2009

An Efficient Certificateless Encryption Scheme in the Standard Model

Hua Guo
Beihang University

Xiyong Zhang
Zhengzhou Information Science and Technology Institute

Yi Mu
University of Wollongong, ymu@uow.edu.au

Zhoujun Li
Beihang University

An Efficient Certificateless Encryption Scheme in the Standard Model

Abstract

We propose an efficient certificateless public key encryption (CL-PKE) scheme which is provably secure against chosen ciphertext attacks without random oracles. Our scheme is more computationally efficient than the existing schemes and provides the shortest public key length compared to other existing CL-PKEs with random oracles. We also propose a practical self-generated-certificate encryption (SGC-PKE) scheme based on our CL-PKE scheme. One of merits of such cryptographic systems is that it can be applied to countermeasure "Denial-of-Decryption (DoD) Attacks" that is inherent in CL-PKE.

Disciplines

Physical Sciences and Mathematics

Publication Details

Guo, H., Zhang, X., Mu, Y. & Li, Z. (2009). An Efficient Certificateless Encryption Scheme in the Standard Model. In Y. Xiang, J. Lopez, H. Wang & W. Zhou (Eds.), 2009 Third International Conference on Network and System Security (pp. 302-309). Gold Coast, Australia: IEEE.

2009 Third International Conference on Network and System Security

An Efficient Certificateless Encryption Scheme in the Standard Model

Hua Guo*, Xiyong Zhang†, Yi Mu‡, Zhoujun Li*
* *School of Computer Science & Engineering, Beihang University, Beijing, PRC*
† *Zhengzhou Information Science and Technology Institute, Zhengzhou, PRC*
‡ *CCISR, School of Computer Science Software Engineering, University of Wollongong, NSW, Australia*

***Abstract*—We propose an efficient Certificateless Public Key Encryption (CL-PKE) scheme which is provably secure against chosen ciphertext attacks without random oracles. Our scheme is more computationally efficient than the existing schemes and provides the shortest public key length compared to other existing CL-PKEs with random oracles. We also propose a practical Self-Generated-Certificate Encryption (SGC-PKE) scheme based on our CL-PKE scheme. One of merits of such cryptographic systems is that it can be applied to countermeasure "Denial-of-Decryption (DoD) Attacks" that is inherent in CL-PKE.**

***Keywords*-Certificateless Encryption; Self-Generated-Certificate Encryption; Standard Model.**

I. Introduction

In traditional public key cryptography (PKC), cryptographic keys are generated randomly with no connection to user identity. Therefore, it suffers from the so-called man-in-the-middle attack. This problem can be solved by introducing public key certificates where a public key is certified by a trusted certification authority (CA). A public key certificate consists of user identity, public key, time stamp, and other information, which are digitally signed by CA. There are some concerns about the deployment of public key certification, due to issues in certificate revocation, storage and distribution.

Identity-based cryptography (IBC) was introduced by Shamir.IBC solves the inherent problem of key authenticity in a different way. In an identity-based system, users can choose an arbitrary string, such as email address and IP number, as their public key. The corresponding private key is created by binding the identity string with a master secret of a trusted authority called Key Generation Centre (KGC). In this way, the certificate is provided implicitly and it is no longer necessary to explicitly authenticate public keys. The disadvantage of such system is that KGC knows every user's private key, which gives PKG the great power to impersonate any user. This problem is referred to as the key-escrow problem.

In order to eliminate the key-escrow problem of identity-based cryptography, certificateless public key cryptography (CL-PKC) was proposed by Al-Riyami and Paterson [1] in 2003. Unlike ID-based systems, a user's private key consists of two parts: **partial private key** corresponding to the ID is generated by KGC, while the other part is generated by user itself which is unknown to others. The user also selects a public key associated to its private key (the second part).

CL-PKE is a new paradigm which binds identity-based cryptography and traditional public key cryptography. It preserves the attractive advantage of ID-based cryptography without requiring digital certificates, but it is no longer "ID-based" since public keys are no longer arbitrary strings. The public key in CL-PKC does not need to be explicitly certified by a trusted party as it has been generated with some "partial secret key" from the KGC.

Unfortunately, due to lack of authentication, the public key associated with the private key of a user in CL-PKC may be replaced by anyone. Consequently, the receiver is not able to decrypt ciphertext properly. Liu and Au [1] called this attack Denial-of-Decryption (DoD) Attack. In order to resist this attack, Liu and Au [1] further proposed a new paradigm called Self-Generated-Certificate Public Key Cryptography (SGC-PKC) based on CL-PKE system. But, being different from CL-PKE cryptography, every user needs to generate a certificate for his public key using his own secret key. This signature binds the identity and the public key together. It is implicitly included in the user's public key and can be verified by using the user's identity and public key only without the help of a third trusted party. By this means, one can check if a victim's public key is replaced through a verification of the signature. Thus this cryptography is immune to the DoD attack while inheriting all appealing advantages of CL-PKE cryptography.

Since the first construction of CL-PKE [1] by Al-Riyami and Paterson, many efficient CL-PKE schemes using bilinear pairings have been proposed [1], [2], [6], [14], [19], including several generic constructions [5], [11], [12] and CL-PKE schemes without using a pairing [4], [15]. Several good security analyses are also found in the literature [3], [13], [20]. Most of these schemes were proven secure against chosen ciphertext attacks in the random oracle model. We observe the following three schemes that do not resort to random oracles. Yum and Lee [20] provided a generic construction provably secure in the standard model from general cryptographic primitives such including PKE and IBE. However, Hu et al.[13] showed that the Yum-Lee construction has security weakness and proposed a fix in the standard model. Liu et al. [17] first proposed a concrete certificateless encryption scheme provably secure without

978-0-7695-3838-9/09 $26.00 © 2009 IEEE
DOI 10.1109/NSS.2009.31

random oracles. Later Dent, Libert, and Paterson improved this scheme and achieved stronger security. Park et al. [18] proposed a CL-PKE scheme which is provably secure in the selective-ID security model against chosen ciphertext attacks without random oracles. Park et al.'s scheme makes use of Gentry identity-based encryption scheme. However, these three schemes suffer from the expensive computation cost.

In this paper, we propose a new CL-encryption scheme which is provably secure in the standard model. Compared with other schemes without random oracles, our scheme is superior in terms of computation cost and the size of public key. Our scheme is inspired from Gentry's ID-PKE scheme [10]. We also give an SGC-PKE scheme based our CL-PKE scheme to resist the DoD attack. Our scheme is more efficient than that in [1] in terms of computation cost and public key length.

The rest of this paper is organized as follows. In Section 2, we describe the preliminaries including Weil pairing, review the formulation of CL-PKE, and give the security model for CL-PKE. In Section 3, we present our CL-PKE scheme and provide a detailed security proof to show that our scheme is semantically secure. Then, we remark the CL-PKE scheme on complexity and security. In Section 4, we extend the scheme to the SGC-PKE scheme. In section 5, we conclude the paper.

II. Preliminaries

In this section, we review some basic concepts, including the pairing primitives, assumptions, and the formulation of CL-PKE and its security model.

A. Bilinear Pairing and Security Assumptions

Here we briefly review some basic definitions of pairings.

Definition 1: Let $\mathbb{G}$ be an additive group of prime order p and $\mathbb{G}_T$ a multiplicative group of the same order. Let P denote a generator of $\mathbb{G}$. An admissible pairing is a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ which has the following properties:

1. Bilinear: For all Q, $R \in \mathbb{G}$ and $a, b \in Z_p^*$, we have $\hat{e}(aQ, bR) = \hat{e}(Q, R)^{ab}$.
2. Non-degenerate: $\hat{e}(P, P) \neq 1_{\mathbb{G}_T}$.
3. Computable: $\hat{e}$ is efficiently computable.

In the following, we describe two assumptions which are related to the security of our schemes.

Truncated Decision q-ABDHE Assumption [10]: For $P, Q \in \mathbb{G}$, where $\mathbb{G}$ is a cyclic group of prime order p, $\alpha \in \mathbb{Z}_p$, given a vector of $q+3$ elements $(Q, \alpha^{q+2}Q, P, \alpha P, \alpha^2 P, \cdots, \alpha^q P)$ as input, an algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the truncated decision q-ABDHE if

$$\big|\Pr[\mathcal{B}(Q, \alpha^{q+2}Q, P, \alpha P, \alpha^2 P, \cdots, \alpha^q P, \hat{e}(\alpha^{q+1}P, Q)) = 0] - \Pr[\mathcal{B}(Q, \alpha^{q+2}Q, P, \alpha P, \alpha^2 P, \cdots, \alpha^q P, Z) = 0]\big| \geq \epsilon$$

where $\hat{e}$ is a bilinear pairing from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_T$, the probability is over the random choice of generators $P, Q \in \mathbb{G}$, the random choice of $\alpha \in \mathbb{Z}_p$, the random choice of $Z \in \mathbb{G}_T$, and the random bits consumed by $\mathcal{B}$.

The (ϵ, t, q)-ADBHE assumption holds in a group $\mathbb{G}$ if no algorithm running in time at most t can solve the (ϵ, t, q)-ADBHE problem in $\mathbb{G}$ with advantage at least ϵ.

Decision 1-BDHI Assumption: For $P \in \mathbb{G}$, where $\mathbb{G}$ is a cyclic group of prime order p, and the given element αP as input, an algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the truncated decision 1-BDHI if

$$|\Pr[\mathcal{B}(P, \alpha P, \hat{e}(P, P)^{1/\alpha}) = 0] - \Pr[\mathcal{B}(P, \alpha P, Z) = 0]| \geq \epsilon$$

where $\hat{e}$ is a bilinear pairing from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_T$, the probability is over the random choice of generator $P \in \mathbb{G}$, the random choice of $\alpha \in \mathbb{Z}_p$, the random choice of $Z \in \mathbb{G}_T$, and the random bits consumed by $\mathcal{B}$.

We say that the $(\epsilon, t, 1)$-BDHI assumption holds in a group $\mathbb{G}$ if no algorithm running in time at most t can solve that $(\epsilon, t, 1)$-BDHI problem in $\mathbb{G}$ with advantage at least ϵ.

B. Formulation of CL-PKE and its Security Model

In this section, we review the definition and security model for CL-PKE from [1].

Definition 2: A CL-PKE scheme is specified by seven algorithms (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 k as input and returns the system parameters params and the master-key. 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, master-key and an identifier for entity A, ID_A as input. It returns a partial private key ppk_A.
- **Set-Secret-Value** is a probabilistic algorithm that takes as input params and outputs a secret value sk_A.
- **Set-Private-Key** is a deterministic algorithm that takes as input params, ppk_A and sk_A and returns S_A, a (full) private key.
- **Set-Public-Key** is a deterministic algorithm that takes params and S_A as input and outputs a public key pk_A.
- **Encrypt** is a probabilistic algorithm that takes params, $M \in \mathcal{M}$, sk_A and ID_A as input 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 input 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**. These three algorithms will be called **User-Key-Generation** for simplicity in this paper. Usually, A is the only entity in possession S_A and pk_A. **Setup** and **Partial-Private-Key-Extract** are usually run by a trusted third party, called Key Generation Center (KGC) [1].

In the following, we focus on the security model which can be used to prove the semantic security of CL-PKE schemes. Al-Riyami and Patersion presented the full IND-CCA security model for CL-PKE in [1]. In the following, we describe the actions that a general adversary $\mathcal{A}$ against a CL-PKE scheme may carry out and how each action should be handled by the challenger $\mathcal{C}$.

- **Extract partial private key of entity A:** Challenger $\mathcal{C}$ responds by running **Partial-Private-Key-Extract** to generate the partial private key D_A for entity A.
- **Extract private key for entity A:** If A's public key has not been replaced, $\mathcal{C}$ responds by running algorithm **Set-Private-Key** to generate the private key S_A for entity A. But it is unreasonable to expect $\mathcal{C}$ to be able to respond to such a query if A's public key has already been replaced by $\mathcal{A}$.
- **Request public key of entity A:** $\mathcal{C}$ 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).
- **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'_A of its choice. The current value of an entity's public key is used by $\mathcal{C}$ in any computations or responses to the adversary's requests.
- **Decryption query for ciphertext C and entity A:** On input a ciphertext and an identity, returns the decrypted plaintext using the secret key corresponding to the current value of the public key associated with the identity of the user. If the user's public key has been replaced, it requires an additional input of the corresponding secret key for the decryption. If it is not given this secret key, it outputs $\perp$.

The IND-CCA security model of [1] 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 master-key. A Type II adversary $\mathcal{A}_{II}$ is equipped with the master-key but is not allowed to replace public keys of entities. This adversary models security against an eavesdropping KGC.

Weak CL-PKE Type I IND-CCA Adversary: Such an adversary $\mathcal{A}_I$ does not have access to the master-key. 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. $\mathcal{A}_I$ cannot make a decryption query on the ciphertext C for the combination (ID_A, pk_A) such that pk_A has already been replaced.
5. In Phase 2, $\mathcal{A}_I$ cannot make a decryption query on the challenge ciphertext C for the combination (ID_{ch}, pk_{ch}) that was used to encrypt M_b.

Strong CL-PKE Type II IND-CCA Adversary:([1], [8]) Such an adversary $\mathcal{A}_{II}$ has access to the master-key, and can replace public keys of entities. $\mathcal{A}_{II}$ can compute partial private keys for himself, given the master-key. 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 extract the private key of any identity for which it has replaced the public key.
2. Adversary $\mathcal{A}_{II}$ cannot extract the private key for ID_{ch} at any point.
3. Adversary $\mathcal{A}_{II}$ cannot extract the partial private key at any point.
4. In Phase 2, $\mathcal{A}_{II}$ cannot make a decryption query on the challenge ciphertext C for the combination (ID_{ch}, pk_{ch}) that was used to encrypt M_b.

Definition 3: A CL-PKE scheme is said to be IND-CCA secure if no polynomially bounded adversary $\mathcal{A}$ of Weak Type I or Strong Type II has a non-negligible advantage in the following game:

Setup: The challenger $\mathcal{C}$ takes a security parameter as input and runs the Setup algorithm. It gives $\mathcal{A}$ the resulting system parameters params. If $\mathcal{A}$ is of Type I, then $\mathcal{C}$ keeps the master-key to himself, otherwise, he gives the master-key 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. $\mathcal{C}$ now picks a random bit $b \in \{0, 1\}$ and computes C^*, the encryption of M_b under the current public key pk_{ch} for ID_{ch}, then delivery C^* to $\mathcal{A}$.

Phase 2: Now, $\mathcal{A}$ issues a second sequence of requests as in Phase 1, which is 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 $Adv(\mathcal{A}) := |\ \Pr[b = b'] - \frac{1}{2}\ |$.

Remark 1: If Type I Adversary can output a valid plaintext without any additional input even in the case that the corresponding public key has been replaced, we call it Strong Type I Adversary and the CL-PKE scheme to be IND-CCA secure of Strong Type I. Note that the only difference between a Weak Type I and Strong Type I Adversary is on the **Decryption-Oracle**.

III. Our CL-PKE Scheme in the Standard Model

In this section, we give a new CL-PKE scheme.

A. The Scheme

As all other identity-based systems, we assume the existence of a trusted Key Generation Center (KGC) that is responsible for the creation and secure distribution of users' partial private keys.

Setup: This algorithm takes a security parameter k as its input and conducts the following steps:

1. Generate a prime p, and a bilinear pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are all cyclic groups of order p. Then, choose random generators $P, Q_1, Q_2, Q_3 \in \mathbb{G}$.
2. Choose a value $\alpha \in \mathbb{Z}_p^*$ and compute $P_1 = \alpha P$.
3. Choose one cryptographic hash functions $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$.

The KGC publishes $params = \langle p, \mathbb{G}, \mathbb{G}_T, \hat{e}, P, P_1, Q_1, Q_2, Q_3, H\rangle$ as the system parameters, and keeps α as his own secret master key. The parameters are distributed to the users of the system through a secure authenticated channel.

Partial-Secret-Key (PSK) Extract: The KGC takes as input $params$, master-key, and an arbitrary $ID \in \mathbb{Z}_p^*$, generates random $r_{ID,i} \in \mathbb{Z}_p^*$, and outputs the PSK $\{psk_{ID,i} = (r_{ID,i}, h_{ID,i}), i = \{1,2,3\}\}$, where $h_{ID,i} = \frac{1}{\alpha - ID}(Q_i - r_{ID,i}P)$.

Set-Secret-Value: User selects a secret $r \in \mathbb{Z}_p^*$ as his secret key $sk_{ID} = r$.

Set-Private-Key: The user sets his full private key as $S_{ID} = \{sk_{ID}, psk_{ID,1}, psk_{ID,2}, psk_{ID,3}\} = \{r, (r_{ID,i}, h_{ID,i})(i = 1,2,3)\}$.

Set-Public-Key: The user sets his public key as $pk_{ID} = r(\alpha - ID)P = r(P_1 - ID \cdot P)$.

Encrypt: To encrypt $m \in \mathbb{G}_T$ using the user's public key $pk_{ID} = r(\alpha - ID)P$, the sender generates random $s \in \mathbb{Z}_p^*$ and sends the ciphertext

$$C = (s \cdot pk_{ID}, \hat{e}(P,P)^s, m \cdot \hat{e}(P,Q_1)^{-s}, \hat{e}(P,Q_2)^s \hat{e}(P,Q_3)^{s\beta}).$$

where $\beta = H(u,v,w)$. Denote C as $C = (u,v,w,y)$.

Note that $\hat{e}(P,P), \hat{e}(P,Q_1), \hat{e}(P,Q_2), \hat{e}(P,Q_3)$ can be precomputed so that encryption algorithm requires no pairing computations.

Decrypt: To decrypt ciphertext $C = (u,v,w,y)$ for ID, the recipient computes $\beta = H(u,v,w)$ and checks whether

$$y = \hat{e}(u, h_{ID,2} + \beta h_{ID,3})^{1/sk_{ID}} \cdot v^{r_{ID,2} + \beta r_{ID,3}}.$$

If it fails, the recipient outputs $\perp$. Otherwise outputs $m = w \cdot \hat{e}(u, h_{ID,1})^{1/sk_{ID}} \cdot v^{r_{ID,1}}$.

Correctness: we can easily verify the correctness of the checking equation:

$$\begin{aligned}
&\hat{e}(u, h_{ID,2} + \beta h_{ID,3})^{1/sk_{ID}} \cdot v^{r_{ID,2} + r_{ID,3}\beta} \\
&= \hat{e}(sk_{ID} \cdot s(\alpha - ID)P, \tfrac{1}{\alpha - ID}(Q_2 + \beta Q_3 - (r_{ID,2} + \\
&+ r_{ID,3}\beta)P))^{1/sk_{ID}} \cdot \hat{e}(P,P)^{s(r_{ID,2} + r_{ID,3}\beta)} \\
&= \hat{e}(s(\alpha - ID)P, \tfrac{1}{\alpha - ID}(Q_2 + \beta Q_3)) \\
&= \hat{e}(P,Q_2)^s \hat{e}(P,Q_3)^{s\beta} = y.
\end{aligned}$$

And the decryption correctness follows by:

$$\begin{aligned}
&w \cdot \hat{e}(u, h_{ID,1})^{1/sk_{ID}} \cdot v^{r_{ID,1}} \\
&= m \cdot \hat{e}(P,Q_1)^{-s} \cdot (\hat{e}(sk_{ID} \cdot s(\alpha - ID)P, \\
&\tfrac{1}{\alpha - ID}(Q_1 - r_{ID,1}P)))^{1/sk_{ID}} \cdot \hat{e}(P,P)^{s r_{ID,1}} \\
&= m \cdot \hat{e}(P,Q_1)^{-s} \cdot \hat{e}(P,Q_1)^s = m.
\end{aligned}$$

B. Efficiency Analysis and Comparison

We consider four major operations: Pairing, Scalar, Exponentiation and Hashing, of which pairing is the most expensive one. For simplicity, we denote these operations respectively by P (Pairing), S (scalar multiplication in $\mathbb{G}$), E (exponentiation in $\mathbb{G}_T$), and H (Hashing). Some CL-PKE schemes require message authentication or signature operations for the IND-CCA security. We denote by $(Mac, Verf)$ the message authentication algorithm and by (Sig, Ver) the signature algorithm.

We compare our scheme with some other schemes involving pairing computations, in terms of the security model, computation cost and public key length, without considering the pre-computation. The comparison is outlined in Table I where the second column and the third column respectively denote the Type I security model and Type II security model used in the corresponding CL-PKE scheme. We denote by Len the public key length in the corresponding schemes.

According to Table I, every scheme has some weakness, and it is unfair to claim that one scheme is better than another. However, we can analyze them as follows:

Compared to the first three schemes, our scheme is proven security in the standard model. Moveover, our scheme has less computational cost than the first two. A special hashing could be around 20 times more costly than a pairing [7], so our scheme is more efficient than Shi-Li's.

Compared to the next three schemes provably secure without random oracles, our scheme is the most efficient one due to its computation cost of encryption and decryption, which highlights the advantage of our scheme. Furthermore, They need to publish two elements as the user's public key, while our scheme needs only one. This would save the bandwidth in some situations.

Schemes	Type I	Type II	Model	Encrypt	Decrypt	Len
A-P [1]	Weak	Weak	ROM	3P+1S+1E+4H	1P+1S+3H	2
C-C [6]	Weak	Weak	ROM	1P+2S+1E+4H	1P+2S+3H	1
S-L [19]	Strong	Weak	ROM	3S+1E+5H	1P+2S+4H	1
P-C-H-L [18]	Strong	Strong	Standard	5S+2E+1Sig	6P+5S+2E+1Ver	2
L-A-S [17]	Weak	Strong	Standard	4P+4S+1E+1Mac	3P+1S+1E+1Verf	2
Dent [9]	Strong	Strong	Standard	3P+3S+1E+1H	4P+1H	2
Proposed	Weak	Strong	Standard	1S+4E+1H	2P+1S+4E+1H	1

Table I
THE COMPARISON OF THE PROPOSED SCHEME AND OTHER CL-PKE SCHEMES. THE "STRONG" SECURITY MODEL THAT DOES NOT HAVE THE "EXTRACT PRIVATE KEY" QUERY IS A LITTLE DIFFERENT FROM THE DENT'S SECURITY MODEL ([8], [9]).

IV. SECURITY ANALYSIS

In this section, we study the security of the proposed CL-PKE scheme. The following theorem shows that the CL-PKE scheme is a semantic IND-CCA secure certificateless encryption scheme in the standard model. Due to the limited space, here we only give a sketch proof.

Theorem 1: Let $q = q_{ID} + 2$. Assume the truncated decision $(t, \epsilon, q)-$ABDHE assumption holds for $(\mathbb{G}, \mathbb{G}_T, \hat{e})$. Then our CL-PKE scheme is $(t', \epsilon', q_{ID}, q_C)$ **IND-CCA** secure against weak Type I adversary with advantage at most ϵ', running in time at most t' and making at most q_{ID} **Partial Private Key Extraction** queries and at most q_C **Decryption** queries, where $t' = t - \mathcal{O}(t_{exp} \cdot q^2)$ and $\epsilon' \geq \epsilon$, t_{exp} is the time required to exponentiate in $\mathbb{G}$.

Proof: Assume there exists a type I adversary $\mathcal{A}$ that $(t', \epsilon', q_{ID}, q_C)$-breaks the **IND-CCA** secure for our scheme. We construct a PPT algorithm $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the truncated decision q-ABDHE problem with parameters $(Q, \alpha^{q+2}Q, P, \alpha P, \cdots, \alpha^q P, Z)$, where Z is either $\hat{e}(\alpha^{q+1}P, Q)$ or a random element of $\mathbb{G}_T$. The algorithm $\mathcal{B}$ proceeds as follows.

Setup: $\mathcal{B}$ generates three random polynomials $f_i(x) \in \mathbb{Z}_p[x]$ of degree q for $i \in \{1, 2, 3\}$. It sets $Q_i = f_i(\alpha)P$. It sends the public key (P, Q, Q_1, Q_2, Q_3) to $\mathcal{A}$. Since $P, \alpha, f_i(x)$ are chosen uniformly at random, Q_1, Q_2, Q_3 are uniformly random and the public key has a distribution identical to that in actual construction. Note that the master-key is α, which is unknown to $\mathcal{B}$.

Phase 1: In this phase $\mathcal{A}$ makes the following oracle queries.

Partial Private Key Extraction: Suppose the query is on ID. There are two cases: If $ID = \alpha$, $\mathcal{B}$ uses α to solve the truncated decision q-ABDHE; Else let $F_{ID,i}(x)$ be the (q-1)-degree polynomial $(f_i(x) - f_i(ID))/(x - ID)(i = 1, 2, 3)$. $\mathcal{B}$ answers the query with $(r_{ID,i}, h_{ID,i}) = (f_i(ID), F_{ID,i}(\alpha)P)$. The validity follows from the fact that

$$F_{ID,i}(\alpha)P = \frac{f_i(\alpha) - f_i(ID)}{\alpha - ID}P = \frac{Q_i - f_i(ID)}{\alpha - ID}P.$$

Private Key Extraction: $\mathcal{B}$ returns the $sk_{ID} = r$ for the given ID if the corresponding public key has not been replaced and $ID \neq ID^*$. Otherwise he outputs $\perp$.

Request for Public Key: $\mathcal{B}$ keeps the database of user key. Upon receiving a query for public key of ID, $\mathcal{B}$ looks up its database to find out the corresponding entry. If it does not exist, $\mathcal{B}$ runs **User-Key-Generation** to generate a secret and public key pair. It stores the key pair in its database and returns the public key as the query output.

User-Key-Generation: To generate a secret and public key pair. He stores the key pair in its database and returns the public key as the query output.

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

- If no tuple corresponding to ID exists on the database, $\mathcal{B}$ follows **User-Key-Generation** algorithm to create a new entry for this identity.
- Otherwise, $\mathcal{B}$ replaces the corresponding public key with PK_{ID}.

Decryption: According to the security model, the simulator only decrypts the ciphertext corresponding to an identity with unreplaced public key. So we assume that $\mathcal{B}$ knows the discrete logarithm of $pk = sk \cdot P = rP$. Under this assumption, $\mathcal{B}$ decrypts and answers the oracle by performing the usual *Decrypt* algorithm with the right partial private key and the private key. If the corresponding public key has been replaced, he outputs $\perp$.

Challenge Phase: $\mathcal{A}$ then outputs two messages M_0, M_1 and an identity ID^*. If $\alpha = ID^*$, $\mathcal{B}$ uses α to solve the truncated decision q-ABDHE immediately. Otherwise, $\mathcal{B}$ generates bit $b \in \{0, 1\}$, and uses **Partial Private Key Extraction** to compute a partial private key $\{r_{ID^*}, h_{ID^*}\}$ for ID^*. According to the security model, assume the secret key $sk_{ID^*} = r$ is known to $\mathcal{B}$.

Let $f(x) = x^{q+2}$ and $F(x_1, x_2) = (f(x_1) - f(x_2))/(x_1 - x_2) = x_1^{q+1} + F'(x_1, x_2)$, where $F'(x, y)$ is a polynomial of degree $q + 1$. $\mathcal{B}$ will compute the ciphertext as follows: $u = r \cdot (f(\alpha) - f(ID^*))Q, v = Z \cdot \hat{e}(Q, F'(\alpha, ID^*))$, $w = M_b/(\hat{e}(u, h_{ID^*})v^{r_{ID^*}})$, $y = \hat{e}(u, h_{ID^*,2} + \beta \cdot h_{ID^*,3}) \cdot v^{r_{ID^*,2} + \beta \cdot r_{ID^*,3}}$, where $beta = H(u, v, w)$.

Let $s = log_P Q \cdot F(\alpha, ID^*)$. If $Z = \hat{e}(\alpha^{q+1}P, Q)$, then $u = r \cdot s(\alpha - ID^*)P = s(\alpha - ID^*) \cdot rP, v = \hat{e}(\alpha^{q+1}P, Q) \cdot \hat{e}(Q, F'(\alpha, ID^*)P) = \hat{e}(Q, F(\alpha, ID^*)P) = \hat{e}(P, P)^s$, and $w = M_b/\hat{e}(u, h_{ID^*})v^{r_{ID^*}} = M_b/\hat{e}(P, Q_1)^s = M_b \cdot$

$\hat{e}(P,Q_1)^{-s}$, $y = \hat{e}(u, h_{ID^*,2}+\beta\cdot h_{ID^*,3})\cdot v^{r_{ID^*,2}+\beta\cdot r_{ID^*,3}} = \hat{e}(P, Q_2 + \beta Q_3)^s$. Here, s is random, so (u, v, w, y) is a valid and appropriately-distributed challenge ciphertext for $\mathcal{A}$.([10])

Phase 2: $\mathcal{B}$ repeats the same method it used in Phase 1.

Guess: Finally, the adversary outputs guess $b' \in \{0, 1\}$. If $b = b'$, $\mathcal{B}$ outputs 0 (indicating $Z = \hat{e}(\alpha^{q+1}P, Q)$); otherwise it outputs 1.

As the security proof in [10], the values $r_{ID,i}$ issued by $\mathcal{B}$ are appropriately distributed in $\mathcal{A}$'s view. For the probability analysis of $\mathcal{B}$'s advantage ϵ' in solving the truncated decision $q - ABDHE$ problem, we should consider in two cases.

When $Z = \hat{e}(\alpha^{q+1}P, Q)$, by using Z, $\mathcal{B}$ replies with a valid ciphertext with a distribution identical to that in the actual construction. So the simulator has $|\Pr[b = b'] - 1/2| \geq \epsilon$. When Z is random in $\mathbb{G}_T$, then the simulator has $\Pr[b = b'] = 1/2$.

Thus we have

$$\Pr[\mathcal{B}(Q, \alpha^{q+2}Q, P, \alpha P, \alpha^2 P, \cdots, \alpha^q P, \hat{e}(\alpha^{q+2}P, Q)) = 0] - \Pr[\mathcal{B}(Q, \alpha^{q+2}Q, P, \alpha P, \alpha^2 P, \cdots, \alpha^q P, Z) = 0]| = \epsilon' \geq \epsilon.$$

The time-complexity of the algorithm is identical to that of the proof in [10]. In the simulation, $\mathcal{B}$'s operation is dominated by computing $F_{ID,i}(\alpha)P$ in response to $\mathcal{A}$'s **Partial Private Key Extraction** query on ID, where $F_{ID,i}(x)$ is a polynomial of degree $q - 1$. Each such computation requires $\mathcal{O}(q)$ exponentiations in $\mathbb{G}$. So the time-complexity is $t' = t - \mathcal{O}(t_{exp} \cdot q^2)$. ∎

Theorem 2: Assume that the decision $(t, 1, \epsilon)$-BDHI assumption holds in $\mathbb{G}$. Then our CL-PKE scheme is (t, q, ϵ')-IND-CCA secure against the Type II adversary, with advantage at most ϵ', running in time at most t and making at most q **Request for Public Key** queries, where $\epsilon' \geq \epsilon/q$.

Proof: Assume there exists a type II adversary $\mathcal{A}$ against our scheme. We construct a PPT algorithm $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the decision 1-BDHI problem. $\mathcal{B}$ takes as input a decision 1-BDHI problem instance $(P, \alpha P, Z)$ and is to decide if $Z = \hat{e}(P, P)^{1/\alpha}$. In order to use $\mathcal{A}$ to solve the problem, $\mathcal{B}$ needs to simulate a challenger and the oracles for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

Setup: $\mathcal{B}$ generates two random elements $x_1, x_2, x_3 \in \mathbb{Z}_p^*$. Randomly chooses γ as the master key and chooses an identity ID^* and relates it to the public key $(\gamma\text{-}ID^*)\alpha P$. It sets $Q = \gamma P, Q_1 = x_1 P, Q_2 = x_2 P, Q_3 = x_3 P$ and sends the public key (P, Q, Q_1, Q_2, Q_3) and the master key γ to $\mathcal{A}$. Since P and γ are chosen uniformly at random, Q_1,Q_2 and Q_3 are uniformly random and the public key has a distribution identical to that in actual construction.

Phase 1: In this phase $\mathcal{A}$ can make the following oracle queries.

Request for Public Key: $\mathcal{B}$ keeps the database of user key. Upon receiving a query for public key of ID, $\mathcal{B}$ looks up its database to find out the corresponding entry. If it does not exists, $\mathcal{B}$ runs **User-Key-Generation** to generate a secret and public key pair. It stores the key pair in its database and returns the public key as the query output.

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

- If no tuple corresponding to ID exists on the database, $\mathcal{B}$ follows **User-Key-Generation** algorithm to create a new entry for this identity.
- Otherwise, $\mathcal{B}$ replaces the corresponding public key with PK_{ID}.

Extract private key for entity A**:** If the corresponding public key has not been replaced, $\mathcal{B}$ returns the $sk_{ID} = r$ for the given ID; Otherwise he outputs $\perp$.

User-Key-Generation: To generate a secret and public key pair, $\mathcal{B}$ stores the key pair in its database and returns the public key as the query output.

Decryption: $\mathcal{B}$ can decrypt the ciphertext (u, v, w, y) as follows:

Let $\beta = H(u, v, w)$. By using the trap values x_2, x_3, the simulator checks whether $y = v^{x_2+\beta x_3} (= \hat{e}(sP, Q_2 + \beta Q_3) = \hat{e}(P, Q_2)^s \hat{e}(P, Q_3)^{s\beta})$. If it fails, outputs $\perp$. Otherwise, we have $v^{x_1} = \hat{e}(P, Q_1)^s$. So the simulator outputs $m = w \cdot v^{x_1}$.

Challenge Phase: $\mathcal{A}$ outputs two messages M_0, M_1 and an identity ID_0. If $ID^* \neq ID_0$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ generates a bit $e \in \{0, 1\}$ and a random element $t \in \mathbb{Z}_p^*$, and constructs the ciphertext as follows.

$$u = (\gamma - ID_0) \cdot tP, v = Z^t, w = M_e / Z^{x_1 t}, y = Z^{(x_2+x_3\beta)t}$$

where $\beta = H(u, v, w)$.

In the above, if we let $s = t/\alpha$ and $Z = \hat{e}(P, P)^{1/\alpha}$, then $u = s \cdot pk_{ID^*}, v = \hat{e}(P, P)^s, w = M_e / \hat{e}(P, Q_1)^s, y = \hat{e}(P, Q_2)^s \cdot \hat{e}(P, Q_3)^{\beta s}$.

The above randomness of s comes from the randomness of t, so (u, v, w, y) is a valid and appropriately-distributed challenge ciphertext for $\mathcal{A}$ ([10]).

Phase 2: $\mathcal{B}$ repeats the same method it used in Phase 1.

Guess: Finally, the adversary outputs guess $e' \in \{0, 1\}$. If $e = e'$, $\mathcal{B}$ outputs 0 (indicating $Z = \hat{e}(P, P)^{1/\alpha}$); otherwise it outputs 1.

The key pairs and challenge cipher-text issued by $\mathcal{B}$ have a distribution identical to that in the actual construction. If $ID_0 \neq ID^*$, the simulation aborts the game. This happens with probability $1/q$ since $\mathcal{A}$ make q different **Request for Public Key** queries. If the algorithm does not abort during the simulation then $\mathcal{A}$'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.

For the probability analysis of $\mathcal{B}$'s advantage ϵ' in solving the decision 1-BDHI problem, we should consider in two cases.

When $Z = \hat{e}(P, P)^{1/\alpha}$, by using Z, $\mathcal{B}$ replies with a valid ciphertext with a distribution identical to that in the actual construction. So the simulator has $|\Pr[e = e'] - 1/2| \geq \epsilon$.

When Z is random in $\mathbb{G}_T$, then the simulator has $\Pr[e = e'] = 1/2$.

Thus we have

$$\Pr[\mathcal{B}(P, \alpha P, \hat{e}(P,P)^{1/\alpha}) = 0] - \Pr[\mathcal{B}(P, \alpha P, Z) = 0] = \epsilon' \geq \epsilon/2q.$$

The time-complexity of the algorithm $\mathcal{B}$ is identical to that of $\mathcal{A}$. ■

V. An Efficient SGC-PKE Scheme

In this section, we give an efficient Self-Generated-Certificate (SGC) encryption scheme based on the above Certificateless encryption scheme. Most algorithms are the same as the algorithms of certificateless encryption scheme, except for **SetPublicKey** and **Encrypt**. In order to distinguish the algorithm of CL-encryption, we will add the prefix **CL.** to the corresponding algorithms. For example, we use **CL.Setup** to denote the encryption algorithm of the CL-encryption scheme. The proposed SGC-encryption scheme is described as follow:

A. The Scheme

As our CL-PKE system, we assume the existence of a trusted Key Generation Center (KGC) that is responsible for the creation and secure distribution of users' partial secret keys.

Setup: Same as **CL.Setup**. This algorithm takes a security parameter k as its input, and outputs $params = \langle q, \mathbb{G}, \mathbb{G}_T, \hat{e}, P, P_1, Q_1, Q_2, Q_3, H\rangle$ as the system parameters, and keeps α as his own secret master key.

Partial-Secret-Key (PSK) Extract: As **CL.Partial-Secret-Key (PSK) Extract**.

Set-Secret-Value: As **CL.Set-Secret-Value**.

Set-Private-Key: As **CL.Set-Private-Key**.

Set-Public-Key: User selects a secret $r \in \mathbb{Z}_p^*$ as his secret key sk_{ID} and computes his public key as $pk_{ID} = r(\alpha - ID)P$.

Next, it does the following to sign the pk_{ID} using the user's private key sk_{ID} and $psk_{ID,2}$. Let $\sigma_1 = \frac{1}{sk_{ID}} h_{ID,2}$, $\sigma_2 = -r_{ID,2}P$, $\sigma_3 = sk_{ID}P$. Then, the signature is $\sigma = (\sigma_1, \sigma_2, \sigma_3)$.

Encrypt: To encrypt $m \in \mathbb{G}_T$ using the user's public key $pk_{ID} = r(\alpha - ID)P$, the sender first checks the two equations $\hat{e}(pk_{ID}, P) = \hat{e}(\sigma_3, P_1 - ID \cdot P)$ and $\hat{e}(pk_{ID}, \sigma_1) = \hat{e}(P, Q_2) \cdot \hat{e}(P, \sigma_2)$. If not all of them are satisfied, then the sender outputs $\perp$, indicating that the public key pk_{ID} has been replaced.

Next, the sender generates random $s \in \mathbb{Z}_p^*$ and sends the ciphertext $C = (s \cdot pk_{ID}, \hat{e}(P,P)^s, m \cdot \hat{e}(P, Q_1)^{-s}, \hat{e}(P, Q_2)^s \hat{e}(P, Q_3)^{s\beta})$. where $\beta = H(u, v, w)$. Denote C as $C = (u, v, w, y)$.

Note that $\hat{e}(P,P), \hat{e}(P, Q_1), \hat{e}(P, Q_2), \hat{e}(P, Q_3)$ can be precomputed. So the encryption algorithm requires 4 pairing computations in all (including the checking operations).

Decrypt: As **CL.Decrypt**.

Correctness: If the user's public key $pk_{ID} = r(\alpha - ID)P$ has not been replaced, the checking equation in encryption is correct:

$$\begin{aligned}
\hat{e}(pk_{ID}, P) &= \hat{e}(r(\alpha - ID)P, P) \\
&= \hat{e}(rP, (\alpha - ID)P) \\
&= \hat{e}(\sigma_3, P_1 - ID \cdot P), \\
\hat{e}(pk_{ID}, \sigma_1) &= \hat{e}(r(\alpha - ID)P, \tfrac{1}{sk_{ID}} h_{ID,2}) \\
&= \hat{e}(P, Q_2 - r_{ID,2}P) \\
&= \hat{e}(P, Q_2) \cdot \hat{e}(P, \sigma_2).
\end{aligned}$$

The other correctness in the decryption are the same as our CL-encryption scheme.

B. Discussion

The first SGC-PKE scheme was proposed by Liu and Au [17]. It is generic based on a CL-encryption scheme and a CL-signature scheme that use the same set of public parameters and user key generation algorithm. This construction mechanism substantially decreases the computational efficiency.

Recently, Lai and Kou [15] proposed a new SGC-PKE scheme without using pairing. However their SGC-PKE scheme construction is different from their CL-PKE's in the **Partial-Key-Extract** algorithm and **Set-Private-Key** algorithm, which may be problematic in the implementation.

Our scheme is the third SGC-PKE scheme to date. The scheme is more efficient than that in [17]. Moveover the public key length of the scheme is shorter than theirs. It follows our CL-PKE scheme by adding an extra signature with the other operations of the CL-PKE scheme unchanged, which has some advantages over the scheme in [15] in the implementation.

VI. Conclusion

In this paper, we presented an efficient CL-PKE scheme which is constructed from the identity-based encryption scheme proposed by Gentry [10]. Based on the security of the q-ABDHE assumption and 1-BDHI assumption we showed that the presented scheme has the weak type I and strong Type II security in the standard model. The new scheme is much more efficient than the existing CL-PKE schemes on computation and published public key information. Furthermore, based on the new CL-PKE scheme we also gave an SGC-encryption scheme by introducing an extra signature with other parts unchanged, which is more practical than the existing schemes. Nevertheless, we can only achieve the weak Type I security for the CL-PKE scheme. It is still an open problem to design a practical CL-PKC and SGC-PKC scheme with strong Type I and strong Type II security in the standard model.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (Grant No. 60473057 and 60803154) and Doctoral Innovation Foundation of Beihang University (Grant No. 211619). Part of this work was done while the first author was with University of Wollongong, Australia.

REFERENCES

[1] S.S. Al-Riyami, K.G. Paterson, *Certificateless Public Key Cryptography*, In: Advances in Cryptology-ASIACRYPT 2003, Springer, Heidelberg: LNCS 2894, 452-473, 2003.

[2] S.S. Al-Riyami, K.G. Paterson, *CBE from CL-PKE: A Generic Construction and Efficient Schemes*, In: Public Key Cryptography-PKC 2005, Springer, Heidelberg: LNCS 3386, 398-415, 2005.

[3] M. H. Au, J. Chen, J. K. Liu, Y. Mu, D. S. Wong and G. Yang, *Malicious KGC Attack in Certificateless Cryptography*, In Proc. ACM Symposium on Information, Computer and Communications Security, ACM Press, 2007.

[4] J. Baek, R. Safavi-Naini, W. Susilo, *Certificateless Public Key Encryption Without Pairing*, In: 8th Information Security Conference (ISC 2005), Springer, Heidelberg: LNCS 3650, 134-148, 2005.

[5] K. Bentahar, P. Farshim, J. Malone-Lee, N.P. Smart, *Generic Constructions of Identity-Based and Certificateless KEMs*, Journal of Cryptology, 21(2): 178-199, 2008.

[6] Z. Cheng, L. Chen, L. Ling, R. Comley, *General and Efficient Certificateless Public Key Encryption Constructions*, In: Pairing 2007, Springer, Heidelberg: LNCS 4575, 83-107, 2007.

[7] L. Chen, Z. Cheng, N.P. Smart, *Identity-based Key Agreement Protocols From Pairings*, International Journal Information Security, 6: 213-241, 2007.

[8] A.W. Dent, *A Survey of Certificateless Encryption Schemes and Security Models*, International Journal of Information Security, 7(5): 349–377, 2008.

[9] A. W. Dent, B. Libert, and K. G. Paterson, *Certificateless Encryption Schemes Strongly Secure in the Standard Model*, In: Public Key Cryptography-PKC 2008, Springer, Heidelberg: LNCS 4939, 344–359, 2008.

[10] C. Gentry, *Practical Identity-Based Encryption Without Random Oracles*, In Porc. of EUROCRYPT 2006, Springer, Heidelberg: LNCS 4004, 445-464, 2006.

[11] D. Galindo, P. Morillo, C. Rafols, *Breaking Yum and Lee Generic Constructions of Certificateless and Certificate-Based Encryption Schemes*, In: EuroPKI 2006, Springer, Heidelberg: LNCS 4043, 81-91, 2006.

[12] Q. Huang, D.S. Wong, *Generic Certificateless Encryption in the Standard Model*, In: 2nd International Workshop on Security (IWSEC 2007), Springer, Heidelberg: LNCS 4752, 278-291, 2007.

[13] B. Hu, D.S. Wong, Z. Zhang, X. Deng, *Key Replacement Attack Against a Generic Construction of Certificateless Signature*, In: The 11th Australasian Conference on Information Security and Privacy (ACISP 2006), Springer, Heidelberg: LNCS 4058, 235-246, 2006.

[14] X. Huang, W. Susilo, Y. Mu, and F. Zhang, *Certificateless Designated Verifier Signature Schemes*, In: The Second International Workshop on Security in Networks and Distributed Systems (SNDS 2006), IEEE Computer Society, 15-19, 2006.

[15] J. Lai and W. Kou, *Self-Generated-Certificate Public Key Encryption Without Pairing*, In: Public Key Cryptography-PKC'07, Springer, Heidelberg: LNCS 4450, 476-489, 2007.

[16] B. Libert, J.J. Quisquater, *On Constructing Certificateless Cryptosystems From Identity Based Encryption*, In: Public Key Cryptography-PKC'06, Springer, Heidelberg: LNCS 3958, 474-490, 2006.

[17] J.K. Liu, M.H. Au, 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, New York, 2007.

[18] J. H. Park, K. Y. Choi, J. Y. Hwang and D. H. Lee, *Certificateless Public Key Encryption in the Selective-ID Security Model (Without Random Oracles)*, In: Pairing 2007, Springer, Heidelberg: LNCS 4575, 60-82, 2007.

[19] Y. Shi, J. Li, *Provable Efficient Certificateless Public Key Encryption*, Cryptology ePrint Archive, Report 2005/287, http://eprint.iacr.org/2005/287, 2005.

[20] D. Yum, P. Lee, *Generic Construction of Certificateless Encryption*, In: International Conference of Computational Science and Its Applications-ICCSA'04, Springer, Heidelberg: LNCS 3043, 802-811, 2004.