1, H_2 are random oracles and there exists a Type I IND-CCA adversary $\mathcal{A}_I$ against our scheme with advantage ϵ within time t, making q_{par} partial private key extraction queries, q_{pub} public key requests, q_D decryption queries and q_i random oracle queries to H_i ($i = 1, 2$). Then, for any $0 \leqslant \nu \leqslant \epsilon$, there exists*

- *an algorithm $\mathcal{B}$ that can solve the Gap-BDH problem with advantage $\epsilon' \geqslant \frac{1}{q_1}(\epsilon - \nu)$ within time $t' \leqslant t + (q_1 + q_{par} + q_{pub})\tau_{mul} + q_2\phi + q_D(\tau_{sym} + 2\varphi)$.*
- *or an attacker that breaks the chosen-ciphertext security of the symmetric encryption scheme (K, E, D) with advantage ν within time t'.*

where τ_{mul} denotes the time for a scalar multiplication in $\mathbb{G}_1$, ϕ is the time for a call to the DBDH oracle, while τ_{sym} and φ respectively denote the time for a symmetric decryption and the one for a pairing computation.

Proof. We assume that the symmetric encryption scheme used here is chosen-ciphertext secure.

Let $\mathcal{A}_I$ be a Type I IND-CCA adversary against our CLPKE scheme and $(P, aP, bP, cP, \mathcal{O}_{DBDH})$ be an input of a random instance of the Gap-BDH problem, where $\mathcal{O}_{DBDH}$ is a decision oracle that on input (P, aP, bP, cP, w), answers 1 if $w = e(P,P)^{abc}$ and 0 otherwise. We show how to construct from $\mathcal{A}_I$ an algorithm $\mathcal{B}$ to solve the Gap-BDH problem.

$\mathcal{B}$ begins by choosing an index I uniformly at random with $1 \leqslant I \leqslant q_1$ and then gives $\mathcal{A}_I$ $params = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_0 = aP, H_1, H_2, \lambda, E, D, n \rangle$, where H_1, H_2 are random oracles controlled by $\mathcal{B}$.

$\mathcal{A}_I$ may make hash queries to random oracles H_1, H_2 at any time during its attack and $\mathcal{B}$ responds as follows:
H_1 **queries**: $\mathcal{B}$ maintains a H_1 List of tuples $\langle ID_i, Q_i, d_i \rangle$. On receiving a H_1 query on ID_i, $\mathcal{B}$ does the following:

1. If there is a tuple of the form $\langle ID_i, Q_i, d_i \rangle$ on the H_1 list, then $\mathcal{B}$ returns Q_i as answer.
2. Otherwise, if $i \neq I$, choose $d_i \in_R \mathbb{Z}_q^*$, compute $Q_i = d_i P$, add $\langle ID_i, Q_i, d_i \rangle$ to the H_1 list and return Q_i as answer.
3. If $i = I$, add $\langle ID_i, Q_i = cP, ? \rangle$ to the H_1 list and return cP as answer.

H_2 **queries**: These queries are processed using three lists L_{21}, L_{22} and L_{23} which are initially empty.

- L_{21} contains tuples $(Q_i, U_j, u_j, w_j, h_{2j})$.
- L_{22} contains tuples (Q_i, U_j, u_j, w_j) such that $(Q_i, U_j, u_j, w_j, h_{2j})$ exists in L_{21} and $\mathcal{O}_{DBDH}(Q_i, P_0, U_j, w_j) = 1$.
- L_{23} contains tuples (Q_i, U_j, u_j, h_{2j}) for which $\mathcal{B}$ has implicitly assigned a value $h_{2i} \in \{0,1\}^\lambda$ to $H_2(Q_i, U_j, u_j, w_j)$, though the value w_j such that $\mathcal{O}_{DBDH}(Q_i, P_0, U_j, w_j) = 1$ is unknown.

More precisely, when $\mathcal{B}$ receives a H_2 query on (Q, U, u, w),

- $\mathcal{B}$ first checks if L_{21} contains a tuple of the form (Q, U, u, w, h_2). If it does, h_2 is returned to $\mathcal{A}_I$.
- Otherwise, $\mathcal{B}$ submits (Q, P_0, U, w) to the oracle $\mathcal{O}_{DBDH}$ which decides whether it is a valid BDH tuple.
  - If it is, then
    - If $U = bp$ and $Q = Q_I$, we output w as the searched solution.
    - Otherwise, $\mathcal{B}$ adds (Q, U, u, w) to the list L_{22}.
    - If L_{23} contains a tuple (Q, U, u, h_2), $\mathcal{B}$ adds (Q, U, u, w, h_2) to the list L_{21} and h_2 is returned to $\mathcal{A}_I$.

- Otherwise, $\mathcal{B}$ selects $h_2 \in_R \{0,1\}^{\lambda}$, inserts the tuple (Q, U, u, w, h_2) to the list L_{21} and h_2 is returned to $\mathcal{A}_I$.

Phase 1: $\mathcal{A}_I$ launches Phase 1 of its attack by making a series of requests, each of which is either a partial private key extraction, a private key extraction, a request of a public key, a public key replacement or a decryption query. We assume that $\mathcal{A}_I$ always makes the appropriate H_1 query before making one of these requests.
Partial Private Key Extraction: $\mathcal{B}$ maintains a partial private key list of tuples $\langle ID_i, D_i \rangle$. On receiving a partial private key extraction query $\mathsf{PPK}(ID_i)$, $\mathcal{B}$ responds as follows:

1. If there is a tuple of the form $\langle ID_i, D_i \rangle$ on the partial private key list, return D_i as answer.
2. Otherwise, if $i \neq I$, search the H_1 list for a tuple $\langle ID_i, Q_i, d_i \rangle$, compute $D_i = d_i P_0$, add $\langle ID_i, D_i \rangle$ to the partial private key list and return D_i as answer.
3. If $i = I$, $\mathcal{B}$ aborts.

Private Key Extraction: $\mathcal{B}$ maintains a private key list of tuples $\langle ID_i, (x_i, D_i) \rangle$. On receiving a private key extraction query $\mathsf{PrK}(ID_i)$, $\mathcal{B}$ responds as follows:

1. If there is a tuple of the form $\langle ID_i, (x_i, D_i) \rangle$ on the private key list, return (x_i, D_i) as answer.
2. Otherwise, if $i \neq I$, run the simulation algorithm of public key request to get a tuple $\langle ID_i, P_i, x_i \rangle$ and the simulation algorithm of partial private key extraction to get a tuple $\langle ID_i, D_i \rangle$, add $\langle ID_i, (x_i, D_i) \rangle$ to the private key list and return (x_i, D_i) as answer.
3. If $i = I$, $\mathcal{B}$ aborts.

Public Key request: $\mathcal{B}$ maintains a public key list of tuples $\langle ID_i, P_i, x_i \rangle$. On receiving a query on $\mathsf{PK}(ID_i)$, $\mathcal{B}$ responds as follows:

1. If there is a tuple of the form $\langle ID_i, P_i, x_i \rangle$ on the public key list, return P_i as answer.
2. Otherwise, choose $x_i \in_R \mathbb{Z}_q^*$, compute $P_i = x_i P$, add $\langle ID_i, P_i, x_i \rangle$ to the public key list and return P_i as answer.

Public Key Replacement: $\mathcal{A}$ may replace public keys with new values of its choices and $\mathcal{B}$ records all these changes.
Decryption queries: Suppose the decryption query is $\mathsf{D}(ID_i, P_{ID_i}, C = \langle U, V \rangle)$. If the public key has not been replaced, $\mathcal{B}$ searches the public key list for $\langle ID_i, P_{ID_i}, x_i \rangle$ and computes $u_i = x_i U$. Then $\mathcal{B}$ does the following:

– It checks if (Q_i, U, u_i, w_i) exists in L_{22} for some $w_i \in \mathbb{G}_2$ (satisfying $e(U, P_{ID_i}) = e(u_i, P)$ if the public key has been replaced). If it does, $\mathcal{B}$ retrieves the tuple $(Q_i, U, u_i, w_i, h_{2i})$ from L_{21} and computes $m = D_{h_{2i}}(V)$.
– Otherwise, it tests whether L_{23} contains a tuple (Q_i, U, u_i, h_{2i}) (satisfying $e(U, P_{ID_i}) = e(u_i, P)$ if the public key has been replaced) for some $h_{2i} \in \{0,1\}^\lambda$. If it does, h_{2i} is used to do the decryption $m = D_{h_{2i}}(V)$ which is returned as a result. Otherwise, $\mathcal{B}$ chooses a random $h_2 \in \{0,1\}^\lambda$ and adds (Q_i, U, u_i, h_2) (or $(Q_i, U, ?P_{ID_i}, h_2)$ when the public key has been replaced) to L_{23}, while $m = D_{h_2}(V)$ is returned to $\mathcal{A}_I$.

Challenge Phase: $\mathcal{A}_I$ picks ID^* with public key P_{ID^*} and two messages m_0, m_1 on which it wishes to be challenged. If $ID^* \neq ID_I$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ performs the following:

1. Set $U^* = bP$.
2. If the public key has not been replaced, $\mathcal{B}$ searches the public key list for $\langle ID^*, P_{ID^*}, x^* \rangle$ and computes $u^* = x^* U^*$. Then $\mathcal{B}$ checks if $(Q_{ID^*}, U^*, u^*, h_2^*)$ exists in L_{23} for some $h_2^* \in \{0,1\}^\lambda$. (If the public key has been replaced, $e(U^*, P_{ID^*}) = e(u^*, P)$ should be satisfied.)
   (a) If it does, $\mathcal{B}$ computes $V^* = E_{h_2^*}(m_\beta)$, $\beta \in_R \{0,1\}$.
   (b) Otherwise, $\mathcal{B}$ chooses a random $h_2^* \in \{0,1\}^\lambda$, inserts $(Q_{ID^*}, U^*, u^*, h_2^*)$ (or $(Q_{ID^*}, U^*, ?P_{ID^*}, h_2^*)$ when the public key has been replaced) to L_{23} and then computes $V^* = E_{h_2^*}(m_\beta)$, $\beta \in_R \{0,1\}$.
3. Return $C^* = (U^*, V^*)$ as the challenge ciphertext.

Phase 2: $\mathcal{B}$ continues to respond to $\mathcal{A}_I$'s requests in the same way as it did in Phase 1. Note that $\mathcal{A}$ can not make queries on the partial private or the private key for ID^*. Also, no decryption query should be made on C^* for the combination of ID^* and P_{ID^*}.
Guess: Eventually, $\mathcal{A}_I$ should make a guess β' for β.

Analysis. We let $\mathsf{AskH_2}$ denote the event that $(Q_{ID^*}, U^*, u^*, w^*)$ is queried to H_2 and $\neg\mathsf{Abort}$ the event that $\mathcal{B}$ does not abort during the simulation game. Suppose $\mathsf{E} = \mathsf{AskH_2}|\neg\mathsf{Abort}$. Then

$$\Pr[\beta' = \beta] = \Pr[\beta' = \beta|\mathsf{E}]\Pr[\mathsf{E}] + \Pr[\beta' = \beta|\neg\mathsf{E}]\Pr[\neg\mathsf{E}]$$

By the definition, we know that $\Pr[\beta' = \beta] \geqslant \frac{1+\epsilon}{2}$. If the event E does not occur, the probability of correct guess for $\mathcal{A}_I$ relies on the security level of the symmetric encryption scheme we adopted here, so we have

$\Pr[\beta' = \beta|\neg\mathsf{E}] \leqslant \frac{1+\nu}{2}$, where ν denotes the advantage of the symmetric encryption attacker. Hence, it is not difficult for us to obtain

$$\frac{1+\epsilon}{2} \leqslant \Pr[\mathsf{E}] + \frac{1+\nu}{2}\Pr[\neg\mathsf{E}] = \frac{1+\nu}{2} + \frac{1-\nu}{2}\Pr[\mathsf{E}]$$

then

$$\Pr[\mathsf{E}] \geqslant \frac{\epsilon-\nu}{1-\nu} \geqslant \epsilon - \nu$$

Finally, we get

$$\epsilon - \nu \leqslant \Pr[\mathsf{E}] \leqslant \frac{\Pr[\mathsf{AskH_2}]}{\Pr[\neg\mathsf{Abort}]} \leqslant \frac{\Pr[\mathsf{AskH_2}]}{\frac{1}{q_1}}$$

$$\Pr[\mathsf{AskH_2}] \geqslant \frac{1}{q_1}(\epsilon - \nu)$$

If $\mathsf{AskH_2}$ happens, $\mathcal{B}$ can give his solution to the Gap-BDH problem directly. That is to say $\epsilon' = \Pr[\mathsf{AskH_2}]$. Hence, we obtain

$$\epsilon' \geqslant \frac{1}{q_1}(\epsilon - \nu)$$

The running time is $t' \leqslant t + (q_1 + q_{par} + q_{pub})\tau_{mul} + q_2\phi + q_D(\tau_{sym} + 2\varphi)$. where τ_{mul} denotes the time for a scalar multiplication in $\mathbb{G}_1$, ϕ is the time for a call to the DBDH oracle, while τ_{sym} and φ respectively denote the time for a symmetric decryption and the one for a pairing computation.

Lemma 2. *Suppose H_1, H_2 are random oracles and there exists a Type II IND-CCA adversary $\mathcal{A}_{II}$ against our CLPKE scheme with advantage ϵ within time t, making q_{pri} private key extraction queries, q_{pub} public key requests, q_D decryption queries and q_i random oracle queries to H_i $(i = 1, 2)$. Then, for any $0 \leqslant \nu \leqslant \epsilon$, there exists*

- *an algorithm $\mathcal{B}$ that can solve the CDH problem with advantage $\epsilon' \geqslant \frac{1}{q_1}(\epsilon - \nu)$ within time $t' \leqslant t + (q_{pri} + q_{pub})\tau_{mul} + 2q_2\varphi + q_D\tau_{sym}$.*
- *or an attacker that breaks the chosen-ciphertext security of the symmetric encryption scheme (K, E, D) with advantage ν within time t'.*

where τ_{mul} denotes the time for a scalar multiplication in $\mathbb{G}_1$, while φ and τ_{sym} respectively denote the time for a pairing computation and the one for a symmetric decryption.

Proof. We assume that the symmetric scheme used here is chosen-ciphertext secure with advantage ν.

Let $\mathcal{A}_{II}$ be a Type II IND-CCA adversary against our CLPKE scheme and (P, aP, bP) be an instance of the CDH problem. We show how to construct from A_{II} an algorithm $\mathcal{B}$ to solve the CDH problem.

$\mathcal{B}$ begins by choosing an index I uniformly at random with $1 \leqslant I \leqslant q_1$. $\mathcal{B}$ selects $s \in_R \mathbb{Z}_q^*$, computes $P_0 = sP$ and supplies $\mathcal{A}_{II}$ with the public parameters $params = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_0, H_1, H_2, \lambda, E, D, n \rangle$ together with the master key s, where H_1 and H_2 are random oracles controlled by $\mathcal{B}$.

$\mathcal{A}_{II}$ may make queries to random oracles $H_i (i = 1, 2)$ at any time during its attack and $\mathcal{B}$ responds as follows:
H_1 **queries**: $\mathcal{B}$ maintains a H_1 list of tuples $\langle ID_i, Q_i \rangle$. On receiving a H_1 query on ID_i, $\mathcal{B}$ does the following:

1. If there is a tuple of the form $\langle ID_i, Q_i \rangle$ on the H_1 list, then $\mathcal{B}$ returns Q_i as answer.
2. Otherwise, $\mathcal{B}$ chooses $Q_i \in_R \mathbb{G}_1^*$, adds $\langle ID_i, Q_i \rangle$ to the H_1 list and returns Q_i as answer.

H_2 **queries**: these queries are processed using three lists L_{21}, L_{22} and L_{23} which are initially empty.

- L_{21} contains tuples $(Q_i, U_j, u_j, w_j, h_{2j})$.
- L_{22} contains tuples (Q_i, U_j, u_j, w_j) such that $(Q_i, U_j, u_j, w_j, h_{2j})$ exists in L_{21} and $e(P_i, U_j) = e(u_j, P)$.
- L_{23} contains tuples (Q_i, U_j, w_j, h_{2j}) for which $\mathcal{B}$ has implicitly assigned a value $h_{2j} \in \{0,1\}^\lambda$ to $H_2(Q_i, U_j, u_j, w_j)$, though the value u_j such that $e(P_i, U_j) = e(u_j, P)$ is unknown.

  More precisely, when $\mathcal{B}$ receives a H_2 query on (Q, U, u, w),

- $\mathcal{B}$ first checks if L_{21} contains a tuple of the form (Q, U, u, w, h_2). if it does, h_2 is returned to $\mathcal{A}_{II}$.
- Otherwise, $\mathcal{B}$ checks whether $e(P_{ID}, U) = e(u, P)$.
  - If it is, then
    - If $U = bp$ and $Q = Q_I$, we output u as the searched solution.
    - Otherwise, $\mathcal{B}$ adds (Q, U, u, w) to the list L_{22}.
    - If L_{23} contains a tuple (Q, U, w, h_2), $\mathcal{B}$ adds (Q, U, u, w, h_2) to the list L_{21} and h_2 is returned to $\mathcal{A}_{II}$.
  - Otherwise, $\mathcal{B}$ selects $h_2 \in_R \{0,1\}^\lambda$, inserts the tuple (Q, U, u, w, h_2) to the list L_{21} and h_2 is returned to $\mathcal{A}_{II}$.

Phase 1: $\mathcal{A}_{II}$ launches Phase 1 of its attack by making a series of requests, each of which is either a private key extraction, a request of a public key or a decryption query. We assume that $\mathcal{A}_{II}$ always makes the appropriate H_1 query before making one of these requests.
Private Key Extraction: $\mathcal{B}$ maintains a private key list of tuples $\langle ID_i, (x_i, D_i)\rangle$. On receiving a private key extraction query $\mathsf{PrK}(ID_i)$, $\mathcal{B}$ responds as follows:

1. If there is a tuple of the form $\langle ID_i, (x_i, D_i)\rangle$ on the private key list, return (x_i, D_i) as answer.
2. Otherwise, if $i \neq I$, compute $D_i = sQ_i$, run the simulation algorithm of public key request to get a tuple $\langle ID_i, P_i, x_i\rangle$, add $\langle ID_i, (x_i, D_i)\rangle$ to the private key list and return (x_i, D_i) as answer.
3. If $i = I$, $\mathcal{B}$ aborts.

Public Key request: $\mathcal{B}$ maintains a public key list of tuples $\langle ID_i, P_i, x_i\rangle$. On receiving a public key request query $\mathsf{PK}(ID_i)$, $\mathcal{B}$ responds as follows:

1. If there is a tuple of the form $\langle ID_i, P_i\rangle$ on the public key list, return P_i as answer.
2. Otherwise, if $i \neq I$, choose $x_i \in_R \mathbb{Z}_q^*$, compute $P_i = x_iP$, add $\langle ID_i, P_i, x_i\rangle$ to the public key list and return P_i as answer.
3. If $i = I$, set $P_I = aP$, add $\langle ID_i, P_i, ?\rangle$ to the public key list and return aP as answer.

Decryption queries: Suppose the decryption query is $\mathsf{D}(ID_i, P_{ID_i}, C = \langle U, V\rangle)$.

- If $i \neq I$, $\mathcal{B}$ runs the simulation algorithm of private key extraction to get the private key (x_i, D_i) for ID_i. Then $\mathcal{B}$ decrypts C by using (x_i, D_i).
- Otherwise, $\mathcal{B}$ computes $w_i = e(U, D_i)$ and then does the following:
  - It checks if (Q_i, U, u_i, w_i) exists in L_{22} for some u_i. If it does, $\mathcal{B}$ retrieves the tuple $(Q_i, U, u_i, w_i, h_{2i})$ from L_{21} and computes $m = D_{h_{2i}}(V)$.
  - Otherwise, it tests whether L_{23} contains a tuple (Q_i, U, w_i, h_{2i}). In this case, h_{2i} is used to do the decryption $m = D_{h_{2i}}(V)$ which is returned as a result. Otherwise, $\mathcal{B}$ chooses a random $h_2 \in \{0,1\}^\lambda$ and adds (Q_i, U, w_i, h_2) to L_{23}, while $m = D_{h_2}(V)$ is returned to $\mathcal{A}_{II}$.

Challenge Phase: $\mathcal{A}_{II}$ picks ID^* and two messages m_0, m_1 on which it wishes to be challenged. If $ID^* \neq ID_I$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ does the following:

1. Set $U^* = bP$.
2. Check whether L_{23} contains a tuple $(Q_{ID^*}, U^*, w^*, h_2^*)$ for some $h_2^* \in \{0,1\}^\lambda$, where $w^* = e(U^*, D_{ID^*})$.
   (a) If it does, $\mathcal{B}$ computes $V^* = E_{h_2^*}(m_\beta)$, $\beta \in_R \{0,1\}$.
   (b) Otherwise, $\mathcal{B}$ chooses a random $h_2^* \in \{0,1\}^\lambda$, inserts $(Q_{ID^*}, U^*, w^*, h_2^*)$ to L_{23} and then computes $V^* = E_{h_2^*}(m_\beta)$, $\beta \in_R \{0,1\}$.
3. Return $C^* = (U^*, V^*)$ as the challenge ciphertext.

Phase 2: $\mathcal{B}$ continues to respond to $\mathcal{A}_{II}$'s requests in the same way as it did in Phase 1. Note that $\mathcal{A}_{II}$ can not make private key extraction queries on ID^*. If any decryption query is equal to C^* for ID^*, then $\mathcal{B}$ aborts.
Guess: Eventually, $\mathcal{A}_{II}$ should make a guess β' for β.

Analysis. It is identical to the *Analysis* for Lemma 1, i.e. the success probability for $\mathcal{B}$ is $\epsilon' \geqslant \frac{1}{q_1}(\epsilon - \nu)$. The running time is $t' \leqslant t + (q_{pri} + q_{pub})\tau_{mul} + 2q_2\varphi + q_D\tau_{sym}$, where τ_{mul} denotes the time for a scalar multiplication in $\mathbb{G}_1$, while φ and τ_{sym} respectively denote the time for a pairing computation and the one for a symmetric decryption.

5 Performance Comparison of the CLPKE schemes

In this section, we show that our scheme has the best performance in both bandwidth and decryption efficiency compared with all the existing CLPKE schemes. All the schemes have three major operations, i.e., Pairing (p), Scalar multiplication(s) and Exponentiation (e). When tabulating the computation efficiency, hash function and block cipher evaluations are ignored. The performance of the CLPKE schemes are listed in Table 1, where we compare the schemes on security model (Sec-Mod), computation complexity of encryption (Enc) and decryption (Dec), ciphertext length (Cipher-Len), security level (Sec-Lev) and security assumption (Sec-Asm). Note that we do not list all the CLPKE schemes but some representative ones.

Table 1. Comparison of the CLPKE schemes

Scheme	Sec-Mod	Enc	Dec	Cipher-Len	Sec-Lev	Sec-Asm
[1]	RO	3p+1s+1e	1p+1s	$L_r + 2n$	Strong T-I	GBDHP[1]
[6]	RO	1p+2s+1e	1p+2s	$L_r + 2n$	Weak T-Ib*	BDH
[18]	RO	3s+1e	1p+3s	$L_r + 2n$	Strong T-I*	k-BDHI[18]
[14]	RO	1s+2e	1p+1s+1e	$L_r + 2n$	Strong T-I	k-BDHI
[15]	ST	1p+4e	3p+1e	$3L_r + 2n$	Weak T-Ia	DBDH-1[15]
[19]	RO	6e	3e	$L_r + 2n$	Strong T-I*	CDH
[9]	ST	1p+3s+1e	4p	$3L_r + n$	Strong T-I	3-DDH[9]
Ours	RO	1p+2s+1e	1p+1s	$L_r + n$	Strong T-I*	Gap-BDH

In the table, "RO" means random oracle model and "ST" is standard model; L_r and n denote the randomness size and the plaintext size, respectively. we can see that our scheme enjoys an excellent efficiency for the receiver who only needs one pairing and one scalar multiplication on decryption. The most attractive, of course, is the optimal bandwidth: the ciphertext only has L_r more bits than the plaintext which is necessary for an encryption scheme to reach the CCA security level. As to the security level for Type I adversary (Strong T-I, Weak T-Ia, Weak T-Ib*, etc.) and Type II adversary, we refer the readers to [8].

6 Concluding Remarks

We have constructed the first *secure* certificateless public key encryption scheme *without redundancy*. Our scheme outperforms the existing CLPKE schemes on both ciphertext length and decryption process. In the random oracle model, we have proved that the new scheme is strongly secure against adaptive chosen ciphertext attacks.

However, the security of our scheme relies on the strong Gap-BDH assumption. To avoid the use of the Gap-BDH, we can apply the recent twinning technique proposed by Cash et al [7] to the construction of a CLE scheme with short ciphertext under a standard assumption. However, the twinning technique introduces more operations (usually more pairing computation) in encryption and decryption, thus seems less suitable for applications e.g. on smart card. How to design an *efficient* CLE scheme with *short ciphertext* under a *standard* assumption, this is an interesting open problem.

References

1. S. S. Al-Riyami and K. Paterson. Certificateless public key cryptography. In Asiacrypt 2003, volume 2894 of LNCS, pages 452 - 473. Springer, 2003.
2. S. S. Al-Riyami and K. Paterson. CBE from CL-PKE: A generic construction and efficient schemes. In PKC 2005, volume 3386 of LNCS, pages 398 - 415. Springer, 2005.
3. J. Baek, R. Safavi-Naini and W. Susilo. Certificateless public key encryption without pairing. In ISC 2005, volume 3650 of LNCS, pages 134 - 148. Springer, 2005.
4. K. Bentahar, P. Farshim, J. Malone-Lee and N. P. Smart. Generic constructions of identity-based and certificateless KEMs. Available from `http://eprint.iacr.org/2005/058`.
5. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In CRYPTO 2001, volume 2139 of LNCS, pages 213 - 229. Springer, 2001.
6. Z. Cheng and R. Comley. Efficient certificateless public key encryption. Available from `http://eprint.iacr.org/2005/012`.

7. D. Cash, E. Kiltz and V. Shoup, The Twin Diffie-Hellman Problem and Applications, In EuroCrypt 2008, LNCS 4965, pp. 127-145, Springer, 2008.
8. A. W. Dent. A Survey of Certificateless Encryption Schemes and Security Models. Available from `http://eprint.iacr.org/2006/211.`
9. A. W. Dent, B. Libert and K. G. Paterson. Certificateless Encryption Schemes Strongly Secure in the Standard Model. In PKC 2008, volume 4939 of LNCS, pages 344 - 359. Springer, 2008.
10. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetirc and Symmetric Encryption Schemes, In Crypto 1999, volume 1666 of LNCS, page 537 - 554. Springer, 1999.
11. E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-key Encryption at Minimal Cost. In PKC 1999, volume 1560 of LNCS, pages 53 - 68. Springer, 1999.
12. Q. Huang and D. S. Wong. Generic Certificateless Encryption in the Standard Model. Available from `http://eprint.iacr.org/2007/095`.
13. B. Libert and J. J. Quisquater. Identity based encryption without redundancy. In ACNS 2005, volume 3531 of LNCS, pages 285 - 300. Springer, 2005.
14. B. Libert and J. J. Quisquater. On constructing certificateless cryptosystems from identity based encryption. In PKC 2006, volume 3958 of LNCS, pages 474 - 490. Springer, 2006.
15. J. K. Liu, M. H. Au and W. Susilo. Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model. In Proc. ACM Symposium on Information, Computer and Communications Security. ACM Press, 2007.
16. T. Okamoto and D. Pointcheval. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In PKC 2001, volume 1992 of LNCS, page 104 - 118. Springer, 2001.
17. A. Shamir. Identity-based Cryptosystems and Signature Schemes. In Crypto 1984, volume 196 of LNCS, pages 47 - 53. Springer, 1984.
18. Y. Shi and J. Li. Provable efficient certificateless public key encryption. Available from `http://eprint.iacr.org/2005/287`.
19. Y. Sun, F. Zhang and J. Baek. Strongly Secure Certificateless Public Key Encryption without Pairing. In CANS 2007, volume 4856 of LNCS, page 194 - 208. Springer, 2007.
20. D. Yum and P. Lee. Generic Construction of Certificateless Encryption. In ICCSA 2004, volume 3043 of LNCS, pages 802 - 811. Springer, 2004.
21. Z. Zhang and D. Feng. On the security of a certificateless public-key encryption. Available from `http://eprint.iacr.org/2005/426`.