Efficient certificate-based encryption schemes without pairing — Source link

Minh-Ha Le, Intae Kim, Seong Oun Hwang

Institutions: Hongik University

Published on: 01 Dec 2016 - Security and Communication Networks (John Wiley & Sons, Ltd)

Topics: Key escrow, Probabilistic encryption, Encryption, Certificate-based encryption and Public-key cryptography

SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2016; **9**:5376–5391
Published online 8 December 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1703

RESEARCH ARTICLE

Efficient certificate-based encryption schemes without pairing

Minh-Ha Le[1], Intae Kim[1] and Seong Oun Hwang[2]*

[1] Department of Electronics and Computer Engineering, Hongik University, Sejong-ro 2639, Jochiwon, Sejong 339-701, Korea
[2] Department of Computer and Information Communications Engineering, Hongik University, Sejong-ro 2639, Jochiwon, Sejong 339-701, Korea

ABSTRACT

Recently, a lot of researches focused on identity-based encryption (IBE). The advantage of this scheme is that it can reduce the cost of the public key infrastructure by simplifying certificate management. Although IBE has its own innovations, one of its weaknesses is the key escrow problem. That is, the private key generator in IBE knows decryption keys for all identities and consequently can decrypt any ciphertexts. The certificate-based encryption (CBE) scheme proposed in EUROCRYPT 2003 provides a solution for the key escrow problem by allowing the certification authority to possess a partial decryption key that comprises the full decryption key together with the user-generated private key. In this paper, we propose new CBE schemes without pairing and prove them to be Indistinguishability under Chosen Ciphertext Attack secure in the random oracle model based on the hardness of the computational Diffie–Hellman problem. When compared with other CBE schemes, our schemes are significantly efficient in terms of performance, which makes our schemes suitable for computation-limited node (e.g., sensor, wearable device) networks. Copyright © 2016 John Wiley & Sons, Ltd.

KEYWORDS
cryptography; certificate-based encryption; identity-based encryption; pairing

***Correspondence**
Seong Oun Hwang, Department of Computer and Information Communications Engineering, Hongik University, Sejong-ro 2639, Jochiwon, Sejong 339-701, Korea.
E-mail: sohwang@hongik.ac.kr

1. INTRODUCTION

Cryptography initially started with the concept of symmetric key cryptography. In this method, a sender and a receiver share the same key for both encryption and decryption. It requires a secure channel to share the private key. In practice, the secure channel is not easy to achieve. In order to resolve the drawback of symmetric key cryptography, Diffie and Hellman [1] introduced a new concept of public key cryptography (PKC).

In PKC, each user has a pair of keys: public key and private key. The private key is kept secretly by the user and the public key is published so that any one who wants to encrypt a message for this user can obtain it. The two keys are mathematically related, but it is difficult to find the corresponding private key when only the public key is given. If Alice wants to encrypt a message and send it to Bob, she uses Bob's public key for encryption. When Bob receives the encrypted message, he uses his private key to decrypt it. Nevertheless, the problem of authenticating the public keys still remains: How does Alice make sure that the public key she uses for encryption belongs to Bob? To overcome this problem, the public key infrastructure (PKI) is usually deployed.

In PKI, there is a trusted party called Certificate Authority (CA) to generate a certificate corresponding to the user's public key. If Alice wants to make encryption to Bob, she needs to obtain Bob's certificate and then extract the public key from this. It seems to be a good solution, but in the other side, it raises another issue: the PKI practically requires a huge cost and high complexity for management of certificates, and this becomes one of barriers that hinder PKC from being widely deployed in the real world.

To resolve the issue of certificate management, identity-based encryption (IBE) was invented by Shamir [2] in 1984, and the first practical IBE scheme was proposed by Boneh and Franklin in 2001 [3]. After that, a lot of IBEs have been published [4–8] .The idea of IBE is to use the identity (ID) of a user as his or her public key. The ID is obviously certified by out-of-band method that all users can trust. In IBE, there is a private key generator (PKG) that generates the private key for users corresponding to their IDs. The security of IBE is based on the assumption that the PKG is fully trusted. This point becomes a weakness of IBE, the so-called *key escrow* problem—the PKG can decrypt any encrypted messages of users without their permissions.

The concept of certificate-based encryption (CBE) was introduced by Gentry [9] that combined the ideas of PKC and IBE. In CBE, each user generates his own private and public keys. The public key together with identity information is given to the CA, which in turn generates a certificate based on these parameters. The certificate is sent to the users over an open channel. A user encrypts a message using a receiver's public key and does not need to query the certificate's status corresponding to the public key. Decryption is performed using both the private key and the certificate. The point is that the CA does not know the full decryption key so that the key escrow problem is resolved, and the PKI can be maintained in a more efficient way because it eliminates third-party query.

In parallel with CBE, the certificateless public key encryption (CL-PKE) is another solution for key escrow problem [10]. In CL-PKE, the PKG generates a partial private key for each user. Users generate their public and private keys for themselves, but to obtain the full decryption keys, they need to combine their private keys and the partial private keys from the PKG. CL-PKE also lessens the certificate management problem in a traditional PKC. CL-PKE does not require certificates, but it requires secure channels to distribute partial private keys of the PKG to the users.

1.1. Related work

The first CBE scheme was proposed in 2003 by Gentry [9]. The idea of the first CBE scheme is based on the IBE scheme of Boneh-Franklin [3], where the authors used the Fujisaki–Okamoto transform to obtain full security in the random oracle model. Yum [11] and Dodis [12] constructed a generic construction of CBE from IBE, respectively. Later, Yum's construction was broken by Galindo [13]. Many researchers have investigated the relationship between CBE and CL-PKE. In 2005, Al-Riyami and Pterson [14] proposed a generic construction of CBE from CL-PKE but its security was controversial. Kang and Park [15] found out a flaw in the security proof in [14]. Following this direction, Wu *et al.* [16] proposed a secure construction of CBE from CL-PKC, which is proved in the random oracle model, but it still involves collision hash functions. An important version in terms of security for generic construction of CBE from CL-PKC is revisited in [17]. In 2006, Morillo *et al.* introduced a concrete construction of CBE in the standard model [18]. Later, Galindo, Morillo, and Rafols [19] proposed an improved scheme from [18]. Another CBE scheme in the standard model was proposed by Lu and Li [20] against key replacement attack. Recently, many applications of CBE have been proposed. Hyla and Pejas [21] proposed a certificate-based group-oriented encryption scheme with an effective secret-sharing based on general access structure. Sur *et al.* [22] proposed a certificate-based proxy re-encryption for public cloud storage, which has the advantages of CBE while providing the functionalities of proxy re-encryption. Fan *et al.* [23] proposed anonymous multi-receiver CBE that is CPA secure. Liu [24] and Lu [25] proposed efficient pairing-based CBE schemes, respectively. Lu and Li [26] proposed a pairing-free certificate-based proxy re-encryption scheme for secure data sharing in public clouds. Yu *et al.* [27,28] proposed CBE schemes that are resilient to key leakage. Li *et al.* [29] discussed continuous leakage-resilient security model of CBE where the adversary continuously obtains partial information about the secret states through the continuous leakage attacks. Recently, Yao [30] and Lu [31] proposed new CBE schemes without pairing, respectively. Currently, there are three CBE schemes that are not based on pairing, [30–32]. Unfortunately, Lu and Zhang in [32] pointed out a security law in [30] so that there remained only two legitimate CBE schemes without pairing.

1.2. Our contribution

Our goal is to construct efficient CBE schemes without pairing, which was raised as an open problem in the paper [33]. We solve this problem in the positive direction by proposing new CBE schemes without pairing under IND-CCA security. Our CBE scheme is one of the most efficient CBE schemes that we know at the moment.

1.3. Organization

The rest of paper is organized as follows. In Section 2, we give some preliminaries of the CDH problem, CBE scheme, and its security model. In Section 3, we give the constructions and security analysis of our scheme. In Section 4, we compare our scheme with other proposed CBE schemes in terms of performance. Finally, we give conclusions in Section 5.

2. PRELIMINARIES

In this section, we first briefly introduce the bilinear pairing map and the computational Diffie–Hellman (CDH) problem. Then we introduce the syntax of CBE scheme and its security model.

2.1. Pairing

We first define bilinear pairing groups $(q, \mathbb{G}, \mathbb{G}_T, g, g_T, e)$, where q is a prime, $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order q, g and g_T are the generators of $\mathbb{G}$ and $\mathbb{G}_T$, $g_T := e(g, g)$, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a non-degenerate bilinear pairing map such that for $a, b \in Z_q$:

- $e(g^a, g^b) = e(g,g)^{ab}$ (bilinearity)
- $e(g,g) \neq 1$ (non-degeneracy)
- $e(g,g)$ is efficiently computable in a polynomial time.

2.2. Computational Diffie–Hellman problem

Let p_1 and p_2 be primes such that $p_2 \mid p_1 - 1$. Let g be a generator of $\mathbb{Z}_{p_1}^*$. The CDH problem can be formulated as follows:

Given (g, g^a, g^b) *for uniformly chosen* $a, b \in \mathbb{Z}_{p_2}^*$, *compute* $T = g^{ab}$

Let $Adv_{\mathbb{Z}_{p_2}^*}^{CDH}(\mathcal{A})$ be the advantage of adversary $\mathcal{A}$ in solving the CDH problem. We say that $\mathcal{A}$ solves the CDH problem if and only if the advantage of $\mathcal{A}$ is greater than ϵ within running time t. The CDH problem is said to be (t, ϵ)-intractable if there is no attacker $\mathcal{A}$ that solves the CDH problem with (t, ϵ).

2.3. Certificate-based encryption scheme

A CBE scheme generally consists of five algorithms as follows:

- **Setup:**
  The algorithm takes as input the security parameter 1^k and return the CA's master key *msk* and the system parameters *params*.
- **SetKeyPair:**
  It takes as input *ID* and *params* and outputs the public key *pk* and the private key *sk* for a user.
- **Certification:**
  It takes as input *ID*, *msk*, *params*, and *pk*, and outputs the certificate *Cert* and new *pk*, which are sent to the user over an open channel.
- **Encryption:**
  It takes as input *ID*, *params*, *pk*, and plaintext *M*, and returns a ciphertext *C*.
- **Decryption:**
  It takes as input *ID*, *params*, *Cert*, *sk*, and ciphertext *C*, and outputs the plaintext *M*.

2.4. Security models of certificate-based encryption

We define two different types of adversaries for CBE scheme: adversary Type I ($\mathcal{A}_I$) and Type II ($\mathcal{A}_{\mathcal{II}}$). $\mathcal{A}_I$ is a kind of an uncertified entity that has no access to the master secret key *msk*. Essentially, $\mathcal{A}_I$ can make queries for a user's private keys, public keys, public key replacements, and certifications of any user that it chooses except the challenge identity ID^*. $\mathcal{A}_I$ can freely replace the public keys of any user but is not allowed to query the target user's certificate. The adversary $\mathcal{A}_{\mathcal{II}}$ is a kind of certificate entity that is given the master secret key *msk*. $\mathcal{A}_{\mathcal{II}}$ is capable of producing certificates. We note that $\mathcal{A}_{\mathcal{II}}$ is not allowed to replace the public key.

Security against type I adversary:

A CBE scheme is Indistinguishability under Chosen Ciphertext Attack secure against type I adversary if no probabilistic polynomial time adversary $\mathcal{A}_I$ has non-negligible advantage in the following game:

- **Setup phase:**
  The challenger $\mathcal{B}$ is given the security parameter 1^k, runs the setup algorithm and returns public parameters (*params*) and master secret key (*msk*). It then gives *params* to $\mathcal{A}_I$ and keeps the *msk* for itself.
- **Phase I:**
  In phase I, the adversary $\mathcal{A}_I$ makes queries, and $\mathcal{B}$ answers as follows:

  PublicKey-Queries: On receiving an identity *ID*, the challenger responds public key *pk* for *ID*.
  PrivateKey-Queries: On receiving an identity *ID*, the challenger checks whether the corresponding public key is correct or not. If the public key is correct, then it responds private key *sk* for *ID*. Otherwise, the query will be aborted.
  PublicKeyReplace-Queries: On receiving an identity *ID* and its public key pk', the challenger replaces the user's original public key *pk* with pk'.
  Certification-Queries: On receiving a tuple (ID, pk), the challenger checks whether the corresponding public key is correct or not. If the public key is correct, then it responds *Cert* for *ID*. Otherwise, the query will be aborted.
  Decryption-Queries: On receiving a tuple (ID, pk, C), the challenger checks whether the corresponding public key is correct or not. If the public key is correct, then it responds plaintext massage *M*. If the public key is not correct, the challenger will find a tuple in the list of random oracles to collect the parameters and calculate the plaintext *M*. If the challenger cannot find enough parameters in the random oracles, it will abort the queries.

- **Challenge phase:**
  In this phase, $\mathcal{A}_I$ outputs two equal-length messages M_0 and M_1. The challenger chooses a bit $\gamma \in \{0, 1\}$ at random and computes the ciphertext C^* with input of $(params, ID^*, pk^*, M_\gamma)$.
- **Phase II:**
  $\mathcal{A}_I$ continues to query the oracles as in phase I. But it is restricted to make certification query of $\langle ID^*, pk^* \rangle$ and decryption query of $\langle ID^*, pk^*, C^* \rangle$.

- **Guess phase:**
  Finally, $\mathcal{A}_I$ terminates the game by outputting a guess γ' for γ. The advantage of $\mathcal{A}_I$ in the game is defined to be

$$Adv_{\mathcal{A}_I} = 2 \mid Pr[\gamma = \gamma'] - 1/2 \mid .$$

Security against type II adversary: A CBE scheme is IND-CCA secure against type II adversary if no probabilistic polynomial time adversary $\mathcal{A}_{\mathcal{II}}$ has non-negligible advantage in the following game:

- **Setup phase:**
  The challenger $\mathcal{B}$ is given the security parameter 1^k. It takes this parameter as input and runs the setup algorithm. The public parameters (*params*) and master secret key (*msk*) are generated and returned to $\mathcal{A}_{\mathcal{II}}$.
- **Phase I:** In phase I, the adversary $\mathcal{A}_{\mathcal{II}}$ makes queries and $\mathcal{B}$ answers as follow:

  PublicKey-Queries: On receiving an identity *ID*, the challenger responds public key *pk* for *ID*.
  PrivateKey-Queries: On receiving an identity *ID*, the challenger checks whether the corresponding public key is correct or not. If the public key is correct, then it responds private key *sk* for *ID*. Otherwise, the query will be aborted.
  Decryption-Queries: On receiving a tuple (ID, pk, C), the challenger checks whether the corresponding public key is correct or not. If the public key is correct, then it responds plaintext massage M. If the public key is not correct, the challenger will find a tuple in the list of random oracles to collect the parameters and calculate the plaintext M. If the challenger cannot find enough parameters in the random oracles, it will abort the queries.

- **Challenge phase:**
  In this phase, $\mathcal{A}_{\mathcal{II}}$ outputs two equal-length messages M_0 and M_1. The challenger chooses a bit $\gamma \in \{0,1\}$ at random and computes the ciphertext C^* with input of $(params, ID^*, pk^*, M_\gamma)$.
- **Phase II:**
  $\mathcal{A}_{II}$ continues to query the oracles as in phase I. But it is restricted to make decryption query of $\langle ID^*, pk^*, C^* \rangle$.
- **Guess phase:**
  Finally, $\mathcal{A}_{\mathcal{II}}$ terminates the game by outputting a guess γ' for γ. The advantage of $\mathcal{A}_{\mathcal{II}}$ in the game is defined to be

$$Adv_{\mathcal{A}_{II}} = 2 \mid Pr[\gamma = \gamma'] - 1/2 \mid .$$

3. THE PROPOSED SCHEMES

There have been a lot of CBE schemes proposed so far, but most of them are based on pairing. Our scheme is CBE without pairing. First, we will give the constructions of the scheme and then security analysis.

3.1. Outline

In order to achieve the goal in the contribution section, we use Schnorr signature [34], which is one of the best signature schemes in terms of performance for authentication purpose. Taking the advantage of Schnorr signature, our CBE scheme is built by non-trivially combining the signature scheme with ElGamal encryption [35], which was modified by Fujisaki–Okamoto's transformation [36]. More specifically, we design the setup and certification algorithms by using key generation and signature generation algorithms of Schnorr signature, respectively. Also, by using key generation algorithm of ElGamal encryption, we design SetKeyPair algorithm. The encryption and decryption algorithms are constructed by combining the verification algorithm of Schnorr signature and the encryption and decryption algorithms of the ElGamal's scheme. Note that additional public key is generated when the CA performs the certification algorithm. We consider this modification as an advantage of our scheme where the CA can enhance the trust of a user's public key by attaching another means to verify the user's public key.

3.2. First construction

In this section, we construct two efficient CBE schemes that are IND-CCA secure without pairing based on the CDH assumption.

3.2.1. Construction of $\mathcal{CBE}_1$.

We define our first CBE construction as follows. The scheme consists of five algorithms:

Setup: On input of a security parameter $k \in \mathbf{Z}^+$, the CA does the following steps:

(1) Generate two large primes p_1 and p_2 such that $p_1 = 2p_2 + 1$. Choose a generator g of $\mathbb{Z}_{p_1}^*$.
(2) Select a random element $\alpha \in \mathbb{Z}_{p_2}^*$ and compute $g_1 = g^\alpha$.
(3) Pick hash functions

$$H_1 : \{0,1\}^* \times \mathbb{Z}_{p_1}^* \times \mathbb{Z}_{p_1}^* \to \mathbb{Z}_{p_2}^*;$$
$$H_2 : \mathbb{Z}_{p_2}^* \times \mathbb{Z}_{p_1}^* \to \mathbb{Z}_{p_2}^*;$$
$$H_3 : \{0,1\}^{n_0} \times \{0,1\}^{n_1} \to \mathbb{Z}_{p_2}^*;$$

$H_4 : \mathbb{Z}_{p_1}^* \times \mathbb{Z}_{p_1}^* \to \{0,1\}^n$, where n_0 is length of plaintext, n is length of ciphertext, and n_1 is length of padding in H_3 and $n_0 + n_1 = n$;

$$H_5 : \mathbb{Z}_{p_2}^* \times \mathbb{Z}_{p_1}^* \times \mathbb{Z}_{p_1}^* \times \mathbb{Z}_{p_1}^* \to \mathbb{Z}_{p_2}^*.$$

The system parameters are $params = (p_1, p_2, g, g_1, H_1, H_2, H_3, H_4, H_5)$, and master secret key is $msk = \alpha$.

SetKeyPair: On input of ID and $params$, this algorithm selects a random element $x_{ID} \in \mathbb{Z}^*_{p_2}$. The user's private key is x_{ID}. Compute $U_{ID} = g^{x_{ID}}$. The user's public key is $PK_{ID} = U_{ID}$. Set key pair as (x_{ID}, U_{ID}).
Certification: On input of an identity information ID and public key U_{ID} of the user, the certifier selects random $\beta_{ID} \in \mathbb{Z}^*_{p_2}$, and then calculates $P_{ID} = g^{\beta_{ID}}$. The public key of user is updated as $PK_{ID} = (U_{ID}, P_{ID})$. Set $Q_{ID} = H_1(ID, U_{ID}, P_{ID})$. This algorithm outputs $Cert_{ID} = \beta_{ID} + \alpha H_2(Q_{ID}, P_{ID})$ and PK_{ID}.
Encryption: On input of plaintext M, public parameters $params$, identity ID, and public key PK_{ID} of the receiver, the sender performs the following steps:

(1) Select random $\sigma \in \{0,1\}^{n_1}$ and compute

$$\begin{aligned} r &= H_3(M, \sigma) \\ Q_{ID} &= H_1(ID, U_{ID}, P_{ID}) \\ H_{ID} &= H_5(Q_{ID}, U_{ID}, P_{ID}, g_1) \\ k_1 &= \left(U_{ID}^{H_{ID}}\right)^r \\ k_2 &= \left(P_{ID} g_1^{H_2(Q_{ID}, P_{ID})}\right)^r \end{aligned}$$

(2) Compute $C_0 = g^r$
(3) Compute $C_1 = H_4(k_1, k_2) \oplus (M \parallel \sigma)$
The sender outputs $C = (C_0, C_1)$

Decryption: On input of ciphertext $C = (C_0, C_1)$, the private key x_{ID}, public key PK_{ID}, and certificate $Cert_{ID}$ of the user, the receiver does the following steps:

(1) Compute

$$\begin{aligned} Q_{ID} &= H_1(ID, U_{ID}, P_{ID}) \\ H_{ID} &= H_5(Q_{ID}, U_{ID}, P_{ID}, g_1) \end{aligned}$$

(2) Compute the concatenation of message M and σ:

$$M \parallel \sigma = H_4\left(C_0^{x_{ID} H_{ID}}, C_0^{Cert_{ID}}\right) \oplus C_1$$

(3) If $g^{H_3(M,\sigma)} = C_0$, then return M by removing σ from $M \parallel \sigma$. Else return $\perp$.

3.2.2. Correctness.

We show that

$$\begin{aligned} & H_4\left(C_0^{x_{ID} H_{ID}}, C_0^{Cert_{ID}}\right) \\ &= H_4\left((g^r)^{x_{ID} H_5(Q_{ID}, U_{ID}, P_{ID}, g_1)}, (g^r)^{\beta_{ID} + \alpha H_2(Q_{ID}, P_{ID})}\right) \\ &= H_4\left(g^{r x_{ID} H_5(Q_{ID}, U_{ID}, P_{ID}, g_1)}, g^{r(\beta_{ID} + \alpha H_2(Q_{ID}, P_{ID}))}\right) \\ &= H_4\left(\left(g^{x_{ID} H_5(Q_{ID}, U_{ID}, P_{ID}, g_1)}\right)^r, \left(g^{\beta_{ID}} (g^{\alpha})^{H_2(Q_{ID}, P_{ID})}\right)^r\right) \\ &= H_4\left(\left(U_{ID}^{H_5(Q_{ID}, U_{ID}, P_{ID}, g_1)}\right)^r, \left(P_{ID} g_1^{H_2(Q_{ID}, P_{ID})}\right)^r\right) \\ &= H_4(k_1, k_2). \end{aligned}$$

Hence

$$H_4\left(C_0^{x_{ID} H_{ID}}, C_0^{Cert_{ID}}\right) = H_4(k_1, k_2)$$

Now, from decryption, we have

$$\begin{aligned} & H_4\left(C_0^{x_{ID} H_{ID}}, C_0^{Cert_{ID}}\right) \oplus C_1 \\ &= H_4(k_1, k_2) \oplus H_4(k_1, k_2) \oplus (M \parallel \sigma) \\ &= M \parallel \sigma. \end{aligned}$$

The aforementioned equation shows that our decryption is correct.

3.2.3. Security analysis of $\mathcal{CBE}_1$.

In this section, we give the security proof for the first construction under the CDH assumption. The two theorems that follow show that our scheme is IND-CCA secure in the random oracle model.

Theorem 1. *If there exists a polynomial time IND-CCA adversary $\mathcal{A_I}$ who can win the security game with an advantage ϵ in random oracle, by making at most q_{H_i} queries to oracle $H_i (i = 1, 2, 3, 4, 5)$, q_{pub} public key queries, q_{prv} private key queries, q_{cer} certification queries, and q_D decryption queries, then there exists a probabilistic polynomial time algorithm $\mathcal{B}$ that can solve the CDH problem with an advantage at least*

$$\epsilon' > \frac{1}{q_{H_3}}\left(\frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{n_1}} - \frac{q_D q_{H_2}}{2^{n_1}} - \frac{q_D}{p_2}\right)$$

Here, e is the base of natural logarithm.

Proof. Let $\mathcal{A}_I$ be an IND-CCA adversary Type I attacker. We construct an algorithm $\mathcal{B}$ that solves the CDH problem by using $\mathcal{A}_I$. Suppose that H_1, H_2, H_3, H_4, and H_5 are random oracles and $\mathcal{A}_I$ can win the game with advantage ϵ. Then $\mathcal{B}$ can have advantage ϵ'.

$\mathcal{B}$ is given an instance of the CDH problem: p_1, p_2, g, g^a, g^b. $\mathcal{B}$ simulates a challenger and answers queries from $\mathcal{A}_I$ as follows:

- **Setup:**
  $\mathcal{B}$ set $g_1 = g^b$. The *params* are set as $(p_1, p_2, g, g_1, H_1, H_2, H_3, H_4, H_5)$. Then the *params* are given to $\mathcal{A}_I$.
  $\mathcal{A}_I$ can query to all succeeding oracles at any time during its attack. $\mathcal{B}$ answers the queries as follows:
  - **H_1 – queries:** On receiving query (ID, U_{ID}, P_{ID}) to H_1: If **H_1List** contains $\langle ID, U_{ID}, P_{ID}, h_1\rangle$, then it returns h_1 to $\mathcal{A}_I$. Else, pick random $h_1 \in \mathbb{Z}^*_{p_2}$, return h_1 to $\mathcal{A}_I$, and add $\langle ID, U_{ID}, P_{ID}, h_1\rangle$ to **H_1List**.
  - **H_2 – queries:** On receiving query (Q_{ID}, P_{ID}) to H_2: If **H_2List** contains $\langle Q_{ID}, P_{ID}, h_2\rangle$, then

it returns h_2 to $\mathcal{A}_I$. Else, pick random $h_2 \in \mathbb{Z}_{p_2}^*$, return h_2 to $\mathcal{A}_I$, and add $\langle Q_{ID}, P_{ID}, h_2 \rangle$ to **H$_2$List**.

- **H$_3$ – queries:** On receiving query (M, σ) to H_3: If **H$_3$List** contains $\langle M, \sigma, h_3 \rangle$, then it returns h_3 to $\mathcal{A}_I$. Else, pick random $h_3 \in \mathbb{Z}_{p_2}^*$, return h_3 to $\mathcal{A}_I$, and add $\langle M, \sigma, h_3 \rangle$ to **H$_3$List**.
- **H$_4$ – queries** : On receiving query (k_1, k_2) to H_4: If **H$_4$List** contains $\langle k_1, k_2, h_4 \rangle$, then it returns h_4 to $\mathcal{A}_I$. Else, pick random $h_4 \in \{0,1\}^n$, return h_4 to $\mathcal{A}_I$, and add $\langle k_1, k_2, h_4 \rangle$ to **H$_4$List**.
- **H$_5$ – queries** : On receiving query $(Q_{ID}, U_{ID}, P_{ID}, g_1)$ to H_5: If **H$_5$ – List** contains $\langle Q_{ID}, U_{ID}, P_{ID}, g_1, h_5 \rangle$, then it returns h_5 to $\mathcal{A}_I$. Else, pick random $h_5 \in \mathbb{Z}_{p_2}^*$, return h_5 to $\mathcal{A}_I$, and add $\langle Q_{ID}, U_{ID}, P_{ID}, g_1, h_5 \rangle$ to **H$_5$List**.

- **Phase I:**

  - **PublicKey-Queries:** On receiving user's identity ID, $\mathcal{B}$ searches on **KeyList** the tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin \rangle$. If the tuple exists, then $\mathcal{B}$ returns PK_{ID} to $\mathcal{A}_I$. Otherwise, choose a random $coin \in \{0,1\}$. Let δ be a probability that $coin = 0$, that is, $Pr[coin = 0] = \delta$. $\mathcal{B}$ chooses a random elements $x_{ID}, \beta_{ID} \in \mathbb{Z}_{p_2}^*$ and computes $U_{ID} = g^{x_{ID}}$, $P_{ID} = g^{\beta_{ID}}$, adds tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin \rangle$ to **KeyList**, and sends PK_{ID} to $\mathcal{A}_I$.
  - **PrivateKey-Queries:** On receiving user's identity ID, $\mathcal{B}$ searches on **KeyList** the tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin \rangle$. If the tuple exists and the $coin = 0$, then $\mathcal{B}$ returns x_{ID} to $\mathcal{A}_I$. Otherwise, abort the simulation and terminate.
  - **PublicKeyReplace-Queries:** On receiving user's identity ID and a new public key PK'_{ID}, $\mathcal{B}$ searches on **KeyList** the tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin \rangle$. $\mathcal{B}$ set $coin = 1$ and updates this tuple as $\langle ID, x_{ID}, \beta_{ID}, PK'_{ID}, coin \rangle$.
  - **Certification-Queries:** On receiving the tuple $\langle ID, PK_{ID} \rangle$, $\mathcal{B}$ searches on **CertList** the tuple $\langle ID, Cert_{ID} \rangle$. If the tuple exists, then $\mathcal{B}$ returns $Cert_{ID}$ to $\mathcal{A}_I$. Otherwise, run **PublicKey-Queries** to obtain the tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin \rangle$. If challenger finds that the PK_{ID} is not correct, it will set $coin = 1$ and update to the **KeyList**.

    * If $coin = 0$, compute $Q_{ID} = H_1(ID, U_{ID}, P_{ID})$ and $Cert_{ID} = \beta_{ID} + \alpha H_2(Q_{ID}, P_{ID})$. Then add $\langle ID, Cert_{ID} \rangle$ to the **CertList** and return $Cert_{ID}$ to $\mathcal{A}_I$.
    * If $coin = 1$, add $\langle ID, * \rangle$ to **CertList**. (Note: * can be any value).

  - **Decryption-Queries:** On receiving tuple $\langle ID, PK_{ID}, C \rangle$ from adversary $\mathcal{A}_I$, $\mathcal{B}$ searches **CertList** for tuple $\langle ID, Cert_{ID} \rangle$. Query to the **PublicKey-Queries** oracle for tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin \rangle$. If challenger finds that the ID and PK_{ID} given by adversary do not match, it will set $coin = 1$ and update to the **KeyList**.

    * If $coin = 0$: Compute $H_{ID} = H_5(Q_{ID}, U_{ID}, P_{ID}, g_1)$ and run the decryption: $M \parallel \sigma = H_4\left(C_0^{x_{ID}H_{ID}}, C_0^{Cert_{ID}}\right) \oplus C_1$. Return message M to $\mathcal{A}_I$.
    * If $coin = 1$: Run **H$_1$ – queries:** to obtain the tuple $\langle ID, U_{ID}, P_{ID}, h_1 \rangle$ and **H$_2$ – queries:** to obtain the tuple $\langle Q_{ID}, P_{ID}, h_2 \rangle$. If there exist $\langle M, \sigma, h_3 \rangle$ in **H3List**, $\langle k_1, k_2, h_4 \rangle$ in **H$_4$List** and $\langle Q_{ID}, U_{ID}, P_{ID}, g_1, h_5 \rangle$ in **H$_5$List**, such that $C_0 = g^r, C_1 = h_3 \oplus (M \parallel \sigma)$, and $k_1 = \left(U_{ID}^{h_5}\right)^{h_3}$, $k_2 = \left(P_{ID} g_1^{h_2}\right)^{h_3}$, then return message M to $\mathcal{A}_I$.
    * Reject otherwise.

- **Challenge:**

  $\mathcal{A}_I$ outputs two messages (M_0, M_1) together with challenge identity ID^*. On receiving the challenge query, $\mathcal{B}$ answers as follows: Run the aforementioned simulation for **PublicKey-Queries** with input ID^* to obtain the tuple $\langle ID^*, x_{ID}^*, \beta^*, PK_{ID}^*, coin \rangle$ from **KeyList**. If $coin = 0$, $\mathcal{B}$ aborts the simulation and returns "Abort." Otherwise, $\mathcal{B}$ does the following:

  - Pick random values:

$$\sigma^* \in \{0,1\}^{n_1}, C_1^* \in \{0,1\}^n, \gamma \in \{0,1\}$$

  - Set:

$$C_0^* = g^a$$
$$Q_{ID}^* = H_1\left(ID^*, U_{ID}^*, P_{ID}^*\right)$$
$$h_2^* = H_2\left(Q_{ID}^*, P_{ID}^*\right)$$
$$h_5^* = H_5\left(Q_{ID}^*, U_{ID}^*, P_{ID}^*, g_1\right)$$

  - Define: $a = H_3(M_\gamma, \sigma^*)$ and $H_4(k_1, k_2) = C_1^* \oplus (M_\gamma, \sigma^*)$, where $k_1 = \left(U_{ID}^{*h_5^*}\right)^a, k_2 = \left(P_{ID}^* g_1^{h_2^*}\right)^a$.
  - Return: $C^* = \left(C_0^*, C_1^*\right)$.

Note: By the construction earlier, we have

$$\begin{aligned} C_1^* &= H_4(k_1, k_2) \oplus (M_\gamma \parallel \sigma^*) \\ &= H_4\left(g^{ax_{ID}^* H_{ID}}, g^{a\left(\beta^* + bH_2\left(Q_{ID}^*, g^{\beta^*}\right)\right)}\right) \oplus (M_\gamma \parallel \sigma^*) \\ &= H_4\left((g^a)^{x_{ID}^* H_{ID}}, (g^a)^{\beta^*}(g^{ab})^{H_2\left(Q_{ID}^*, g^{\beta^*}\right)}\right) \oplus (M_\gamma \parallel \sigma^*) \end{aligned}$$

where $H_{ID} = H_5(Q_{ID}^*, g^{x_{ID}^*}, g^{\beta^*}, g^b)$.

- **Phase II:**

  $\mathcal{B}$ repeats the same simulations as Phase I, except:

  - Issue (ID^*, PK_{ID}^*) as certification query.
  - Issue (ID^*, PK_{ID}^*, C^*) as decryption query.

- **Guess:**

  Adversary $\mathcal{A}_I$ outputs a guess γ' of γ and then sends it to the challenger $\mathcal{B}$. Now $\mathcal{B}$ chooses a tuple $\langle k_1, k_2, h_4 \rangle$ from the **H₄List** and returns $(\frac{k_2}{(g^a)^{\beta^*}})^{1/h_2^*}$ as the solution for the CDH problem. From the construction of H_1, H_2, and H_5, it is clear that H_1, H_2, and H_5 are perfect. As long as $\mathcal{A}_\mathcal{I}$ does not query $\langle M_\gamma, \sigma^* \rangle$ to H_3 and $\langle (U_{ID}^{*h_5^*})^a, (P_{ID} g_1^{h_2^*})^a \rangle$ to H_4, the simulation of H_3 and H_4 is perfect. We denote

  - $\mathbf{Q}_{\mathbf{H_3}}^*$ is the event that $\langle M_\gamma, \sigma^* \rangle$ has been queried to H_3.
  - $\mathbf{Q}_{\mathbf{H_4^*}}$ is the event that $\langle (U_{ID}^{*h_5^*})^a, (P_{ID} g_1^{h_2^*})^a \rangle$ has been queried to H_4.

If $\mathbf{Q}_{\mathbf{H_4^*}}$ happens, then $\mathcal{B}$ will be able to solve the CDH problem by choosing a tuple $\langle k_1, k_2, h_4 \rangle$ from the **H₄List** and compute $(\frac{k_2}{(g^a)^{\beta^*}})^{1/h_2^*}$ with probability as least $1/q_{H_4}$. Hence, we have

$$\epsilon' \geq \frac{1}{q_{H_4}} Pr\left[\mathbf{Q}_{\mathbf{H_4^*}}\right] \tag{1}$$

Next, it is noticed that if $\mathcal{B}$ does not abort, then simulation of **PublicKey-Queries**, **PrivateKey-Queries**, **Certification-Queries,** and target ciphertext is identically distributed as the real one from the construction.

Now we evaluate the construction of the decryption oracle. If PK_{ID} has been produced as *coin* = 0, the simulation is perfect as $\mathcal{B}$ knows the corresponding private key x_{ID} of U_{ID}. Otherwise, an error simulation may occur. We denote this event is $\mathbf{D_{err}}$ and find probability of this event. Suppose that the input $\langle ID, PK_{ID}, C \rangle$ of **Decryption-Queries** oracle is valid, where $C = (C_0, C_1)$. Let $\mathbf{C_{valid}}$ is the event that C is valid. Let $\mathbf{Q_{H_3}}$ and $\mathbf{Q_{H_4}}$ respectively be events that $\langle M_\gamma, \sigma \rangle$ has been queried to H_3 and $\langle k_1, k_2 \rangle$ has been queried to H_4 with respect to $C = (C_0, C_1) = (g^r, H_4(k_1, k_2)), k_1 = (U_{ID}^{h_5})^r, k_2 = (P_{ID} g_1^{h_2})^r$, where $h_5 = H_5(Q_{ID}, P_{ID}, U_{ID}, g_1)$ and $h_3 = H_3(M, \sigma)$. Even if C is valid, there is a possibility that C can be produced without querying to H_4. We have $Pr[\mathbf{D_{err}}] = q_D Pr[\mathbf{C_{valid}} \mid \neg \mathbf{Q_{H_4}}]$. But

$$\begin{aligned} &Pr[\mathbf{C_{valid}} \mid \neg\mathbf{Q_{H_4}}] \leq Pr[\mathbf{C_{valid}} \cap \mathbf{Q_{H_3}} \mid \\ &\neg\mathbf{Q_{H_4}}] + Pr[\mathbf{C_{valid}} \cap \neg\mathbf{Q_{H_3}} \mid \neg\mathbf{Q_{H_4}}] \\ &\leq Pr[\mathbf{Q_{H_3}} \mid \neg\mathbf{Q_{H_4}}] + Pr[\mathbf{C_{valid}} \mid \neg\mathbf{Q_{H_3}} \cap \\ &\neg\mathbf{Q_{H_4}}] \\ &\leq \frac{q_{H_3}}{2^{n_1}} + \frac{1}{p_2} \end{aligned}$$

So $Pr[\mathbf{D_{err}}] \leq \frac{q_D q_{H_3}}{2^{n_1}} + \frac{q_D}{p_2}$.

Now, we define an event $\mathbf{E}$: $(\mathbf{Q}_{\mathbf{H_4}}^* \cup (\mathbf{Q}_{\mathbf{H_3}}^* \mid \neg\mathbf{Q_{H_4}}) \cup \mathbf{D_{err}} \mid \neg\mathbf{Abort})$, where **Abort** denotes $\mathcal{B}$ aborts during simulation. $Pr[\neg\mathbf{Abort}] = \delta^{q_{prv}}(1-\delta)$, which is maximized at $\delta = 1 - 1\backslash(q_{prv}+1)$. So, $Pr[\neg\mathbf{Abort}] \leq \frac{1}{e(p_{prv}+1)}$, where e denotes the base of the natural logarithm.

We have

$$Pr[\mathbf{E}] = \left(Pr\left[\mathbf{Q}_{\mathbf{H_4}}^*\right] + Pr\left[\mathbf{Q}_{\mathbf{H_3}}^* \mid \neg\mathbf{Q}_{\mathbf{H_4}}^*\right] + Pr[\mathbf{D_{err}}]\right) \frac{1}{Pr[\neg\mathbf{Abort}]}$$

So that $Pr[\mathbf{Q}_{\mathbf{H_4}}^*] = Pr[\mathbf{E}]Pr[\neg\mathbf{Abort}] - Pr[\mathbf{Q}_{\mathbf{H_3}}^* \mid \neg\mathbf{Q}_{\mathbf{H_4}}^*] - Pr[\mathbf{D_{err}}]$

Since $Pr[\mathbf{Q}_{\mathbf{H_3}}^* \mid \neg\mathbf{Q}_{\mathbf{H_4}}^*] \leq \frac{q_{H_3}}{2^{n_1}}$

we obtain

$$Pr\left[\mathbf{Q}_{\mathbf{H_4}}^*\right] \geq \frac{Pr[\mathbf{E}]}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{n_1}} - \frac{q_D q_{H_2}}{2^{n_1}} - \frac{q_D}{p_2} \tag{2}$$

Meanwhile

$$\begin{aligned} \epsilon &\leq \mid Pr[\gamma' = \gamma] - \frac{1}{2} \mid \\ &= \mid Pr[\gamma' = \gamma \mid \neg\mathbf{E}]Pr[\neg\mathbf{E}] \\ &\quad + Pr[\gamma' = \gamma \mid \mathbf{E}]Pr[\mathbf{E}] - \frac{1}{2} \\ &\leq \mid \frac{1}{2}Pr[\neg\mathbf{E}] + Pr[\mathbf{E}] - \frac{1}{2} \mid = \frac{1}{2}Pr[\mathbf{E}] \end{aligned}$$

So

$$Pr[\mathbf{E}] \geq 2\epsilon \tag{3}$$

From (2) and (3), we have

$$Pr\left[\mathbf{Q}_{\mathbf{H_4}}^*\right] \geq \frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{n_1}} - \frac{q_D q_{H_2}}{2^{n_1}} - \frac{q_D}{p_2} \tag{4}$$

From (1) and (4), we have

$$\epsilon' > \frac{1}{q_{H_4}}\left(\frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{n_1}} - \frac{q_D q_{H_2}}{2^{n_1}} - \frac{q_D}{p_2}\right)$$

□

Theorem 2. *If there exists a polynomial time IND-CCA adversary $\mathcal{A}_{\mathcal{II}}$ who can win the security game with an advantage ϵ in random oracle, by making at most q_{H_i} queries to oracle H_i ($i = 1,2,3,4,5$), q_{pub} public key queries, q_{prv} private key queries, and q_D decryption queries, then there exists a probabilistic polynomial time algorithm $\mathcal{B}$ that can solve the CDH problem with advantage at least*

$$\epsilon' > \frac{1}{q_{H_3}}\left(\frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{n_1}} - \frac{q_D q_{H_2}}{2^{n_1}} - \frac{q_D}{p_2}\right)$$

Here, e is the base of natural logarithm.

Proof. Let $\mathcal{A}_{\mathcal{II}}$ be an IND-CCA adversary Type II attacker. We construct an algorithm $\mathcal{B}$ that solve the CDH problem by using $\mathcal{A}_{\mathcal{II}}$. Suppose that H_1, H_2, H_3, H_4, and H_5 are random oracles and $\mathcal{A}_{\mathcal{II}}$ can win the game with advantage ϵ. Then $\mathcal{B}$ can have advantage ϵ'.

$\mathcal{B}$ is given an instance of the CDH problem: p_1, p_2, g, g^a, g^b. $\mathcal{B}$ simulates a challenger and answers queries from $\mathcal{A}_{\mathcal{II}}$ as follows:

- **Setup:** $\mathcal{B}$ selects random element $\alpha \in \mathbb{Z}^*_{p_2}$ and calculates $g_1 = g^{\alpha}$. The *params* are set as $(p_1, p_2, g, g_1, H_1, H_2, H_3, H_4, H_5)$ and *msk* is α . Then the *params* and *msk* are given to $\mathcal{A}_{\mathcal{II}}$.

  In this proof, we assume that H_1, H_2, H_3, H_4, and H_5 are random oracles and adversary $\mathcal{A}_{\mathcal{II}}$ can query to all these oracles at any time during its attack. $\mathcal{B}$ answers the queries as follows:

  - $\mathbf{H_1}$ – **queries:** On receiving query (ID, U_{ID}, P_{ID}) to H_1: If $\mathbf{H_1}$**List** contains $\langle ID, U_{ID}, P_{ID}, h_1\rangle$, then it returns h_1 to $\mathcal{A}_{\mathcal{II}}$. Else, pick random $h_1 \in \mathbb{Z}^*_{p_2}$, return h_1 to $\mathcal{A}_{\mathcal{II}}$, and add $\langle ID, U_{ID}, P_{ID}, h_1\rangle$ to $\mathbf{H_1}$**List**.
  - $\mathbf{H_2}$ – **queries:** On receiving query (Q_{ID}, P_{ID}) to H_2: If $\mathbf{H_2}$**List** contains $\langle Q_{ID}, P_{ID}, h_2\rangle$, then it returns h_2 to $\mathcal{A}_{\mathcal{II}}$. Else, pick random $h_2 \in \mathbb{Z}^*_{p_2}$, return h_2 to $\mathcal{A}_{\mathcal{II}}$, and add $\langle Q_{ID}, P_{ID}, h_2\rangle$ to $\mathbf{H_2}$**List**.
  - $\mathbf{H_3}$ – **queries:** On receiving query (M, σ) to H_3: If $\mathbf{H_3}$**List** contains $\langle M, \sigma, h_3\rangle$, then it returns h_3 to $\mathcal{A}_{\mathcal{II}}$. Else, pick random $h_3 \in \mathbb{Z}^*_{p_2}$, return h_3 to $\mathcal{A}_{\mathcal{II}}$, and add $\langle M, \sigma, h_3\rangle$ to $\mathbf{H_3}$**List**.
  - $\mathbf{H_4}$ – **queries** : On receiving query (k_1, k_2) to H_4: If $\mathbf{H_4}$**List** contains $\langle k_1, k_2, h_4\rangle$, then it returns h_4 to $\mathcal{A}_{\mathcal{II}}$. Else, pick random $h_4 \in \{0,1\}^n$, return h_4 to $\mathcal{A}_{\mathcal{II}}$, and add $\langle k_1, k_2, h_4\rangle$ to $\mathbf{H_4}$**List**.
  - $\mathbf{H_5}$ – **queries** : On receiving query $(Q_{ID}, U_{ID}, P_{ID}, g_1)$ to H_5: If $\mathbf{H_5}$ – **List** contains $\langle Q_{ID}, U_{ID}, P_{ID}, g_1, h_5\rangle$, then it returns h_5 to $\mathcal{A}_{\mathcal{II}}$. Else, pick random $h_5 \in \mathbb{Z}^*_{p_2}$, return h_5 to $\mathcal{A}_{\mathcal{II}}$, and add $\langle Q_{ID}, U_{ID}, P_{ID}, g_1, h_5\rangle$ to $\mathbf{H_5}$**List**.

- **Phase I:**

  - **PublicKey-Queries:** On receiving user's identity ID, $\mathcal{B}$ searches on **KeyList** the tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin\rangle$. If the tuple exists, then $\mathcal{B}$ returns PK_{ID} to $\mathcal{A}_{\mathcal{II}}$. Otherwise, choose a random $coin \in \{0,1\}$, let δ be a probability that $coin = 0$, that is, $Pr[coin = 0] = \delta$. $\mathcal{B}$ chooses random element $x_{ID}, \beta_{ID} \in \mathbb{Z}^*_{p_2}$ and computes $U_{ID} = g^{x_{ID}}$, $P_{ID} = g^{\beta_{ID}}$, adds tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin\rangle$ to **KeyList**, then sends PK_{ID} to $\mathcal{A}_{\mathcal{II}}$.
  - **PrivateKey-Queries:** On receiving user's identity ID, $\mathcal{B}$ searches on **KeyList** the tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin\rangle$. If the tuple exists and the $coin = 0$, then $\mathcal{B}$ returns x_{ID} to $\mathcal{A}_{\mathcal{II}}$. Otherwise, abort the simulation and terminate.
  - **Decryption-Queries:** On receiving tuple $\langle ID, PK_{ID}, C\rangle$ from adversary $\mathcal{A}_{\mathcal{II}}$, $\mathcal{B}$ queries to the **PublicKey-Queries** oracle for tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin\rangle$. If it founds that ID and PK_{ID} given by adversary do not match, it will set $coin = 1$ and update to the **KeyList**.

    * If $coin = 0$, then compute $H_{ID} = H_5(Q_{ID}, U_{ID}, P_{ID}, g_1)$ and run the decryption:
      $M \parallel \sigma = H_4(C_0^{x_{ID}H_{ID}}, C_0^{Cert_{ID}}) \oplus C_1$. Return message M to $\mathcal{A}_{\mathcal{II}}$.
    * If $coin = 1$: Run $\mathbf{H_1}$ – **queries:** to obtain the tuple $\langle ID, U_{ID}, P_{ID}, h_1\rangle$ and $\mathbf{H_2}$ – **queries:** to obtain the tuple $\langle Q_{ID}, P_{ID}, h_2\rangle$. If there exists $\langle M, \sigma, h_3\rangle$ in **H3List**, $\langle k_1, k_2, h_4\rangle$ in $\mathbf{H_4}$**List**, and $\langle Q_{ID}, U_{ID}, P_{ID}, g_1, h_5\rangle$ in $\mathbf{H_5}$**List**, such that $C_0 = g^r, C_1 = h_3 \oplus (M \parallel \sigma)$, and $\langle k_1, k_2\rangle = \langle (U_{ID}^{h_5})^{h_3}, (P_{ID}g_1^{h_2})^{h_3}\rangle$, then return message M to $\mathcal{A}_{\mathcal{II}}$.
    * Reject otherwise.

- **Challenge:**
  $\mathcal{A}_{\mathcal{II}}$ outputs two messages (M_0, M_1) together with challenge identity ID^*. On receiving the challenge query, $\mathcal{B}$ answers as follows: Run the aforementioned simulation for **PublicKey-Queries** with input ID^* to obtain the tuple $\langle ID^*, x^*_{ID}, \beta^*, PK^*_{ID}, coin\rangle$ from **KeyList**. If $coin = 0$, $\mathcal{B}$ aborts the simulation and returns "Abort." Otherwise, $\mathcal{B}$ does the following:

– Pick random values:

$$\sigma^* \in \{0,1\}^{n_1}, C_1^* \in \{0,1\}^n, \gamma \in \{0,1\}$$

– Set: $P_{ID}^* = (g^b)^{\beta^*}$
– Set: $C_0^* = g^a$

$$Q_{ID}^* = H_1\left(ID^*, U_{ID}^*, P_{ID}^*\right)$$
$$h_2^* = H_2\left(Q_{ID}^*, P_{ID}^*\right)$$
$$h_5^* = H_5\left(Q_{ID}^*, U_{ID}^*, P_{ID}, g_1\right)$$

– Define: $a = H_3(M_\gamma, \sigma^*)$

$$H_4(k_1, k_2) = C_1^* \oplus (M_\gamma, \sigma^*)$$

where $k_1 = (U_{ID}^{*h_5^*})^a, k_2 = (P_{ID}^* g_1^{h_2^*})^a$.
– Return: $C^* = (C_0^*, C_1^*)$. Note: By the construction earlier, we have

$$C_1^* = H_4(k_1, k_2) \oplus (M_\gamma \parallel \sigma^*)$$
$$= H_4\left(g^{ax_{ID}^* H_{ID}}, g^{a\left(b\beta^* + \alpha H_2\left(Q_{ID}^*, g^{\beta^*}\right)\right)}\right) \oplus$$
$$(M_\gamma \parallel \sigma^*)$$
$$= H_4\left(g^{ax_{ID}^* H_{ID}}, (g^{ab})^{\beta^*}(g^a)^{\alpha H_2\left(Q_{ID}^*, g^{\beta^*}\right)}\right)$$
$$\oplus (M_\gamma \parallel \sigma^*)$$

where $H_{ID} = H_5\left(Q_{ID}^*, g^{x_{ID}^*}, g^{\beta^*}, g^b\right)$

- **Phase II:**
  $\mathcal{B}$ repeats the same simulations as Phase I, except $\langle ID^*, PK_{ID}^*, C^* \rangle$ as decryption query.
- **Guess:**
  Adversary $\mathcal{A}_{\mathcal{II}}$ outputs a guess γ' of γ then sends it to challenger $\mathcal{B}$. Now $\mathcal{B}$ chooses a tuple $\langle k_1, k_2, h_4 \rangle$ from the **H₄List** and returns $(\frac{k_2}{g^{\alpha a h_2^*}})^{1/\beta^*}$ as the solution for the CDH problem.
- **Analysis:**
  With the same method as Theorem 1, we have

$$\epsilon' > \frac{1}{q_{H_4}}\left(\frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{n_1}} - \frac{q_D q_{H_2}}{2^{n_1}} - \frac{q_D}{p_2}\right)$$

□

3.3. Second construction

Inspired from the construction of CL-PKE in [37], we propose a more computation-efficient CBE scheme that is IND-CCA secure without pairing under the CDH assumption. Instead of using exponentiation for verification of the decrypted message in the first construction, we use hash function for this purpose in the second construction. It makes the length of ciphertext longer but reduces the computational cost.

3.3.1. Construction of $\mathcal{CBE}_2$.

We define our second CBE construction as follows. The scheme consisting of five algorithms is almost the same as the first scheme except **Encryption** and **Decryption** algorithms.

Setup: On input of a security parameter $k \in \mathbf{Z}^+$, the CA does the following steps:

(1) Generate two large primes p_1 and p_2 such that $p_1 = 2p_2 + 1$. Choose a generator g of $\mathbb{Z}_{p_1}^*$.
(2) Select a random element $\alpha \in \mathbb{Z}_{p_2}^*$ and compute $g_1 = g^{\alpha}$.
(3) Pick hash functions:

$$H_1 : \{0,1\}^* \times \mathbb{Z}_{p_1}^* \times \mathbb{Z}_{p_1}^* \to \mathbb{Z}_{p_2}^*$$
$$H_2 : \mathbb{Z}_{p_2}^* \times \mathbb{Z}_{p_1}^* \to \mathbb{Z}_{p_2}^*$$
$$H_3 : \{0,1\}^{n_0} \times \{0,1\}^{n_1} \to \mathbb{Z}_{p_2}^*$$

$H_4 : \mathbb{Z}_{p_1}^* \to \{0,1\}^n$, where n_0 is length of plaintext, n is length of ciphertext, and n_1 is length of padding in H_3 and $n_0 + n_1 = n$ and

$$H_5 : \{0,1\}^{n_1} \times \mathbb{Z}_{p_1}^* \to \mathbb{Z}_{p_2}^*$$

The system parameters are $params = (p_1, p_2, g, g_1, H_1, H_2, H_3, H_4, H_5)$ and master secret key is $msk = \alpha$.

SetKeyPair: On input of ID and $params$, this algorithm selects a random element $x_{ID} \in \mathbb{Z}_{p_2}^*$. Calculate $U_{ID} = g^{x_{ID}}$. The key pair will be (x_{ID}, U_{ID}).
Certification: On input of an identity information ID and public key U_{ID} of a user, the certifier selects random $\beta_{ID} \in \mathbb{Z}_{p_2}^*$ and then calculates $P_{ID} = g^{\beta_{ID}}$. The public key is updated as $PK_{ID} = (U_{ID}, P_{ID})$. Set $Q_{ID} = H_1(ID, U_{ID}, P_{ID})$ and compute $Cert_{ID} = \beta_{ID} + \alpha H_2(Q_{ID}, P_{ID})$. This algorithm outputs $Cert_{ID}$ and PK_{ID}.
Encryption: On input of plaintext M, public parameters $params$, identity ID of the receiver, U_{ID}, and a part of $Cert_{ID}$, the sender performs the following steps:

(1) Select random $\sigma \in \{0,1\}^{n_1}$ and compute as follows:

$$r = H_3(M, \sigma)$$
$$Q_{ID} = H_1(ID, U_{ID}, P_{ID})$$
$$k = \left(U_{ID} P_{ID} g_1^{H_2(Q_{ID}, P_{ID})}\right)^r$$

(2) Compute $C_0 = g^r$.
(3) Compute $C_1 = H_4(k) \oplus (M \parallel \sigma)$.
(4) Compute $C_2 = H_5(C_0, M)$. The sender outputs $C = (C_0, C_1, C_2)$.

Decryption: On input of ciphertext $C = (C_0, C_1, C_2)$, the private key x_{ID}, public key PK_{ID}, and certificate $Cert_{ID}$ of the user, the receiver does the following steps:

(1) Compute the concatenation of message M and σ:

$$M \parallel \sigma = H_4\left(C_0^{x_{ID}+Cert_{ID}}\right) \oplus C_1$$

(2) If $H_5(C_0, M) = C_2$, then return M by removing σ from $M \parallel \sigma$. Else return $\perp$.

3.3.2. Correctness.

We show that

$$\begin{aligned} & H_4\left(C_0^{x_{ID}+Cert_{ID}}\right) \\ &= H_4\left((g^r)^{x_{ID}+\beta_{ID}+\alpha H_2(Q_{ID},P_{ID})}\right) \\ &= H_4\left(g^{rx_{ID}+r(\beta_{ID}+\alpha H_2(Q_{ID},P_{ID}))}\right) \\ &= H_4\left(\left(g^{x_{ID}} g^{\beta_{ID}} (g^{\alpha})^{H_2(Q_{ID},P_{ID})}\right)^r\right) \\ &= H_4\left(\left(U_{ID} P_{ID} g_1^{H_2(Q_{ID},P_{ID})}\right)^r\right) \\ &= H_4(k) \end{aligned}$$

Hence

$$H_4\left(C_0^{x_{ID}+Cert_{ID}}\right) = H_4(k)$$

Now, from decryption, we have

$$\begin{aligned} & H_4\left(C_0^{x_{ID}+Cert_{ID}}\right) \oplus C_1 \\ &= H_4(k) \oplus H_4(k) \oplus (M \parallel \sigma) \\ &= M \parallel \sigma \end{aligned}$$

The aforementioned equation shows that our decryption is correct.

3.3.3. Security analysis of $\mathcal{CBE}_2$.

In this section, we briefly describe the security proof for the second construction under the CDH assumption. The two theorems given in the first construction are applied in the same way to the second scheme showing that the second construction is also IND-CCA secure in the random oracle model. Because the overall proof structure of the second construction is the same as that of the first construction, we describe major differences as follows. The full proof of the second construction is presented in the Appendix.

- **H_5 – queries** in the game against Adversary Types I and II: On receiving query (C_0, M) to H_5: If **H_5**List contains $\langle C_0, M, h_5\rangle$, then it returns h_5 to $\mathcal{A}_I$. Else, pick random $h_5 \in \mathbb{Z}_{p2}^*$, return h_5 to $\mathcal{A}_I$, and add $\langle C_0, M, h_5\rangle$ to **H_5**List.
- **Guess Phase** in the game against Adversary Type I: Adversary $\mathcal{A}_I$ outputs a guess γ' of γ and then sends it to the challenger $\mathcal{B}$. Then $\mathcal{B}$ chooses a tuple $\langle k, h_4\rangle$ from the **H_4List** and returns
$$\left(\frac{k}{(g^a)^{x_{ID}^*}(g^a)^{\beta^*}}\right)^{1/h_2^*}$$
as the solution for the CDH problem, where $k = (U_{ID}^* P_{ID}^* g_1^{h_2^*})^a$.
- **Guess Phase** in the game against Adversary Type II: Adversary $\mathcal{A}_{\mathcal{II}}$ outputs a guess γ' of γ then sends it to challenger $\mathcal{B}$. Then $\mathcal{B}$ chooses a tuple $\langle k, h_4\rangle$ from the **H_4List** and returns $\left(\frac{k}{(g^a)^{x_{ID}^*}(g^a)^{\alpha h_2^*}}\right)^{1/\beta^*}$ as the solution for the CDH problem.

4. PERFORMANCE COMPARISON

Efficiency of the proposed certificate-based encryption schemes is analyzed and compared with the previous certificate-based encryption schemes in terms of encryption and decryption.

Table I shows the comparison results among the schemes in terms of computation overheads. As shown in Table I, we can see that most schemes carry out the pairing operation except ours. So, we can expect that our schemes are more efficient than other schemes, which will be explained with the following figure (Figure 1).

In order to provide a practical performance comparison in the real world, we perform simulations as follows.

Table I. Performance comparison.

Scheme	Encryption	Decryption
[9]	$3E + 1E_T + 2P$	$1P$
[16]	$3E + 1E_T + 3P$	$4E + 4P$
[19]	$5E + 2E_T + 2P + S$	$5E + 3P + V$
[18]	$5E + 2E_T + 2P$	$3P$
[38]	$2E + 1E_T + 1P$	$1E + 1E_T + 1P$
[25]	$3E + 1E_T + Enc$	$2E + 2P + Dec$
[31]	$3E_Z$	$3E_Z$
[32]	$3E$	$2E$
$\mathcal{CBE}_1$	$4E_Z$	$3E_Z$
$\mathcal{CBE}_2$	$3E_Z$	$1E_Z$

E, exponentiation operation time in $\mathbb{G}$; E_T, exponentiation operation time in $\mathbb{G}_T$; P, pairing operation time; E_Z, exponentiation operation time in $\mathbb{Z}_p$; S, signature generation time; V, signature verification time; Enc, encryption time of the symmetric encryption; Dec, decryption time of the symmetric decryption.

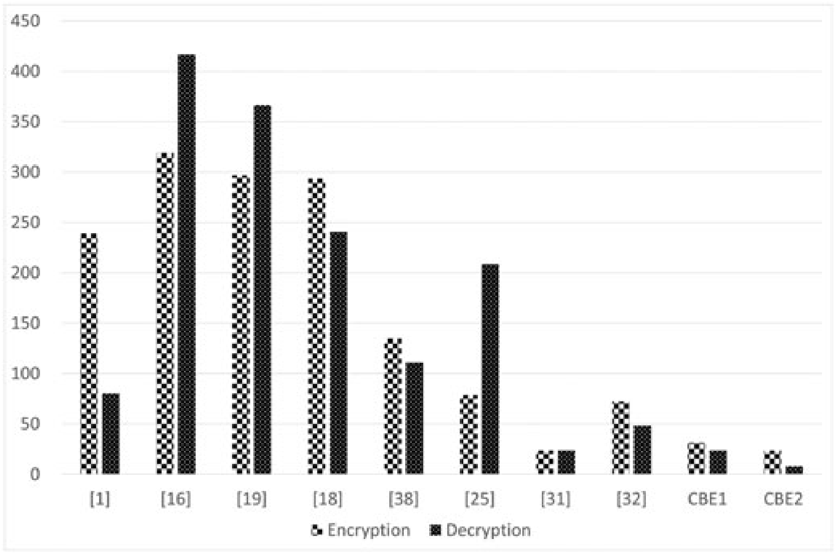

Figure 1. Comparison of the computation time.

Table II. The time of basic operations.

Operation	**Time** (ms)
E	24.039
E_T	6.464
P	80.194
E_Z	7.764
S	3.389
V	5.564
Enc	0.009
Dec	0.006

In Table II, E, E_T, and P are measured in the Tate pairing over a supersingular curve with an embedding degree of 2 at a 128 bit AES security level; E_Z is measured in 2048 bit integer exponentiation operation; S and V are measured in elliptic curve digital signature algorithm [39]; and *Enc* and *Dec* are measured in AES-128.

Each value in Table II was measured using a Windows 7 64-bit system with a 3.70 GHz AMD Phenom II X4 980 processor. The MIRACL v7.0.1 library (https://certivox.org/display/EXT/MIRACL) was used in our test.

In Figure 1, we compare the encryption and decryption time between the existing schemes and ours.

In the encryption aspect, we can see that $\mathcal{CBE}_2$ and [31] are the most efficient schemes because they require only three exponentiations. But, $\mathcal{CBE}_1$ is slightly inefficient than [31], because it requires one more E_Z operation than that of [31].

In the decryption aspect, we can see that $\mathcal{CBE}_2$ is the most efficient scheme because it requires only one exponentiation. However, $\mathcal{CBE}_1$ and [31] are the second most efficient ones, because they require three E_Z operations.

From the earlier analysis, we can confirm that our proposed schemes are significantly efficient in the sense that encryption and decryption can be computed without pairing.

We think that the $\mathcal{CBE}_2$ achieves efficiency particularly in a wireless sensor network where the computation capabilities of nodes are very limited and most frequent computations are decryption. Although the $\mathcal{CBE}_2$ boasts its computational efficiency, it increases communication overhead due to its stretched ciphertext size when compared with the $\mathcal{CBE}_1$. Therefore, the $\mathcal{CBE}_1$ seems appropriate in bandwidth-critical networks as in regular key update network, for example, where all of client nodes make key update requests to the key management server at the same time, which generates a big volume of data on the network and may result in congestion as a result.

5. CONCLUSION

This paper proposed two certificate-based encryption schemes that are significantly efficient in terms of computation by eliminating pairing operations. Simulation results show that the proposed schemes are the most efficient and practical ones compared with other schemes in terms of computation. The proposed schemes are proved secure against Types I and II adversaries with IND-CCA security, which can be used efficiently in some practical applications, particularly resource-constrained node networks.

Appendix.

In this section, we give the security proof for the second construction under the CDH assumption. The two theorems that follow show that it is IND-CCA secure in the random oracle model.

Theorem 3. *If there exists a polynomial time IND-CCA adversary $\mathcal{A}_{\mathcal{I}}$ who can win the security game with an advantage ϵ in random oracle, by making at most q_{H_i} queries to oracle $H_i(i = 1,2,3,4,5), q_{pub}$ public key queries, q_{prv} private key queries, q_{cer} certification queries, and q_D decryption queries, then exist a PPT algorithm $\mathcal{B}$ that can solve the CDH problem with an advantage at least*

$$\epsilon' > \frac{1}{q_{H_3}}\left(\frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{n_1}} - \frac{q_D q_{H_2}}{2^{n_1}} - \frac{q_D}{p_2}\right)$$

Here, e is the base of natural logarithm.

Proof. Let $\mathcal{A}_I$ be an IND-CCA adversary Type I attacker. We construct an algorithm $\mathcal{B}$ that solves the CDH problem by using $\mathcal{A}_I$. Suppose that H_1, H_2, H_3, H_4, and H_5 are random oracles and $\mathcal{A}_I$ can win the game with advantage ϵ in polynomial time, then $\mathcal{B}$ can have advantage ϵ' in polynomial time.

$\mathcal{B}$ is given an instance of the CDH problem: p_1, p_2, g, g^a, g^b. $\mathcal{B}$ simulates a challenger and answers queries from $\mathcal{A}_I$ as follows:

- **Setup:**
  $\mathcal{B}$ set $g_1 = g^b$. The *params* are set as $(p_1, p_2, g, g_1, H_1, H_2, H_3, H_4, H_5)$. Then the *params* are given to $\mathcal{A}_I$.

  $\mathcal{A}_I$ can query to all succeeding oracles at any time during its attack. $\mathcal{B}$ answers the queries as follows:
  - **H_1 – queries:** On receiving query (ID, U_{ID}, P_{ID}) to H_1: If **H_1List** contains $\langle ID, U_{ID}, P_{ID}, h_1\rangle$, then it returns h_1 to $\mathcal{A}_I$. Else, pick random $h_1 \in \mathbb{Z}^*_{p_2}$, return h_1 to $\mathcal{A}_I$, and add $\langle ID, U_{ID}, P_{ID}, h_1\rangle$ to **H_1List**.
  - **H_2 – queries:** On receiving query (Q_{ID}, P_{ID}) to H_2: If **H_2List** contains $\langle Q_{ID}, P_{ID}, h_2\rangle$, then it returns h_2 to $\mathcal{A}_I$. Else, pick random $h_2 \in \mathbb{Z}^*_{p_2}$, return h_2 to $\mathcal{A}_I$, and add $\langle Q_{ID}, P_{ID}, h_2\rangle$ to **H_2List**.
  - **H_3–queries:** On receiving query (M, σ) to H_3: If **H_3List** contains $\langle M, \sigma, h_3\rangle$, then it returns h_3 to $\mathcal{A}_I$. Else, pick random $h_3 \in \mathbb{Z}^*_{p_2}$, return h_3 to $\mathcal{A}_I$, and add $\langle M, \sigma, h_3\rangle$ to **H_3List**.
  - **H_4 – queries** : On receiving query k to H_4: If **H_4List** contains $\langle k, h_4\rangle$, then it returns h_4 to $\mathcal{A}_I$. Else, pick random $h_4 \in \{0,1\}^n$, return h_4 to $\mathcal{A}_I$, and add $\langle k, h_4\rangle$ to **H_4List**.
  - **H_5 – queries** : On receiving query (C_0, M) to H_5: If **H_5**List contains $\langle C_0, M, h_5\rangle$, then it returns h_5 to $\mathcal{A}_I$. Else, pick random $h_5 \in \mathbb{Z}^*_{p_2}$, return h_5 to $\mathcal{A}_I$, and add $\langle C_0, M, h_5\rangle$ to **H_5**List.

- **Phase I:**
  - **PublicKey-Queries:** On receiving user's identity ID, $\mathcal{B}$ searches on **KeyList** the tuple $(ID,\ x_{ID}, \beta_{ID}, PK_{ID})$. If the tuple exists, then $\mathcal{B}$ returns PK_{ID} to $\mathcal{A}_I$. Otherwise, choose a random $coin \in \{0,1\}$. Let δ be a probability that $coin = 0$, that is, $Pr[coin = 0] = \delta$. $\mathcal{B}$ chooses a random element x_{ID} and β_{ID} from $\mathbb{Z}^*_{p_2}$ and computes $U_{ID} = g^{x_{ID}}$, $P_{ID} = g^{\beta_{ID}}$, adds tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin\rangle$ to **KeyList**, and sends PK_{ID} to $\mathcal{A}_I$.
  - **PrivateKey-Queries:** On receiving user's identity ID, $\mathcal{B}$ searches on **KeyList** the tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin\rangle$. If the tuple exists and the $coin = 0$, then $\mathcal{B}$ returns x_{ID} to $\mathcal{A}_I$. Otherwise, abort the simulation and terminate.
  - **PublicKeyReplace-Queries:** On receiving user's identity ID and a new public key PK'_{ID}, $\mathcal{B}$ searches on **KeyList** the tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin\rangle$. $\mathcal{B}$ set $coin = 1$ and updates this tuple as $\langle ID, x_{ID}, \beta_{ID}, PK'_{ID}, coin\rangle$.
  - **Certification-Queries:** On receiving the tuple $\langle ID, PK_{ID}\rangle$, $\mathcal{B}$ searches on **CertList** the tuple $\langle ID, Cert_{ID}\rangle$. If the tuple exists, then $\mathcal{B}$ returns $Cert_{ID}$ to $\mathcal{A}_{\mathcal{II}}$.

    Otherwise, run **PublicKey – Queries** to obtain the tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin\rangle$. If challenger finds that the PK_{ID} is not correct, it will set $coin = 1$ and update to the **KeyList**.
    * If $coin = 0$, compute $Q_{ID} = H_1(ID, U_{ID}, P_{ID})$ and $Cert_{ID} = \beta_{ID} + \alpha H_2(\ Q_{ID}, P_{ID})$. Then, add $\langle ID, Cert_{ID}\rangle$ to the **CertList** and return $Cert_{ID}$ to $\mathcal{A}_{\mathcal{II}}$.
    * If $coin = 1$, add $\langle ID, *\rangle$ to **CertList** (Note: * can be any value).
  - **Decryption-Queries:** On receiving tuple $\langle ID, PK_{ID}, C\rangle$ from adversary $\mathcal{A}_I$, $\mathcal{B}$ queries to the **PublicKey-Queries** oracle for tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin\rangle$. If challenger finds that the ID and PK_{ID} given by adversary do not match, it will set $coin = 1$ and update to the **KeyList**.
    * If $coin = 0$: Run the decryption: $M \parallel \sigma = H_4(C_0^{x_{ID}+Cert_{ID}}) \oplus C_1$. Return message M to $\mathcal{A}_I$.
    * If $coin = 1$: Run **H_1 – queries:** to obtain the tuple $\langle ID, U_{ID}, P_{ID}, h_1\rangle$ and **H_2 – queries:** to obtain the tuple $\langle Q_{ID}, P_{ID}, h_2\rangle$. If there exists $\langle M, \sigma, h_3\rangle$ in **H3List**, $\langle k, h_4\rangle$ in **H_4List**, and $\langle C_0, M, h_5\rangle$ in **H_5**List, such that $C_0 = g^r, C_1 = h_4 \oplus (M \parallel \sigma)$, and $k = (U_{ID}P_{ID}g_1^{h_2})^{h_3}$. Return message M to $\mathcal{A}_I$.
    * Reject otherwise.

- **Challenge:**
  $\mathcal{A}_I$ outputs two messages (M_0, M_1) together with challenge identity ID^*. On receiving the challenge query, $\mathcal{B}$ answers as follows: Run the earlier simulation for **PublicKey-Queries** with input ID^* to obtain the tuple $\langle ID^*, x^*_{ID}, \beta^*, PK^*_{ID}, coin\rangle$ from **KeyList**. If $coin = 0$, $\mathcal{B}$ aborts the simulation and returns "Abort." Otherwise, $\mathcal{B}$ does the following:

  - Pick random values: $\sigma^* \in \{0,1\}^{n_1}, C^*_1 \in \{0,1\}^n, \gamma \in \{0,1\}$
  - Set: $C^*_0 = g^a$, $Q^*_{ID} = H_1(ID^*, U^*_{ID}, P^*_{ID})$, $h^*_2 = H_2(Q^*_{ID}, P^*_{ID})$
  - Define: $a = H_3(M_\gamma, \sigma^*)$ and $H_4(k) = C^*_1 \oplus (M_\gamma, \sigma^*)$, where $k = (U_{ID}P^*_{ID}g_1^{h^*_2})^a$
  - Set: $C^*_2 = H_5(C^*_0, M_\gamma)$
  - Return: $C^* = (C^*_0, C^*_1, C^*_2)$

  Note: By the construction earlier, we have

$$\begin{aligned} C^*_1 &= H_4(k) \oplus (M_\gamma \parallel \sigma^*) \\ &= H_4((g^{x^*_{ID}} g^{\beta^*} g^{bh^*_2})^a) \oplus (M_\gamma \parallel \sigma^*) \\ &= H_4((g^a)^{x^*_{ID}} (g^a)^{\beta^*} (g^{ab})^{h^*_2}) \oplus (M_\gamma \parallel \sigma^*) \end{aligned}$$

- **Phase II:**
  $\mathcal{B}$ repeats the same simulations as Phase I, except:

  - Issue $\langle ID^*, PK^*_{ID}\rangle$ as certification query.
  - Issue $\langle ID^*, PK^*_{ID}, C^*\rangle$ as decryption query.

- **Guess:**
  Adversary $\mathcal{A}_I$ outputs a guess γ' of γ and then sends it to the challenger $\mathcal{B}$. Now $\mathcal{B}$ chooses a tuple $\langle k, h_4\rangle$ from the **H₄List** and returns

$$\left(\frac{k}{(g^a)^{x^*_{ID}}(g^a)^{\beta^*}}\right)^{1/h^*_2}$$

  as the solution for the CDH problem.

  From the construction of H_1, H_2, and H_5, it is clear that H_1, H_2, and H_5 are perfect. As long as $\mathcal{A}_{\mathcal{I}}$ does not query $\langle M, \sigma^*\rangle$ to H_3 and $k = (U_{ID}P^*_{ID}g_1^{h^*_2})^a$ to H_4, the simulation of H_3 and H_4 are perfect. We denote

  – $\mathbf{Q}^*_{\mathbf{H_3}}$ is the event that $\langle M_\gamma, \sigma\rangle$ has been queried to H_3
  – $\mathbf{Q}_{\mathbf{H}^*_4}$ is the event that $k = (U_{ID}P^*_{ID}g_1^{h^*_2})^a$ has been queried to H_4

If $\mathbf{Q}_{\mathbf{H}^*_4}$ happens, then $\mathcal{B}$ will be able to solve the CDH problem by choosing a tuple $\langle k, h_4\rangle$ from the **H₄List** and compute $(\frac{k}{(g^a)^{x^*_{ID}}(g^a)^{\beta^*}})^{1/h^*_2}$ with probability as least $1/q_{H_4}$. Hence, we have

$$\epsilon' \geq \frac{1}{q_{H_4}} Pr\left[\mathbf{Q}_{\mathbf{H}^*_4}\right] \tag{5}$$

Next, it is noticed that if $\mathcal{B}$ does not abort, then simulation of **PublicKey-Queries**, **PrivateKey-Queries**, **Certification-Queries**, and target ciphertext is identically distributed as the real one from the construction.

Now we evaluate the construction of the decryption oracle. If U_{ID} has been produced as $coin = 0$, the simulation is perfect as $\mathcal{B}$ knows the corresponding private key x_{ID} of U_{ID}. Otherwise, an error simulation may occur. We denote this event is $\mathbf{D_{err}}$ and find probability of this event. Suppose that the input $\langle ID, PK_{ID}, C\rangle$ of **Decryption-Queries** oracle is valid, where $C = (C_0, C_1, C_2)$. Let $\mathbf{C_{valid}}$ is the event that C is valid. Let $\mathbf{Q_{H_3}}$ and $\mathbf{Q_{H_4}}$ respectively be events that $\langle M_\gamma, \sigma\rangle$ has been queried to H_3 and $\langle k\rangle$ has been queried to H_4 with respect to $C = (C_0, C_1, C_2) = (g^r, H_4(k), H_5(C_0, M))$, where $r = H_3(M, \sigma), k = (U_{ID}P_{ID}g_1^{h_2})^r$. Even if C is valid, there is a possibility that C can be produced without querying to H_4. We have $Pr[\mathbf{D_{err}}] = q_D Pr[\mathbf{C_{valid}} \mid \neg\mathbf{Q_{H_4}}]$. But

$$\begin{aligned} &Pr[\mathbf{C_{valid}} \mid \neg\mathbf{Q_{H_4}}] \leq Pr[\mathbf{C_{valid}} \cap \mathbf{Q_{H_3}} \\ &\mid \neg\mathbf{Q_{H_4}}] + Pr[\mathbf{C_{valid}} \cap \neg\mathbf{Q_{H_3}} \mid \neg\mathbf{Q_{H_4}}] \\ &\leq Pr[\mathbf{Q_{H_3}} \mid \neg\mathbf{Q_{H_4}}] + Pr[\mathbf{C_{valid}} \mid \neg\mathbf{Q_{H_3}} \cap \\ &\neg\mathbf{Q_{H_4}}] \\ &\leq \frac{q_{H_3}}{2^{n_1}} + \frac{1}{p_2} \end{aligned}$$

So $Pr[\mathbf{D_{err}}] \leq \frac{q_D q_{H_3}}{2^{n_1}} + \frac{q_D}{p_2}$.

Now, we define an event $\mathbf{E}$: $(\mathbf{Q}^*_{\mathbf{H_4}} \cup (\mathbf{Q}^*_{\mathbf{H_3}} \mid \neg\mathbf{Q_{H_4}}) \cup \mathbf{D_{err}} \mid \neg\mathbf{Abort})$, where **Abort** denotes $\mathcal{B}$ aborts during simulation. $Pr[\neg\mathbf{Abort}] = \delta^{q_{prv}}(1-\delta)$ which is maximized at $\delta = 1 - 1\backslash(q_{prv}+1)$. So, $Pr[\neg\mathbf{Abort}] \leq \frac{1}{e(p_{prv}+1)}$, where e denotes the base of the natural logarithm.

We have

$$\begin{aligned} Pr[\mathbf{E}] = \Big(&Pr[\mathbf{Q}^*_{\mathbf{H_4}}] + Pr[\mathbf{Q}^*_{\mathbf{H_3}} \mid \neg\mathbf{Q}^*_{\mathbf{H_4}}] + \\ &Pr[\mathbf{D_{err}}]\Big) \frac{1}{Pr[\neg\mathbf{Abort}]} \end{aligned}$$

So that $Pr[\mathbf{Q}^*_{\mathbf{H_4}}] = Pr[\mathbf{E}]Pr[\neg\mathbf{Abort}] - Pr[\mathbf{Q}^*_{\mathbf{H_3}} \mid \neg\mathbf{Q}^*_{\mathbf{H_4}}] - Pr[\mathbf{D_{err}}]$

Since $Pr\left[\mathbf{Q}^*_{\mathbf{H_3}} \mid \neg\mathbf{Q}^*_{\mathbf{H_4}}\right] \leq \frac{q_{H_3}}{2^{n_1}}$
we obtain

$$Pr\left[\mathbf{Q}^*_{\mathbf{H_4}}\right] \geq \frac{Pr[\mathbf{E}]}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{n_1}} - \frac{q_D q_{H_2}}{2^{n_1}} - \frac{q_D}{p_2} \tag{6}$$

Meanwhile

$$\begin{aligned}\epsilon &\leq\mid Pr[\gamma' = \gamma] - \frac{1}{2}\mid \\ &=\mid Pr[\gamma' = \gamma \mid \neg\mathbf{E}]Pr[\neg\mathbf{E}] \\ &\quad + Pr[\gamma' = \gamma \mid \mathbf{E}]Pr[\mathbf{E}] - \frac{1}{2} \\ &\leq\mid \frac{1}{2}Pr[\neg\mathbf{E}] + Pr[\mathbf{E}] - \frac{1}{2}\mid = \frac{1}{2}Pr[\mathbf{E}]\end{aligned}$$

So that

$$Pr[\mathbf{E}] \geq 2\epsilon. \qquad (7)$$

From (2) and (3), we have

$$Pr[\mathbf{Q}^*_{\mathbf{H_4}}] \geq \frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{n_1}} - \frac{q_D q_{H_2}}{2^{n_1}} - \frac{q_D}{p_2} \qquad (8)$$

From (1) and (4), we have $\epsilon' > \frac{1}{q_{H_4}}(\frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{n_1}} - \frac{q_D q_{H_2}}{2^{n_1}} - \frac{q_D}{p_2})$

□

Theorem 4. *If there exists a polynomial time IND-CCA adversary $\mathcal{A}_{\mathcal{II}}$ who can win the security game with an advantage ϵ in random oracle, by making at most q_{H_i} queries to oracle H_i ($i = 1,2,3,4,5$), q_{pub} public key queries, q_{prv} private key queries, and q_D decryption queries, then there exists a PPT algorithm $\mathcal{B}$ that can solve the CDH problem with an advantage at least*

$$\epsilon' > \frac{1}{q_{H_3}}\left(\frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{n_1}} - \frac{q_D q_{H_2}}{2^{n_1}} - \frac{q_D}{p_2}\right)$$

Here, e is the base of natural logarithm.

Proof. Let $\mathcal{A}_{\mathcal{II}}$ be an IND-CCA adversary Type II attacker. We construct an algorithm $\mathcal{B}$ that solve the CDH problem by using $\mathcal{A}_{\mathcal{II}}$. Suppose that H_1, H_2, H_3, H_4, and H_5 are random oracles and $\mathcal{A}_{\mathcal{II}}$ can win the game with advantage ϵ in polynomial time, then $\mathcal{B}$ can have advantage ϵ' in polynomial time.

$\mathcal{B}$ is given an instance of the CDH problem: p_1, p_2, g, g^a, g^b. $\mathcal{B}$ simulates a challenger and answers queries from $\mathcal{A}_{\mathcal{II}}$ as follows:

- **Setup:**
  $\mathcal{B}$ selects random element $\alpha \in \mathbb{Z}^*_{p_2}$ and calculates $g_1 = g^{\alpha}$. The *params* are set as $(p_1, p_2, g, g_1, H_1, H_2, H_3, H_4, H_5)$ and *msk* is α . Then the *params* and *msk* are given to $\mathcal{A}_{\mathcal{II}}$.

  In this proof, we assume that H_1, H_2, H_3, H_4, and H_5 are random oracles and adversary $\mathcal{A}_{\mathcal{II}}$ can query to all these oracles at any time during its attack. $\mathcal{B}$ answers the queries as follows:

  - $\mathbf{H_1}$ – **queries:** On receiving query (ID, U_{ID}, P_{ID}) to H_1: If $\mathbf{H_1List}$ contains $\langle ID, U_{ID}, P_{ID}, h_1\rangle$, then it returns h_1 to $\mathcal{A}_{\mathcal{II}}$. Else, pick random $h_1 \in \mathbb{Z}^*_{p_2}$, return h_1 to $\mathcal{A}_{\mathcal{II}}$, and add $\langle ID, U_{ID}, P_{ID}, h_1\rangle$ to $\mathbf{H_1List}$.
  - $\mathbf{H_2}$ – **queries:** On receiving query (Q_{ID}, P_{ID}) to H_2: If $\mathbf{H_2List}$ contains $\langle Q_{ID}, P_{ID}, h_2\rangle$, then it returns h_2 to $\mathcal{A}_{\mathcal{II}}$. Else, pick random $h_2 \in \mathbb{Z}^*_{p_2}$, return h_2 to $\mathcal{A}_{\mathcal{II}}$, and add $\langle Q_{ID}, P_{ID}, h_2\rangle$ to $\mathbf{H_2List}$.
  - $\mathbf{H_3}$ – **queries:** On receiving query (M, σ) to H_3: If $\mathbf{H_3List}$ contains $\langle M, \sigma, h_3\rangle$, then it returns h_3 to $\mathcal{A}_{\mathcal{II}}$. Else, pick random $h_3 \in \mathbb{Z}^*_{p_2}$, return h_3 to $\mathcal{A}_{\mathcal{II}}$, and add $\langle M, \sigma, h_3\rangle$ to $\mathbf{H_3List}$.
  - $\mathbf{H_4}$ – **queries** : On receiving query k to H_4: If $\mathbf{H_4List}$ contains $\langle k, h_4\rangle$, then it returns h_4 to $\mathcal{A}_{\mathcal{II}}$. Else, pick random $h_4 \in \{0,1\}^n$, return h_4 to $\mathcal{A}_{\mathcal{II}}$, and add $\langle k, h_4\rangle$ to $\mathbf{H_4List}$.
  - $\mathbf{H_5}$ – **queries** : On receiving query (C_0, M) to H_5: If $\mathbf{H_5}$List contains $\langle C_0, M, h_5\rangle$, then it returns h_5 to $\mathcal{A}_{\mathcal{II}}$. Else, pick random $h_5 \in \mathbb{Z}^*_{p_2}$, return h_5 to $\mathcal{A}_{\mathcal{II}}$, and add $\langle C_0, M, h_5\rangle$ to $\mathbf{H_5}$List.

- **Phase I:**
  - **PublicKey – Queries** : On receiving user's identity ID, $\mathcal{B}$ searches on **KeyList** the tuple $(ID, x_{ID}, \beta_{ID}, PK_{ID})$. If the tuple exists, then $\mathcal{B}$ returns PK_{ID} to $\mathcal{A}_{\mathcal{II}}$. Otherwise, choose a random $coin \in \{0,1\}$, let δ be a probability that $coin = 0$, that is, $Pr[coin = 0] = \delta$. $\mathcal{B}$ chooses random element x_{ID}, β_{ID} from $\mathbb{Z}^*_{p_2}$ and computes $U_{ID} = g^{x_{ID}}$, $P_{ID} = g^{\beta_{ID}}$, adds tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin\rangle$ to **KeyList**, then sends PK_{ID} to $\mathcal{A}_{\mathcal{II}}$.
  - **PrivateKey-Queries:** On receiving user's identity ID, $\mathcal{B}$ searches on **KeyList** the tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin\rangle$. If the tuple exists and the $coin = 0$, then $\mathcal{B}$ returns x_{ID} to $\mathcal{A}_{\mathcal{II}}$. Otherwise, abort the simulation and terminate.
  - **Decryption-Queries:** On receiving tuple $\langle ID, PK_{ID}, C\rangle$ from adversary $\mathcal{A}_{\mathcal{II}}$, $\mathcal{B}$ queries to the **PublicKey-Queries** oracle for tuple $\langle ID, x_{ID}, \beta_{ID}, PK_{ID}, coin\rangle$. If it founds that ID and PK_{ID} given by adversary do not match, it will set $coin = 1$ and update to the **KeyList**.
    * If $coin = 0$, then run the decryption $M \parallel \sigma = H_4(C_0^{x_{ID}+Cert_{ID}}) \oplus C_1$. Return message M to $\mathcal{A}_{\mathcal{II}}$.
    * If $coin = 1$: Run $\mathbf{H_1}$ – **queries:** to obtain the tuple $\langle ID, U_{ID}, P_{ID}, h_1\rangle$ and $\mathbf{H_2}$ – **queries:** to obtain the tuple $\langle Q_{ID}, P_{ID}, h_2\rangle$. If there exists $\langle M, \sigma, h_3\rangle$ in **H3List**, $\langle k, h_4\rangle$ in $\mathbf{H_4List}$, and $\langle C_0, M, h_5\rangle$ in $\mathbf{H_5}$List, such that $C_0 = g^r, C_1 = h_4 \oplus (M \parallel \sigma), C_2 = h_5$, and $k = (U_{ID}P_{ID}g_1^{h_2})^{h_3}$. Return message M to $\mathcal{A}_{\mathcal{II}}$.
    * Reject otherwise.

- **Challenge:**
  $\mathcal{A}_{\mathcal{II}}$ outputs two messages (M_0, M_1) together with challenge identity ID^*. On receiving the challenge query, $\mathcal{B}$ answers as follows: Run the aforementioned simulation for **PublicKey-Queries** with input ID^* to obtain the tuple $\langle ID^*, x^*_{ID}, \beta^*, PK^*_{ID}, coin\rangle$ from **KeyList**. If $coin = 0$, $\mathcal{B}$ aborts the simulation and returns "Abort." Otherwise, $\mathcal{B}$ does following:

  - Pick random values: $\sigma^* \in \{0,1\}^{n_1}, C^*_1 \in \{0,1\}^n, \gamma \in \{0,1\}$
  - Set: $P^*_{ID} = (g^b)^{\beta^*}$
  - Set: $C^*_0 = g^a$, $Q^*_{ID} = H_1(ID^*, U^*_{ID}, P^*_{ID})$, $h^*_2 = H_2(Q^*_{ID}, P^*_{ID})$
  - Define: $a = H_3(M_\gamma, \sigma^*)$ and $H_4(k) = C^*_1 \oplus (M_\gamma, \sigma^*)$, where $k = (U^*_{ID}P^*_{ID}g_1^{h^*_2})^a$
  - Set: $C^*_2 = H_5(C_0, M_\gamma)$
  - Return: $C^* = (C^*_0, C^*_1, C^*_2)$. Note: By the construction earlier, we have

$$\begin{aligned} C^*_1 &= H_4(k) \oplus (M_\gamma \parallel \sigma^*) \\ &= H_4((g^{x^*_{ID}}(g^b)^{\beta^*} g^{\alpha h^*_2})^a) \oplus (M_\gamma \parallel \sigma^*) \\ &= H_4((g^a)^{x^*_{ID}}(g^{ab})^{\beta^*}(g^a)^{\alpha h^*_2}) \oplus (M_\gamma \parallel \sigma^*) \end{aligned}$$

- **Phase II:**
  $\mathcal{B}$ repeats the same simulations as Phase I, except $\langle ID^*, PK^*_{ID}, C^*\rangle$ as decryption query.
- **Guess:**
  Adversary $\mathcal{A}_{\mathcal{II}}$ outputs a guess γ' of γ then sends it to challenger $\mathcal{B}$. Now $\mathcal{B}$ chooses a tuple $\langle k, h_4\rangle$ from the $\mathbf{H_4List}$ and returns $(\frac{k}{(g^a)^{x^*_{ID}}(g^a)^{\alpha h^*_2}})^{1/\beta^*}$ as the solution for the CDH problem.
  With the same method as Theorem 1, we have $\epsilon' > \frac{1}{q_{H_4}}(\frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{n_1}} - \frac{q_D q_{H_2}}{2^{n_1}} - \frac{q_D}{p_2})$

□

ACKNOWLEDGEMENTS

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2014R1A1A2054174).

This research was also supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the Global IT Talent support program (IITP-2016-H0905-15-1004) supervised by the IITP (Institute for Information and Communication Technology Promotion).

REFERENCES

1. Diffie W, Hellman ME. New directions in cryptography. *Information Theory, IEEE Transactions on* 1976; **22**(6): 644–654.
2. Shamir A. *Identity-based Cryptosystems and Signature Schemes. Advances in cryptology*. Springer, 1985.
3. Boneh D, Franklin M. *Identity-Based Encryption from the Weil Pairing. Advances in Cryptology—CRYPTO 2001*. Springer, 2001.
4. Kim I, Hwang SO, Kim S. An efficient anonymous identity-based broadcast encryption for large-scale wireless sensor networks. *Ad Hoc & Sensor Wireless Networks* 2012; **14**(1-2): 27–39.
5. Kim I, Hwang SO. An optimal identity-based broadcast encryption scheme for wireless sensor networks. *IEICE transactions on communications* 2013; **96**(3): 891–895.
6. Kim I, Hwang SO. Efficient identity-based broadcast signcryption schemes. *Security and Communication Networks* 2014; **7**(5): 914–925.
7. Hur J, Park C, Hwang SO. Privacy-preserving identity-based broadcast encryption. *Information Fusion* 2012; **13**(4): 296–303.
8. Yum DH, Lee PJ. *Identity-Based Cryptography in Public Key Management. Public Key Infrastructure*. Springer, 2004.
9. Gentry C. *Certificate-based Encryption and the Certificate Revocation Problem. Advances in Cryptology—EUROCRYPT 2003*. Springer, 2003.
10. Al-Riyami SS, Paterson KG. *Certificateless Public Key Cryptography. Advances in Cryptology-ASIACRYPT 2003*. Springer, 2003.
11. Yum DH, Lee PJ. *Generic Construction of Certificateless Encryption. Computational Science and Its Applications–ICCSA 2004*. Springer, 2004.
12. Dodis Y, Katz J. *Chosen-Ciphertext Security of Multiple Encryption. Theory of Cryptography*. Springer, 2005.
13. Galindo D, Morillo P, Ràfols C. *Breaking Yum and Lee Generic Constructions of Certificate-Less and Certificate-based Encryption Schemes. Public Key Infrastructure*. Springer, 2006.
14. Al-Riyami SS, Paterson KG. *CBE From CL-PKE: A Generic Construction and Efficient Schemes. Public Key Cryptography-PKC 2005*. Springer, 2005.
15. Kang BG, Park JH. Is it possible to have CBE from CL-PKE? *IACR Cryptology ePrint Archive* 2005; **2005**: 431.
16. Wu W, Mu Y, Susilo W, Huang X, Xu L. A provably secure construction of certificate-based encryption from certificateless encryption. *The Computer Journal* 2012; **55**(10): 1157–1168, bxr130.
17. Gao W, Wang G, Wang X, Chen K. Generic construction of certificate-based encryption from certificateless encryption revisited. *The Computer Journal* 2015; **58** (10): 2747–2757, bxv045.

18. Morillo P, Ràfols C. Certificate-based encryption without random oracles, 2006.
19. Galindo D, Morillo P, Ràfols C. Improved certificate-based encryption in the standard model. *Journal of Systems and Software* 2008; **81**(7): 1218–1226.
20. Lu Y, Li J. Efficient certificate-based encryption scheme secure against key replacement attacks in the standard model. *Journal of Information Science and Engineering* 2014; **30**(5): 1553–1568.
21. Hyla T, Pejaś J. *Certificate-based Encryption Scheme with General Access Structure. Computer Information Systems and Industrial Management*. Springer, 2012.
22. Sur C, Park Y, Shin S U, Rhee K H, Seo C. Certificate-based proxy re-encryption for public cloud storage. *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013 Seventh International Conference on*, 2013; 159–166.
23. Fan CI, Tsai PJ, Huang JJ, Chen WT. Anonymous multi-receiver certificate-based encryption. *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2013 International Conference on*, 2013; 19—26.
24. Liu JK, Zhou J. Efficient certificate-based encryption in the standard model. In *SCN*, Vol. 8. Springer, 2008; 144–155.
25. Lu Y, Li J. Constructing efficient certificate-based encryption scheme with pairing in the standard model. *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference on*, 2010; 234–237.
26. Lu Y, Li J. A pairing-free certificate-based proxy re-encryption scheme for secure data sharing in public clouds. *Future Generation Computer Systems* 2016; **62**: 140–147.
27. Yu Q, Li J, Zhang Y, Wu W, Huang X, Xiang Y. Certificate-based encryption resilient to key leakage. *Journal of Systems and Software* 2016; **116**: 101–112.
28. Yu Q, Li J, Zhang Y. Leakage-resilient certificate-based encryption. *Security and Communication Networks* 2015; **8**(18): 3346–3355.
29. Li J, Guo Y, Yu Q, Lu Y, Zhang Y, Zhang F. Continuous leakage-resilient certificate-based encryption. *Information Sciences* 2016; **355**: 1–14.
30. Yao J, Li J, Zhang Y. Certificate-based encryption scheme without pairing. *KSII Transactions on Internet and Information Systems (TIIS)* 2013; **7**(6): 1480–1491.
31. Lu Y, Li J. Constructing pairing-free certificate-based encryption. *International Journal of Innovative Computing Information and Control* 2013; **9**(11): 4509–4518.
32. Lu Y, Zhang Q. Enhanced certificate-based encryption scheme without bilinear pairings. *KSII Transactions on Internet & Information Systems* 2016; **10**(2): 881–896.
33. Baek J, Safavi-Naini R, Susilo W. *Certificateless Public Key Encryption Without Pairing. Information Security*. Springer, 2005.
34. Schnorr CP. Efficient identification and signatures for smart cards. *Conference on the Theory and Application of Cryptology*, Springer, 1989; 239–252.
35. ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. *Workshop on the Theory and Application of Cryptographic Techniques*, Springer, 1984; 10–18.
36. Fujisaki E, Okamoto T. Secure integration of asymmetric and symmetric encryption schemes. *Annual International Cryptology Conference*, Springer, 1999; 537–554.
37. Lai J, Kou W, Chen K. Self-generated-certificate public key encryption without pairing and its application. *Information Sciences* 2011; **181**(11): 2422–2435.
38. Shao Z. Enhanced certificate-based encryption from pairings. *Computers & Electrical Engineering* 2011; **37**(2): 136–146.
39. Jurišic A, Menezes A. Elliptic curves and cryptography. *Dr. Dobb's Journal* 1997: 26–36.