# Provable Efficient Certificateless Public Key Encryption

Yijuan Shi and Jianhua Li

Department of Electronic and Engineering,
Shanghai Jiao Tong University,
Room 408, Building 1, No. 33, Leshan RD, Shanghai, China
cbzsyj130@sohu.com, lijh888@sjtu.edu.cn

**Abstract.** Certificateless public key cryptography was introduced to overcome the key escrow limitation of the identity-based cryptography. It combines the advantages of the identity-based cryptography and the traditional PKI. Recently, Dae Hyun Yum1 and Pil Joong Lee have proposed a generic series construction model of certificateless public key encryption (CL-PKE) which is built from generic primitives: identity-based encryption and public key encryption. However, this model pays much attention on the generic construction and neglects the nice properties of the bilinear pairings. In this paper, we propose an efficient CL-PKE scheme which is based on the nice algebraic properties of Weil pairing. The scheme works in a kind of parallel model and it is more efficient on computation or published public key information than the existing schemes.

## 1 Introduction

Traditionally, a Public Key infrastructure (PKI) is used to provide an assurance to the user about the relationship between a public key and the identity of the holder of the corresponding private key by certificates. However, a PKI faces may challenges in the practice, especially the scalability of the infrastructure and the management of the certificates. To simplify the management of certificates, Shamir [1] proposed identity-based public key cryptography (ID-PKC) in which the public key of each party is derived directly from certain aspects of its identity, for example, an IP address belonging to a network host, or an e-mail address associated with a user. Private keys are generated for entities by a trusted third party called Key Generation Center (KGC). For a long while it was an open problem to obtain a secure and efficient identity based encryption (IBE) scheme. Until 2001, Boneh and Franklin [2] presented an efficient and provably secure identity-based encryption scheme (BF-IBE) using the bilinear pairings on elliptic curves.

The direct derivation of public keys in ID-PKC eliminates the need for certificates and some of the problems associated with them. However, the dependence on a KGC who can generate private keys inevitably introduces key escrow to the identity-based cryptography. Then in [3] Al-Riyami and Patersion introduced the notion of Certificateless Public Key Cryptography (CL-PKC). CL-PKC can overcome the key escrow limitation of ID-PKC without introducing certificates and the management overheads that this entails. It combines the advantages of the ID-PKC and the PKI.

In this paper, we concentrate on the certificateless public key encryption (CL-PKE) schemes. So far almost all the CL-PKE schemes [3,4,5,6] are based on the BF-IBE scheme. Recently, Dae Hyun Yum and Pil Joong Lee [9] have proposed a generic series construction of CL-PKE which is built from generic primitives: identity-based encryption and public key encryption. The CL-PKE scheme in [4] is an instance of such model. However, this model pays much attention on the generic construction and neglects the nice properties of the bilinear pairings. In this paper, we propose an efficient CL-PKE scheme which is based on the nice algebraic properties of Weil pairing. The scheme works in a kind of parallel model and it is more efficient on computation or published public key information than the existing schemes.

The paper is organized as follows: First we review the concepts of CL-PKE and its security model. In section 3, we introduce some mathematic basis of bilinear maps. Then we present our new efficient CL-PKE scheme in section 4 and prove its security. In section 5, we compare our scheme with the existing CL-PKE schemes on performance. Finally, section 6 gives conclusions.

## 2 Certificateless Public Key Encryption

In this section, we review the definition and security model for CL-PKE from [3].

**Definition 1.** [3] A CL-PKE scheme is specified by seven algorithm (Setup, Partial-Private-Key-Extract, Set-Secret-Value, Set-Private-Key, Set-Public-Key, Encrypt, Decrypt) such that:

- **Setup** is a probabilistic algorithm that takes security parameter $\kappa$ as input and returns the system parameters *params* and the *masterkey*. The system parameters include a description of the message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$.
- **Partial-Private-Key-Extract** is a deterministic algorithm which takes $params, masterkey$ and an identifier for entity A, $ID_A \in \{0,1\}^n$, as inputs. It returns a partial private key $D_A$ .
- **Set-Secret-Value** is a probabilistic algorithm that takes as input *params* and outputs a secret value $x_A$.
- **Set-Private-Key** is a deterministic algorithm that takes *params*, $D_A$ and $x_A$ as inputs. The algorithm returns $S_A$, a (full) private key.
- **Set-Public-Key** is a deterministic algorithm that takes *params* and $x_A$ as inputs and outputs a public key $P_A$.
- **Encrypt** is a probabilistic algorithm that takes *params*, $M \in \mathcal{M}$, $x_A$ and $ID_A$ as inputs and returns either a ciphertext $C \in \mathcal{C}$ or the null symbol $\perp$ indicating an encryption failure.

– **Decrypt** is a deterministic algorithm that takes as inputs *params*, $C \in \mathcal{C}$ and $S_A$. It returns a message $M \in \mathcal{M}$ or a message $\perp$ indicating a decryption failure.

Algorithms **Set-Private-Key** and **Set-Public-Key** are normally run by an entity A for himself, after running **Set-Secret-Value**. Usually, A is the only entity in possession $S_A$ and $x_A$. Algorithms **Setup** and **Partial-Private-Key-Extract** are usually run by a trusted third party, called Key Generation Center (KGC) [3].

### 2.1 Security Model for CL-PKE

Al-Riyami and Patersion presented the full IND-CCA security model for CL-PKE in [3]. The following is the actions that an general adversary $\mathcal{A}$ against a CL-PKE scheme may carry out and discuss how each action should be handled by the challenger $\mathsf{C}$ for that adversary.

1. **Extract partial private key of entity A:** Challenger $\mathsf{C}$ responds by running algorithm **Partial-Private-Key-Extract** to generate the partial private key $D_A$ for entity A.

2. **Extract private key for entity A:** If A's public key has not been replaced then $\mathsf{C}$ can respond by running algorithm **Set-Private-Key** to generate the private key $S_A$ for entity A. But it is unreasonable to expect $\mathsf{C}$ to be able to respond to such a query if $\mathcal{A}$ has already replaced A's public key.

3. **Request public key of entity A:** $\mathsf{C}$ responds by running algorithm **Set-Public-Key** to generate the public key $P_A$ for entity A (first running **Set-Secret-Value** for A if necessary).

4. **Replace public key of entity A:** The adversary $\mathcal{A}$ can repeatedly replace the public key $P_A$ for any entity A with any value $P_0$ of its choice. The current value of an entity's public key is used by $\mathsf{C}$ in any computations or responses to the adversary's requests.

5. **Decryption query for ciphertext C and entity A:** In the model of [3], adversary can issue a decryption query for any entity and any ciphertext. It is assumed in [3] that $\mathsf{C}$ should properly decrypt ciphertexts, even for those entities whose public keys have 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, it ensures that the model captures the fact that changing an entity's public key to a value of the adversary's choosing may give that adversary an advantage in breaking the scheme. For further discussion of this feature, see [3].

The IND-CCA security model of [3] distinguishes two types of adversary. A type I adversary $\mathcal{A}_I$ is able to change public keys of entities at will, but does not have access to the *masterkey*. A Type II adversary $\mathcal{A}_{II}$ is equipped with the *masterkey* but is not allowed to replace public keys of entities. This adversary models security against an eavesdropping KGC.

**CL-PKE Type I IND-CCA Adversary:** Such an adversary $\mathcal{A}_I$ does not have access to the *masterkey*. However, $\mathcal{A}_I$ may request public keys and replace public keys with values of its choice, extract partial private and private keys and make decryption queries, all for identities of its choice. As discussed above, we make several natural restrictions on such a Type I adversary:

1. Adversary $\mathcal{A}_I$ cannot extract the private key for $ID_{ch}$ at any point.

2. Adversary $\mathcal{A}_I$ cannot request the private key for any identifier if the corresponding public key has already been replaced.

3. Adversary $\mathcal{A}_I$ cannot both replace the public key for the challenge identifier $ID_{ch}$ before the challenge phase and extract the partial private key for $ID_{ch}$ in some phase.

4. In Phase 2, $\mathcal{A}_I$ cannot make a decryption query on the challenge ciphertext $C^*$ for the combination $(ID_{ch}, P_{ch})$ that was used to encrypt $M_b$.

**CL-PKE Type II IND-CCA Adversary:** Such an adversary $\mathcal{A}_{II}$ does have access to the *masterkey*, but may not replace public keys of entities. $\mathcal{A}_{II}$ can compute partial private keys for himself, given the *masterkey*. It can also request public keys, make private key extraction queries and decryption queries, both for identities of its choice. The restrictions on this type of adversary are:

1. Adversary $\mathcal{A}_{II}$ cannot replace public keys at any point.

2. Adversary $\mathcal{A}_{II}$ cannot extract the private key for $ID_{ch}$ at any point.

3. In Phase 2, $\mathcal{A}_{II}$ cannot make a decryption query on the challenge ciphertext $C^*$ for the combination $(ID_{ch}, P_{ch})$ that was used to encrypt $M_b$.

**Definition 2.** A CL-PKE scheme is said to be 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 $\mathsf{C}$ takes a security parameter $\kappa$ as input and runs the Setup algorithm. It gives $\mathcal{A}$ the resulting system parameters *params*. If $\mathcal{A}$ is of Type I, then $\mathsf{C}$ keeps the *masterkey* to himself, otherwise, he gives the *masterkey* 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 rules on adversary behavior defined above.

**Challenge Phase:** Once $\mathcal{A}$ decides that Phase 1 is over it outputs the challenge identifier $ID_{ch}$ and two equal length plaintexts $M_0, M_1 \in \mathcal{M}$. Again, the adversarial constraints given above apply. $\mathsf{C}$ now picks a random bit $b \in \{0, 1\}$ and computes $C^*$, the encryption of $M_b$ under the current public key $P_{ch}$ for $ID_{ch}$. Then $C^*$ is delivered to $\mathcal{A}$.

**Phase 2:** Now $\mathcal{A}$ issues a second sequence of requests as in Phase 1, again subject to the rules on adversary behavior above.

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

IND-CPA security is similar to IND-CCA security except that the adversary is more limited. That is the adversary cannot issue decryption queries while attacking the scheme.

## 3 Mathematic Basic

Before presenting the new CL-PKE scheme, we first review a few concepts related to bilinear maps. Let $E/F_q$ be an elliptic curve and $m = \#E(F_q)$ be the group order of the curve. Let $n$ be a prime such that $n \mid m$ and $n \nmid q$ . Then the group of $n$-torsion points has the structure $E[n] \cong Z_n \oplus Z_n$ and is thus generated by two elements, say $P_1$ and $P_2$ ($< P_1 > \neq < P_2 >$) . We can denote the elements in the set of $E[n]$ using the form$aP_1 + bP_2$, $a, b \in Z_n^*$ . Denote the group generated by $P_1$ by $G_1$ and the group generated by $P_2$ by $G_2$, i.e. $G_1 =< P_1 >$ and $G_2 =< P_2 >$. $\psi$ is an isomorphism from $G_2$ to $G_1$ with $\psi(P_2) = P_1$. The Weil pairing is a function [13]:

$e_n : E[n] \times E[n] \to \mu_n$.

$e_n$ maps to the group $\mu_n$ of $n$th roots of unity, which is a cyclic group of order $n$ as well. Denote this group by $G_T$. The following are some useful properties of the Weil Pairing:

1. Identity: For all $P \in E[n]$, $e_n(P, P) = 1$.
2. Alternation: For all $P, Q \in E[n]$, $e_n(P, Q) = e_n(Q, P)^{-1}$.
3. Bilinearity: For all $P, Q, R \in E[n]$, $e_n(P, +Q, R) = e_n(P, R) + e_n(Q, R)$, and $e_n(P, Q + R) = e_n(P, Q) + e_n(P, R)$.
4. Non-degeneracy: For all $P \in G_1$ and $Q \in G_2$, $e_n(P, Q) \neq 1$.
5. Computable: For all $P, Q \in E[n]$, $e_n(P, Q)$ is computable in polynomial time.

Note that from [11], we can either assume that $\psi$ is efficiently computable or make our security proof relative to some oracle which computes $\psi$.

In the following, we consider some problems.

**co-BIDH Assumption:** For $a, b, c \in_R Z_q^*, P_2 \in G_2^*, P_1 = \psi(P_2) \in G_1^*, e_n$, given $(P_1, P_2, aP_2, bP_2)$, to compute $e_n(P_1, P_2)^{a^{-1}b}$ is hard.

**k-BCAA1 Assumption:** [8] For an integer $k$, and $x \in_R Z_n^*, P_2 \in G_2^*, P_1 = \psi(P_2) \in G_1^*, e_n$, given $(P_1, P_2, xP_2, h_0, (h_1, \frac{1}{h_1+x}P_2), ...., (h_k, \frac{1}{h_k+x}P_2))$ where $h_i \in_R Z_q^*$ and different from each other for $0 \leq i \leq k$, to compute $e_n(P_1, P_2)^{1/(x+h_0)}$ is hard.

**k-BDHI Assumption:** [8,12] For an integer $k$, and $x \in_R Z_n^*, P_2 \in G_2^*, P_1 = \psi(P_2) \in G_1^*, e_n$, given ($P_1$, $P_2$, $xP_2$,$x^2P_2$,...$x^kP_2$), to compute $e_n(P_1,P_2)^{1/x}$ is hard.

In [8] Chen and Cheng have proved that the following relationship between the k-BCAA1 problem and the k-BDHI problem.

**Theorem 1.** *[8] If there exists a polynomial time algorithm to solve (k-1)-BDHI, then there exists a polynomial time algorithm for k-BCAA1. If there* *exists a polynomial time algorithm to solve (k-1)-BCAA1, then there exists a polynomial time algorithm for k-BDHI.*

## 4 A New CL-PKE Scheme

Inspired by the provable secure SK-IBE scheme [7, 8], we propose a new CL-PKE scheme. We describe our new scheme in a similar method of [2]. First, we give a basic CL-PKE scheme which is only IND-CPA secure. Then in the next section, we will extend the basic scheme to the full scheme which is secure against an IND-CCA attack using a technique due to Fujisaki-Okamoto [10].

### 4.1 BasicCL-PKE

Our basic scheme is consisted of the following algorithms.

**Setup:** Given a security parameter $\kappa$, the generator takes the following steps.

1. Generate a Weil pairing $e : E[q] \times E[q] \to G_T$ with $E[q] = G_1 \oplus G_2$ and an isomorphism $\psi$ from $G_2$ to $G_1$. Pick a random generator $P_2 \in G_2^*$ and set $P_1 = \psi(P_2)$.

2. Pick a random $s \in Z_q^*$ and compute $P_{pub} = sP_1$.

3. Compute $g = e(P_1, P_2)$.

4. Pick five cryptographic hash functions $H_1 : \{0,1\}^* \to Z_q^*$ and $H_2 : G_T \to \{0,1\}^n$,for some $n$.

The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = E_n \times \{0,1\}^n$. The system parameters are $params =< q, G_1, G_2, G_T, e, n, P_1, P_2, g, P_{pub}, H_1, H_2 >$. The $masterkey$ is $s$.

**Partial-Private-Key-Extract:** The algorithm takes as input an identifier $ID \in \{0,1\}^*$, $params$ and the $masterkey$ $s$ and returns the partial private key $D_{ID} = \frac{1}{H_1(ID)+s} P_2$.

**Set-Secret-Value:** The algorithm takes as inputs $params$ and identifier $ID$, selects a random $x_{ID} \in Z_q^*$ and outputs $x_{ID}$ as the entity's secret value.

**Set-Private-Key:** The algorithm takes an inputs $params$, entity $ID$'s partial private key $D_{ID}$ and secret value $x_{ID}$. The output of the algorithm is the pair $S_{ID} =< D_{ID}, x_{ID} >$ .

**Set-Public-Key:** The algorithm takes $params$ and entity $ID$'s secret value $x_{ID}$ as inputs and constructs $ID$'s public key as $P_{ID} = x_{ID} P_2$.

**Encrypt:** To encrypt $M \in \mathcal{M}$ for entity $ID$ with the public key $P_{ID}$, perform the following steps:

1. Check that $P_{ID}$ is in $G_2^*$, if not output $\perp$. This checks the validity of the public key.

2. Compute $Q_{ID} = H_1(ID)P_1 + P_{pub}$.

3. Choose random values $r_1$ and $r_2$ and compute the ciphertext:

$$C =< r_1 Q_{ID} + r_2 P_{ID}, M \oplus H_2(g^{(r_1+r_2)}) >$$

**Decrypt:** Suppose $C =< U, V >$. To decrypt this ciphertext using the private key $S_{ID} =< D_{ID}, x_{ID} >$ compute:

$$M = V \oplus H_2(e(U, D_{ID} - \frac{1}{x_{ID}} P_1)).$$

According to the Weil Pairing's properties, we know $e(P_1, P_1) = 1, e(P_2, P_2) = 1$, and $e(P_2, -P_1) = e(P_1, P_2)$. Hence the consistency of the scheme can be verified by

$$\begin{aligned} e(U, D_{ID} - \tfrac{1}{x_{ID}} P_1) &= e(r_1 Q_{ID} + r_2 P_{ID}, D_{ID} - \tfrac{1}{x_{ID}} P_1) \\ &= e(r_1(H_1(ID) + s)P_1 + r_2 x_{ID} P_2, \tfrac{1}{H_1(ID)+s} P_2 - \tfrac{1}{x_{ID}} P_1) \\ &= e(r_1(H_1(ID) + s)P_1, \tfrac{1}{H_1(ID)+s} P_2) e(r_2 x_{ID} P_2, -\tfrac{1}{x_{ID}} P_1) \\ &= e(P_1, P_2)^{r_1} e(P_1, P_2)^{r_2} \\ &= g^{(r_1+r_2)} \end{aligned}$$

### 4.2 Security of BasicCL-PKE

In this section, we study the security of the BasicCL-PKE scheme. The following theorems show that Basic-PKE scheme is a CPA secure certificateless encryption scheme.

**Theorem 2.** *If there exists a Type-I IND-CPA adversary $\mathcal{A_I}$ against BasicCL-PKE with advantage, then there exists an adversary $\mathcal{B}$ which can solve the k-BCAA1 problem with non-negligible advantage in the random oracle model.*

**Theorem 3.** *If there exists a Type-II IND-CPA adversary $\mathcal{A_{II}}$ against BasicCL-PKE with advantage, then there exists an adversary $\mathcal{B}$ which can solve the co-BIDH problem with non-negligible advantage in the random oracle model.*

To prove the theorem 2, we define the following public key encryption scheme called BasicPub-I.

**BasicPub-I** The scheme includes the following algorithms:

**Key-generation:** Given a security parameter $\kappa$, the generator takes the following steps.

1. Generate the parameters $< q, G_1, G_2, G_T, e, P_1, P_2, g >$ which are identical to the ones of the BasicCL-PKE.

2. Pick a random $s \in Z_q^*$ and compute $P_{pub} = sP_1$. Randomly choose different elements $h_i \in Z_q^*$ and compute $\frac{1}{h_i+s} P_2$ for for $0 \leq i < q_1$.

3. Pick a random $x \in Z_q^*$ and compute $P_{ID} = xP_2$.

4. Pick a hash function $H_2 : G_T \to \{0, 1\}^n$ for some $n$.

The public parameters are $K_{pub-I} =< q, G_1, G_2, G_T, e, n, P_1, P_2, g, P_{pub}, x, P_{ID}, h_0, (h_1, \frac{1}{h_1+s} P_2), (h_2, \frac{1}{h_2+s} P_2), ..., (h_{q_1-1}, \frac{1}{h_{q_1-1}+s} P_2), H_2 >$ and the private key is $K_{pri-I} = \frac{1}{h_0+s} P_2$.

**Encrypt:** To encrypt $M \in \mathcal{M}$, perform the following steps:

1. Check that $P_{ID}$ is in $G_1^*$, if not output $\perp$. This checks the validity of the public key.

2. Choose two random $r_1, r_2 \in Z_q^*$ and compute the ciphertext:

$$C =< r_1(h_0P_1 + P_{pub}) + r_2P_{ID}, M \oplus H_2(g^{(r_1+r_2)})$$

**Decrypt:** Suppose $C =< U, V >$. To decrypt this ciphertext using the private key $K_{pri-I}$ compute:

$$M = V \oplus H_2(e(U, K_{pri-I} - x^{-1}P_1)).$$

**Proof of Theorem 2** This theorem straightforwardly follows from the following lemma 1 and lemma 2.

**Lemma 1:** *Let $H_1$ is a random oracle. Suppose $\mathcal{A_I}$ is an IND-CPA adversary that has advantage $\varepsilon$ against BasicCL-PKE. Suppose $\mathcal{A_I}$ makes at most $q_1$ queries to $H_1$. Then there is an IND-CPA adversary $\mathcal{B}$ that at lest $\varepsilon/q_1$ against BasicPub-I.*

Proof: Let $\mathcal{A_I}$ be a Type I advesary against the new BasicCL-PKE. Suppose $\mathcal{A_I}$ has advantage $\varepsilon$ and makes $q_1$ queries to random oracle $H_1$. We show how to construct from $\mathcal{A_I}$ an IND-CPA adversary $\mathcal{B}$ against the BasicPub-I. Let $\mathcal{C}$ denote the challenger against $\mathcal{B}$ for BasicPub-I. The challenger $\mathcal{C}$ begins by supplying $\mathcal{B}$ with a public key: $K_{pub-I} =< q, G_1, G_2, G_T, e, n, P_1, P_2, g, P_{pub}, x, P_{ID}, h_0,$ $(h_1, \frac{1}{h_1+s}P_2), (h_2, \frac{1}{h_2+s}P_2), ..., (h_{q_1-1}, \frac{1}{h_{q_1-1}+s}P_2), H_2 >$

Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}_I$ in an IND-CPA game as follows:

**Setup:** $\mathcal{B}$ chooses an index $I$ with $1 \leq I \leq q_1$. Then $\mathcal{B}$ gives $\mathcal{A_I}$ the BasicCL-PKE public parameters $params =< q, G_1, G_2, G_T, e, n, P_1, P_2, g, P_{pub}, H_1, H_2 >$. Here $< q, G_1, G_2, G_T, e, n, P_1, P_2, g, P_{pub}, H_2 >$ are taken from $K_{pub-I}$ and $H_1$ is a random oracle controlled by $\mathcal{B}$ as described below.

$H_1$**-queries:** At any time algorithm $\mathcal{A_I}$ can query the random oracle $H_1$. To response to these queries $\mathcal{B}$ maintains a list of tuples $< ID_i, h_i, x_i >$. We refer to this list as the $H_1^{list}$. When $\mathcal{A}$ queries the oracle $H_1$ at a point $ID_i$ algorithm $\mathcal{B}$ responds as follows:

1. If the query $ID_i$ already appears on the $H_1^{list}$ in a tuple $< ID_i, h_i, x_i >$, then algorithm $\mathcal{B}$ responds with $H_1(ID_i) = h_i$.

2. Otherwise, if the query is on the $Ith$ distinct ID, then $\mathcal{B}$ return $H_1(ID_I) = h_0$ and add the tuple $< ID_I, h_0, \perp >$ into the $H_1^{list}$.

3. Otherwise, $\mathcal{B}$ selects a random integer $h_i(i > 0)$ from $K_{pub-I}$ which has not been chosen by $\mathcal{B}$ and returns $H_1(ID_i) = h_i$. Then $\mathcal{B}$ chooses a random $x_i \in Z_q^*$ and adds the tuple $< ID_i, h_i, x_i >$ to the $H_1^{list}$.

**Phase 1:** Now $\mathcal{A_I}$ launches Phase 1 of its attack by making a series of queries. B replies to these queries as follows:

**Partial Private Key Extraction:** Suppose the query is on $ID_i$. There are two cases:

If $i = I$, $\mathcal{B}$ aborts **(Event 1)**.

If $i \neq I$, then $\mathcal{B}$ replies with $\frac{1}{h_i+s}P_2$.

**Private Key Extraction:** Suppose the query is on $ID_i$. There are two cases:

If $i = I$, $\mathcal{B}$ aborts **(Event 2)**.

If $i \neq I$, then $\mathcal{B}$ replies with $< \frac{1}{h_i+s}P_2, x_i >$.

**Request for Public Key:** If the query is on $ID_I$ then $\mathcal{B}$ returns $P_{ID}$. Otherwise, if the request is on $ID_i$ with $i \neq I$, then $\mathcal{B}$ outputs $x_i P_2$.

**Replace Public Key:** Suppose the query is to replace the public key for $ID_i$ with value $P_i'$. $\mathcal{B}$ responds as follows.

1. If no tuple corresponding to $ID_i$ exists on the $H_1^{list}$, $\mathcal{B}$ follows the $H_1 - queries$ algorithm to create the tuple with $x_i = \perp$ and sets $P_{ID_i} = P_i'$.

2. Otherwise, $\mathcal{B}$ updates $P_{ID_i}$ with $P_i'$ and sets $x_i = \perp$.

**Challenge Phase:** At some point, $\mathcal{A_I}$ decides to end Phase 1 and picks $ID_{ch}$ and two messages $M_0, M_1$ on which it wants to be challenged. $\mathcal{B}$ responds as follows.

1. If $ID_{ch} \neq ID_I$ then $\mathcal{B}$ aborts **(Event 3)**.

2. Otherwise, $ID_{ch} = ID_I$ and $\mathcal{B}$ gives $\mathcal{C}$ the pair $M_0, M_1$ as the messages on which it wishes to be challenged. $\mathcal{C}$ responds with the challenge ciphertext $C =< U, V, W >$ which is the BasicPub-I encryption of $M_b$ for a random $b \in \{0, 1\}$. Then $\mathcal{B}$ delivers $C$ to $\mathcal{A_{II}}$. It is easy to see that $C$ is the CL-PKE encrypton of $M_b$ for identifier $ID_I$ with public key $P_{ID}$.

**Phase 2:** $\mathcal{B}$ responds to queries as in Phase 1.

**Guess:** Eventually, $\mathcal{A_{II}}$ outputs $b'$ for $b$. Algorithm $\mathcal{B}$ outputs $b'$ as its guess for $b$.

If the algorithm $\mathcal{B}$ does not abort during the simulation then algorithm $\mathcal{A_I}$'s view is identical to its view in the real attack. To complete the proof it remains to calculate the probability that $\mathcal{B}$ does not abort during the simulation. $\mathcal{B}$ could abort when one of the following events happens:(1)Event 1: $\mathcal{A_I}$ queries the Partial Private key Extraction for $ID_I$ at some point; (2)Event 2: $\mathcal{A_I}$ queries the Private key Extraction for $ID_I$ at some point; (3) Event 3: $\mathcal{A}$ does not choose $ID_I$ as $ID_{ch}$. Notice that the $\neg event3$ implies that $\neg Event1$ and $\neg Event2$ (if $\mathcal{A_I}$ choose $ID_{ch}$ equal to $ID_I$, then no partial private key extraction and private key extraction are allowed). Hence, the probability that $\mathcal{B}$ does not abort during the simulation is: $Pr[\neg Event1]Pr[\neg Event2]Pr[\neg Event3] = Pr[\neg Event3] = 1/q_1$. This shows that $\mathcal{B}$'s advantage is at least $\varepsilon/q_1$ as required.

**Lemma 2.** *Let $H_2$ is a random oracle and there exists an IND-CPA adversary $\mathcal{A}$ against BasicPub-I which has non-negligible advantage $\varepsilon$. Suppose $\mathcal{A}$ makes a total of $q_2$ queries to $H_2$. Then there exists an algorithm $\mathcal{B}$ to solve the $(q_1 - 1)$-BCAA1 problem with non-negligible advantage $2\varepsilon/q_2$.*

**Proof:** Algorithm $\mathcal{B}$ is given as input a random $(q_1 - 1)$-BCAA1 instance $< q, G_1, G_2, G_T, e, \psi, P_1, P_2, xP_2, h_0, (h_1, \frac{1}{h_1+x}P_2), ...., (h_{q_1-1}, \frac{1}{h_{q_1-1}+x}P_2) >$ where $x \in Z_q^*$ is a random element. Algorithm $\mathcal{B}$ finds $D = e(P_1, P_2)^{1/(x+h_0)}$ by interacting with $\mathcal{A}$ as follows:

**Setup:** Algorithm $\mathcal{B}$ first simulates algorithm Key-generation of BasicPub-I to create the public parameters as belew.

1.Computes $P_{pub} = \psi(xP_2) \in G_1$.

2. Pick a random $r \in Z_q^*$ and set $P_{ID} = rP_2$.

2. Now $\mathcal{B}$ pasees $\mathcal{A}$ the public parameters $K_{pub-I} =< q, G_1, G_2, G_T, e, \psi, P_1, P_2, P_{pub}, r, P_{ID}, h_0, (h_1, \frac{1}{h_1+x}P_2), ...., (h_{q_1-1}, \frac{1}{h_{q_1-1}+x}P_2) >$. The private key is $K_{pri-I} = \frac{1}{h_0+x}P_2$.

$H_2$**-queries:**At any time algorithm $\mathcal{A}$ can query the random oracle $H_2$. To response to these queries $\mathcal{B}$ maintains a list of tuples $< X_i, H_i >$. We refer to this list as the $H_2^{list}$. When $\mathcal{A}$ queries the oracle $H_2$ at a point $X_i$ algorithm $\mathcal{B}$ responds as follows:

1. If the query $X_i$ already appears on the $H_2^{list}$ in a tuple $< X_i, H_i >$, then algorithm $\mathcal{B}$ responds with $H_2(X_i) = H_i$.

2. Otherwise, $\mathcal{B}$ chooses a random $H_i \in \{0,1\}^n$, return $H_2(X_i) = H_i$, and adds the tuple $< X_i, H_i >$ to the $H_2^{list}$.

**Challenge:** Algorithm $\mathcal{A}$ outputs two message $M_0$ and $M_1$ on which it wants to be challenged. $\mathcal{B}$ chooses a random string $R \in \{0,1\}^n$ and two random integers $r_1, r_2 \in Z_q^*$, and then defines the challenged ciphertext to be $C =< U, V >=< r_1P_1 + r_2P_{ID}, R >$. Observe that the decryption of $C$ is $R \oplus H_2(r_1P_1 + r_2rP_2, \frac{1}{h_0+x}P_2 - r^{-1}P_1) = R \oplus H_2(D^{r_1} * e(P_1, P_2)^c)$.

**Guess:** Algorithm $\mathcal{A}$ outputs it guess $b \in \{0,1\}$. At this point $\mathcal{B}$ pick a random tuple $< X_i, H_i >$ from the $H_2^{list}$ and outputs $(X_i/e(P_1, P_2)^c)^{-r_1}$ as the solution to the given instance of $(q_1 - 1)$-BCAA1 problem.

Let $\mathcal{H}$ be the event that algorithm $\mathcal{A}$ issues a query for $H_2(D^{r_1} * e(P_1, P_2)^c)$ at some point during the simulation. From Claim 1 of [2], we know that $Pr[\mathcal{H}] \geq 2\varepsilon$. Hence, at the end of the simulation, $D^{r_1} * e(P_1, P_2)^c$ appear in some tuple on the $H_2^{list}$ with probability at least $2\varepsilon$. Assume that $\mathcal{A}$ has queries $q_2$ distinct value to $H_2$. Hence, we know that $\mathcal{B}$ produces the correct answer with probability at least $2\varepsilon/q_2$.

To prove the theorem 3, we define the following public key encryption scheme called BasicPub-II.

**BasicPub-II** This scheme includes the following algorithms:

**Key-generation:** Given a security parameter $\kappa$, the generator takes the following steps.

1. Generate the parameters $< q, G_1, G_2, G_T, e, P_1, P_2, g >$ which are identical to the ones of the BasicCL-PKE.

2. Pick a random $s \in Z_q^*$ and compute $P_{pub} = sP_1$. Randomly choose element $h_0 \in Z_q^*$.

3. Pick a random $x \in Z_q^*$ and compute $P_{ID} = xP_2$.

4. Pick a hash function $H_2 : G_T \to \{0,1\}^n$ for some $n$.

The public $K_{pub-II} =< q, G_1, G_2, G_T, e, n, P_1, P_2, g, s, P_{pub}, P_{ID}, h_0, H_2 >$ and the private key is $K_{pri-II} = x$.

**Encrypt:**To encrypt $M \in \mathcal{M}$, perform the following steps:

1. Check that $P_{ID}$ is in $G_1^*$, if not output $\perp$. This checks the validity of the public key.

2. Choose two random $r_1, r_2 \in Z_q^*$ and compute the ciphertext:

$$C =< r_1(h_0 P_1 + P_{pub}) + r_2 P_{ID}, M \oplus H_2(g^{(r_1+r_2)})$$

**Decrypt:**Suppose $C =< U, V >$. To decrypt this ciphertext using the private key $K_{pri-II}$ compute:

$$M = V \oplus H_2(e(U, \frac{1}{h_0+s}P_2 - \frac{1}{K_{pri-II}}P_1)).$$

**Proof of Theorem 3** This theorem straightforwardly follows from the following lemma 3 and lemma 4.

**Lemma 3.** *Let $H_1$ is a random oracle. Suppose $\mathcal{A}_{\mathcal{II}}$ is an IND-CPA adversary that has advantage $\varepsilon$ against BasicCL-PKE. Suppose $\mathcal{A}_{\mathcal{II}}$ makes at most $q_1$ queries to $H_1$ and $q_E$ Private Key Extraction queries. Then there is an IND-CPA adversary $\mathcal{B}$ that at lest $\frac{\varepsilon}{q_1}(1 - \frac{1}{q_1})^{q_E}$ against BasicPub-II.*

Proof: Let $\mathcal{A}_{\mathcal{II}}$ be a Type II advesary against the new CL-PKE. Suppose $\mathcal{A}_{\mathcal{II}}$ has advantage $\varepsilon$ and makes $q_1$ queries to random oracle $H_1$. We show how to construct from $\mathcal{A}_{\mathcal{II}}$ an IND-CPA adversary $\mathcal{B}$ against the BasicPub-II. Let $\mathcal{C}$ denote the challenger against $\mathcal{B}$ for BasicPub-II. The challenger $\mathcal{C}$ begins by supplying $\mathcal{B}$ with a public key:

$$K_{pub-II} =< q, G_1, G_2, G_T, e, n, P_1, P_2, g, s, P_{pub}, P_{ID}, h_0, H_2 >.$$

Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}$ in an IND-CPA game as follows:

**Setup:** $\mathcal{B}$ chooses an index $I$ with $1 \leq I \leq q_{H_1}$. Then $\mathcal{B}$ gives $\mathcal{A}_{\mathcal{II}}$ the CL-PKE public parameters $params =< q, G_1, G_2, G_T, e, n, P_1, P_2, g, P_{pub}, H_1, H_2 >$ and the value of $s$. Here $< q, G_1, G_2, G_T, e, n, P_1, P_2, P_{pub}, g, H_2 >$ and $s$ are taken from $K_{pub-II}$. $H_1$ is a random oracle controlled by $\mathcal{B}$ as described below.

$H_1$**-queries:** At any time algorithm $\mathcal{A}_{\mathcal{II}}$ can query the random oracle $H_1$. To response to these queries $\mathcal{B}$ maintains a list of tuples $< ID_i, h_i, x_i >$. We refer to this list as the $H_1^{list}$. When $\mathcal{A}_{\mathcal{II}}$ queries the oracle $H_1$ at a point $ID_i$ algorithm $\mathcal{B}$ responds as follows:

1. If the query $ID_i$ already appears on the $H_1^{list}$ in a tuple $< ID_i, h_i, x_i >$, then algorithm $\mathcal{B}$ responds with $H_1(ID_i) = h_i$.

2. Otherwise, if the query is on the $Ith$ distinct $ID$, then $\mathcal{B}$ return $H_1(ID_I) = h_0$ and add $< ID_{ID}, h_0, \perp>$ into the $H_1^{list}$.

3. Otherwise, $\mathcal{B}$ chooses a random $h_i \in Z_q^*$ and return $H_1(ID_i) = h_i$. Then $\mathcal{B}$ chooses a random $x_i \in Z_q^*$ and adds the tuple $< ID_i, h_i, x_i >$ to the $H_1^{list}$.

**Phase 1:** Now $\mathcal{A}_{\mathcal{II}}$ launches Phase 1 of its attack by making a series of queries. B replies to these queries as follows:

**Private Key Extraction:** If the query is on $ID_I$ then $\mathcal{B}$ aborts **(Event 1)**. Otherwise, if the request in on $ID_i$ with $i \neq I$, then $\mathcal{B}$ outputs $< \frac{1}{h_i+s}P_2, x_i >$.

**Request for Public Key:** If the query is on $ID_I$ then $\mathcal{B}$ returns $P_{ID}$. Otherwise, if the request is on $ID_i$ with $i \neq I$, then $\mathcal{B}$ outputs $x_i P_2$.

**Challenge Phase:** At some point, $\mathcal{A_{II}}$ decides to end Phase 1 and picks $ID_{ch}$ and two messages $M_0, M_1$ on which it wants to be challenged. $\mathcal{B}$ responds as follows.

1. If $ID_{ch} \neq ID_I$ then $\mathcal{B}$ aborts **(Event 2)**.

2. Otherwise, $ID_{ch} = ID_I$ and $\mathcal{B}$ gives $\mathcal{C}$ the pair $M_0, M_1$ as the messages on which it wishes to be challenged. $\mathcal{C}$ responds with the challenge ciphertext $C =< U, V, W >$ which is the BasicPub-II encryption of $M_b$ for a random $b \in \{0,1\}$. Then $\mathcal{B}$ delivers $C$ to $\mathcal{A_{II}}$. It is easy to see that $C$ is the CL-PKE encrypton of $M_b$ for identifier $ID_I$ with public key $P_{ID}$.

**Phase 2:** $\mathcal{B}$ responds to queries as in Phase 1.

**Guess:** Eventually, $\mathcal{A_{II}}$ outputs $b'$ for $b$. Algorithm $\mathcal{B}$ outputs $b'$ as its guess for $b$.

If the algorithm $\mathcal{B}$ does not abort during the simulation then algorithm $\mathcal{A_{II}}$'s view is identical to its view in the real attack. To complete the proof it remains to calculate the probability that $\mathcal{B}$ does not abort during the simulation. $\mathcal{B}$ could abort when one of the following events happens:(1)Event 1:$\mathcal{A}$ queries the Private key Extraction for $ID_I$ at some point; (2) Event 2: $\mathcal{A}$ does not choose $ID_I$ as $ID_{ch}$. Hence, the probability that $\mathcal{B}$ does not abort during the simulation is: $Pr[\neg Event1]Pr[\neg Event2] = (1 - \frac{1}{q_1})^{q_E}(\frac{1}{q_1})$. This shows that $\mathcal{B}$'s advantage is at least $\frac{\varepsilon}{q_1}(1 - \frac{1}{q_1})^{q_E}$ as required.

**Lemma 4.** *Let $H_2$ is a random oracle and there exists an IND-CPA adversary $\mathcal{A}$ against BasicPub-II which has non-negligible advantage $\varepsilon$. Suppose $\mathcal{A}$ makes a total of $q_2$ queries to $H_2$. Then there exists an algorithm $\mathcal{B}$ to solve the co-BIDH problem with non-negligible advantage $2\varepsilon/q_2$.*

Proof: Algorithm $\mathcal{B}$ is given as input a random co-BIDH problem instance $< P_1, P_2, aP_2, bP_2 >$. Let $D = e(P_1, P_2)^{a^{-1}b}$ be the solution to the co-BIDH problem. Algorithm $\mathcal{B}$ finds $D$ by interacting with $\mathcal{A}$ as follows:

**Setup:** Algorithm $\mathcal{B}$ simulate algorithm Key-generation of the BasicPub-II to create the public $K_{pub-II} =< q, G_1, G_2, G_T, e, n, P_1, P_2, g, s, P_{pub}, P_{ID}, h_0, H_2 >$ by randomly selecting $s, h_0 \in Z_q^*$ and setting $P_{pub} = sP, P_{ID} = aP_2$. $H_2$ is a random oracle controlled by $\mathcal{B}$. The private key $K_{pri-II}$ equals to $a$ which $\mathcal{B}$ does not know. Then algorithm $\mathcal{B}$ passes the public key $K_{pub-II}$ to $\mathcal{A}$ and responds queries as follows.

$H_2$**-queries:** At any time algorithm $\mathcal{A}$ can query the random oracle $H_2$. To response to these queries $\mathcal{B}$ maintains a list of tuples $< X_i, H_i >$. We refer to this list as the $H_2^{list}$. When $\mathcal{A}$ queries the oracle $H_2$ at a point $X_i$ algorithm $\mathcal{B}$ responds as follows:

1. If the query $X_i$ already appears on the $H_2^{list}$ in a tuple $< X_i, H_i >$, then algorithm $\mathcal{B}$ responds with $H_2(X_i) = H_i$.

2. Otherwise, $\mathcal{B}$ chooses a random $H_i \in \{0,1\}^n$, return $H_2(X_i) = H_i$, and adds the tuple $< X_i, H_i >$ to the $H_2^{list}$.

**Challenge:** Algorithm $\mathcal{A}$ outputs two message $M_0$ and $M_1$ on which it wants to be challenged. $\mathcal{B}$ chooses a random string $R \in \{0,1\}^n$ and a random integer $c \in Z_q^*$, and then defines the challenged ciphertext to be $C =< U, V >=< (h_0 + s)cP_1 + bP_2, R >$. Observe that the decryption of $C$ is $R \oplus H_2(e((h_0 + s)cP_1 + bP_2, \frac{1}{h_0+s}P_2 - a^{-1}P_1) = R \oplus H_2(D * e(P_1, P_2)^c)$.

**Guess:** Algorithm $\mathcal{A}$ outputs it guess $b \in \{0,1\}$. At this point $\mathcal{B}$ pick a random tuple $< X_i, H_i >$ from the $H_2^{list}$ and outputs $X_i/e(P_1, P_2)^c$ as the solution to the given instance of co-BIDH problem.

Let $\mathcal{H}$ be the event that algorithm $\mathcal{A}$ issues a query for $H_2(D * e(P_1, P_2)^c)$ at some point during the simulation. From Claim 1 of [2], we know that $Pr[\mathcal{H}] \geq 2\varepsilon$. Hence, at the end of the simulation, $D * e(P_1, P_2)^c$ appear in some tuple on the $H_2^{list}$ with probability at least $2\varepsilon$. Assume that $\mathcal{A}$ has queries $q_2$ distinct value to $H_2$. Hence, we know that $\mathcal{B}$ produces the correct answer with probability at least $2\varepsilon/q_2$.

### 4.3 FullCL-PKE

In this section, we use a technique due to Fujisaki-Okamoto [10] to convert the IND-CPA secure BasicCL-PKE scheme into an IND-CCA secure certificateless encryption scheme in the random oracle. We obtain the following IND-CCA certificateless encryption called FullCL-PKE by applying the Fujisaki-Okamoto transformation.

**Setup:** As in the BasicCL-PKE scheme. In addition, we select two hash functions $H_3 : \{0,1\}^n \times \{0,1\}^n \rightarrow Z_q^*, H_5 : \{0,1\}^n \times \{0,1\}^n \rightarrow Z_q^*, H_4 : \{0,1\}^n \rightarrow \{0,1\}^n$.

**Partial-Private-Key-Extract:** As in the BasicCL-PKE scheme.

**Set-Secret-Value:** As in the BasicCL-PKE scheme.

**Set-Private-Key:** As in the BasicCL-PKE scheme.

**Set-Public-Key:** As in the BasicCL-PKE scheme.

**Encrypt:** To encrypt $M \in \mathcal{M}$ for entity $ID$ with the public key $P_{ID}$, perform the following steps:

1. Check that $P_{ID}$ is in $G_2^*$, if not output $\perp$. This checks the validity of the public key.
2. Compute $Q_{ID} = H_1(ID)P_1 + P_{pub}$.
3. Choose a random $\sigma \in \{0,1\}$ and set $r_1 = H_3(\sigma, M), r_2 = H_5(\sigma, M)$.
4. Choose random values $r_1$ and $r_2$ and compute the ciphertext:

$$C =< r_1 Q_{ID} + r_2 P_{ID}, \sigma \oplus H_2(g^{(r_1+r_2)}), M \oplus H_4(\sigma) >$$

**Decrypt:** Suppose $C =< U, V, W >$. To decrypt this ciphertext using the private key $S_{ID} =< D_{ID}, x_{ID} >$ compute:

1. Compute $V \oplus H_2(e(U, D_{ID} - \frac{1}{x_{ID}}P_1)) = \sigma$.
2. Compute $W \oplus H_4(\sigma) = M$.
3. Set $r_1 = H_3(\sigma, M), r_2 = H_5(\sigma, M)$. Test that $U = r_1 Q_{ID} + r_2 P_{ID}$. If not, reject the ciphtext.

4. Output $M$ as the decryption of $C$.

The security proof of the FullCL-PKE scheme is based on the results of Fujisaki and Okamoto (Theorem 14 in [10]). We can prove the FullCL-PKE to be IND-CCA secure formally in the similar method in [2]. The detail proof will be written in the full paper version.

## 5 Performance Analysis

In this section, we will show that our proposed FullCL-PKE scheme has the best performance, comparing with other existing CL-PKE schemes [3,4,5,6]. All the schemes have three major operations, i.e., Pairing (p), Scalar(s) and Exponentiation (e). Without considering the pre-computation, the properties and performance of the CL-PKE schemes are listed in Table 1, where we compare the schemes on the computation complexity, public key length (Pubkey Len) and the hardness assumption.

We know that pairing computation is more time-consuming than scalar and exponentiation computation [14]. From Table 1 we can see that our new scheme requires no pairing computation in Encrypt and the public key consists of only one element of $G_2$ rather than two required in AP's scheme I and CC's scheme I. Hence, our scheme is more efficient than the existing CL-PKE schemes.

**Table 1. Comparison of the CL-PKE Schemes**

| Schemes | Encrypt | Decrypt | Pubkey Len |
|---|---|---|---|
| AP's scheme I [3] | 3p+1s+1e | 1p+1s | 2 |
| CC's scheme I [5] | 3p+1s+1e | 1p+1s | 2 |
| AP's scheme II [4] | 1p+2s+1e | 1p+2s | 1 |
| CC's scheme II [6] | 1p+2s+1e | 1p+2s | 1 |
| New scheme | 3s+1e | 1p+2s | 1 |

## 6 Conclusions

In this paper, we present an efficient CL-PKE scheme which is constructed by a kind of parallel model rather than a serial model. Based on the security of two public key encryption schemes we prove the security of our scheme formally in the similar method in [2]. Furthermore, our scheme is more efficient than the existing CL-PKE schemes on computation or published public key information.

## References


1. Shamir, A.,: Identity based Cryptosystems and Signature Schemes. Advances in Cryptology-CRYPTO'84, Springer-verlag, LNCS 196 (1985) 47–53
2. Boneh, D. and Franklin, M.,: Identity based Encryption from the Weil Pairing. Advances in Cryptology-CRYPTO'2001, Springer-Verlg, LNCS 2139 (2001) 213–229
3. Al-Riyami, S.S. and Paterson, K.G.,: Certificateless Public Key Cryptography. In Advances in Cryptology C ASIACRYPT 2003, Springer-verlag LNCS vol. 2894 (2003) 452-C473
4. Al-Riyami, S.S. and Paterson, K.G.,: CBE from CL-PKE: A Generic Construciton and Efficient Schemes. PKC 2005, LNCS 3386 (2005) 398–415
5. Cheng, Z.H., Comley, R. and Vasiu, L.,: Remove Key Escrow from The Identity-Based Encryption System. TCS@IFIP, Toulouse, France, August 2004. Foundations of Information Technology in the Era of Network and Mobile Computing.
6. Cheng, Z.H. and Comley, R.,: Efficient Certificateless Public Key Encryption. Cryptology ePrint Archive, Report 2005/012
7. Sakai, R. and Kasahara, M.,: ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054
8. Chen, L.Q. and Cheng, Z.H.,: Security Proof of Sakai-Kasahara's Identity-Based Encryption Scheme. Cryptology ePrint Archive, Report 2005/226
9. Yum, D.H., Lee, P.J.,: Generic Construction of Certificateless Encryption. ACISP 2004. 200–211.
10. Fujisaki, E. and Okamotom, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. Advances in Cryptology - CRYPTO 1999 Proceedings, Springer-Verlag (1999) 535–554
11. Smart, N. and Vercauteren, F.,: On computable isomorphisms in efficient pairing based systems. Cryptology ePrint Archive, Report 2005/116
12. Boneh, D. and Boyen, X.,: efficient selective-ID secure identity-based encryption without random oracles. In Proceedings of Advances in Cryptology - Eurocrypt 2004, LNCS 3027, Springer-Verlag (2004) 223–238
13. Sakai, R., Kasahara, M.,: ID based Cryptosystems with Pairing on Elliptic Curve. Cryptology ePrint Archive, Report 2003/054
14. Barreto, P.S.L.M., Lynn, H.Y. and Scott, M.,: Efficient Algorithms for Pairing-based cryptosystems. Advances in Cryptology-Crypto 2002, LNCS Vol. 2442 (2002) 354–368