# Provably Secure Certificate-Based Signcryption Scheme without Pairings

**Yang Lu and Jiguo Li**
College of Computer and Information Engineering, Hohai University
Nanjing, Jiangsu 211100 - China
[e-mail: luyangnsd@163.com, lijiguo@hhu.edu.cn]
*Corresponding author: Yang Lu

*Received December 24, 2013; revised April 8, 2014; accepted May 15, 2014; published July 29, 2014*

## Abstract

Certificate-based cryptography is a new cryptographic paradigm that provides an interesting balance between identity-based cryptography and traditional public key cryptography. It not only simplifies the complicated certificate management problem in traditional public key cryptography, but also eliminates the key escrow problem in identity-based cryptography. As an extension of the signcryption in certificate-based cryptography, certificate-based signcryption provides the functionalities of certificate-based encryption and certificate-based signature simultaneously. However, to the best of our knowledge, all constructions of certificate-based signcryption in the literature so far have to be based on the costly bilinear pairings. In this paper, we propose a certificate-based signcryption scheme that does not depend on the bilinear pairings. The proposed scheme is provably secure in the random oracle model. Due to avoiding the computationally-heavy paring operations, the proposed scheme significantly reduces the cost of computation and outperforms the previous certificate-based signcryption schemes.

***Keywords:*** Certificate-based cryptography, signcryption, bilinear pairing, random oracle model

This research was supported by the National Natural Science Foundation of China [61272542]. We would like to thank the anonymous referees for their helpful comments.

**http://dx.doi.org/10.3837/tiis.2014.07.020**

## 1. Introduction

In public-key cryptography, each user has a public key and a private key. The public key is published and publicly accessible while the corresponding private key is kept secret by its owner. In traditional public key cryptography, each public key is generated with no connection to the identity of its owner. Therefore, a public key infrastructure (PKI) is employed for vouching the relationship between a user's identity and a public key by certificates. However, the traditional PKI technology is faced with many challenges in practice, especially the complicated certificate management problem. In 1984, Shamir [1] introduced the concept of identity-based cryptography. In identity-based cryptography, a user's public key could be an arbitrary string related to his identity and his private key is computed from his identity by a trusted authority called private key generator (PKG). The biggest merit of identity-based cryptography is that it eliminates the need for public key certificates. However, identity-based cryptography inevitably suffers from the key escrow problem since all the users' private keys are known to the PKG.

In order to fill the gap between traditional public key cryptography and identity-based cryptography, Al-Riyami and Paterson [2] proposed the notion of certificateless public key cryptography in Asiacrypt 2003. In certificateless public key cryptography, a trusted third party called key generation center (KGC) is employed for generating a partial private key for each user. Each user independently generates a pair of secret key and public key, and then combines his own secret key with the partial private key from the KGC to generate his full private key. Since KGC does not know any user's private key, certificateless public key cryptography overcomes the key escrow problem. However, as partial private keys should be sent from KGC to users over secure channels, certificateless public key cryptography suffers from the key distribution problem.

In Eurocrypt 2003, Gentry [3] introduced another new paradigm called certificate-based cryptography that represents an interesting balance between identity-based cryptography and traditional public key cryptography. As in traditional public key cryptography, each user in certificate-based cryptography generates a pair of public key and private key, and then requests a certificate from a trusted third party called certifier. The difference is that certificate-based cryptography provides an effective implicit certificate mechanism so that a user needs both his private key and certificate to perform cryptographic operations (such as decryption and signing), while the other communication parties need not obtain the fresh information on this user's certificate status. As a result, certificate-based cryptography eliminates the third-party queries for the certificate status and simplifies the certificate revocation problem in traditional PKI. Furthermore, since the certifier does not know any user's private key and the certificates can be sent to their owners publicly, certificate-based cryptography overcomes both the key escrow and distribution problems.

Since its advent, certificate-based cryptography has attracted great interest in the research community and many schemes have been proposed, including many encryption schemes (*e.g.* [4-10]) and signature schemes (*e.g.* [11-16]). As an extension of the signcryption [17] in the certificate-based setting, Li *et al.* [18] introduced the concept of certificate-based signcryption, which simultaneously provides the functionalities of certificate-based encryption and certificate-based signature. To the best of our knowledge, there exist three certificate-based signcryption schemes in the literature so far. All these schemes are based on the bilinear pairings. In [18], Li *et al.* proposed the first certificate-based signcryption scheme based on

Chen and Malone-Lee's identity-based signcryption scheme [19]. A subsequent paper by Luo *et al.* [20] proposed a new certificate-based signcryption scheme alone with a security model. However, Luo *et al.* only partly proved the security of their scheme in the random oracle model [21]. Recently, Li *et al.* [22] proposed a publicly verifiable certificate-based signcryption scheme which is provably secure in the random oracle model.

In this paper, we focus on the construction of certificate-based signcryption that does not depend on the costly bilinear pairings. As we know, compared with other common cryptographic operations such as prime modular exponentiations in finite fields, the bilinear pairings may be the most expensive ones. As pairing operations will greatly aggravate the computation load of a device, they are extremely disliked by the power-constrained devices, such as wireless sensors, mobile intelligent terminals, etc. Therefore, it is interesting and worthwhile to construct cryptographic schemes without relying on bilinear parings. Based on the Schnorr signature scheme [23, 24] and the enhanced ElGamal public key encryption scheme proposed by Fujisaki and Okamoto [25], we develop a pairing-free certificate-based signcryption scheme. In the random oracle model, we prove that the proposed certificate-based signcryption scheme has chosen-ciphertext security under the gap Diffie-Hellman assumption and unforgeability security under the gap discrete logarithm assumption. Without pairings, our scheme significantly reduces the cost of computation and is more efficient than the previous pairing-based certificate-based signcryption schemes. This interesting property makes it be particularly suitable for the computation-limited environments, such as wireless sensor networks and mobile wireless networks.

In Section 2, we review some computational assumptions related to our paper. In Section 3, we present the definition and security model of certificate-based signcryption. The proposed certificate-based signcryption scheme is described in Section 4 and analyzed in Section 5 respectively. Finally, we draw our conclusion in Section 6.

## 2. Computational Assumptions

Let $k$ be a security parameter and $p$ be a $k$-bit prime number. Let $G$ denote a cyclic group of prime order $p$, and $g$ a generator of the group $G$. Below, we review the computational assumptions that are used to prove the security of our proposed certificate-based signcryption scheme.

**Definition 1 [26].** The gap Diffie-Hellman (GDH) problem in $G$ is, given a tuple $(g, g^a, g^b)$ for unknown $a, b \in Z_p^*$ and access to a decisional Diffie-Hellman (DDH) oracle $O^{DDH}$ that takes $(g, g^u, g^v, z)$ as input and outputs 1 if $z = g^{uv}$ and 0 otherwise, to compute $g^{ab}$. The advantage of any probabilistic polynomial-time algorithm $A_{GDH}$ in solving the GDH problem in $G$ is defined as

$$Adv(A_{GDH}) = \Pr\{A_{GDH}(G, p, g, g^a, g^b, O^{DDH}) = g^{ab} \mid a, b \in Z_p^*\}. \quad (1)$$

The GDH assumption is that, for any probabilistic polynomial-time algorithm $A_{GDH}$, the advantage $Adv(A_{GDH})$ is negligible.

**Definition 2 [27].** The gap Discrete Logarithm (GDL) problem in $G$ is, given a tuple $(g, g^a)$ for unknown $a \in Z_p^*$ and access to a restricted DDH oracle $O^{rDDH}$ that takes $(g, g^a, g^b, z)$ as input and outputs 1 if $z = g^{ab}$ and 0 otherwise, to compute $a$. The advantage of any probabilistic polynomial-time algorithm $A_{GDL}$ in solving the GDL problem in $G$ is defined as

$$Adv(A_{GDL}) = \Pr\{A_{GDL}(G, p, g, g^a, O^{rDDH}) = a \mid a \in Z_p^*\}. \quad (2)$$

The GDL assumption is that, for any probabilistic polynomial-time algorithm $A_{GDL}$, the advantage $Adv(A_{GDL})$ is negligible.

## 3. Definition and Security Model of Certificate-Based Signcryption

In this paper, a certificate-based signcryption scheme is composed of the following five algorithms: (1) System setup algorithm ***Setup***, which is performed by a certifier to generate a master key and a list of public system parameters; (2) Key-pair generation algorithm ***KeyPairGen***, which is performed by the user to generate a pair of private key and partial public key; (3) Certification algorithm ***Certify***, which is performed by a certifier to generate a certificate and public key for each user in the system; (4) Signcryption algorithm ***Signcrypt***, which is performed by a sender to signcrypt the messages; (5) Designcryption algorithm ***Designcrypt***, which is performed by a receiver to designcrypt the ciphertext sent to him.

**Fig. 1** gives the functional description of a certificate-based signcryption scheme.

(1) ***Setup***$(k) \rightarrow (msk, params)$
Input: a security parameter $k \in Z^+$
Output: a master key $msk$ and a list of public system parameters $params$
(2) ***KeyPairGen***$(params) \rightarrow (SK_U, PPK_U)$
Input: $params$
Output: a private key $SK_U$ and a partial public key $PPK_U$ for a user with identity $id_U$
(3) ***Certify***$(params, msk, id_U, PPK_U) \rightarrow (PK_U, Cert_U)$
Input: $params$, $msk$, a user's identity $id_U$ and the partial public key $PPK_U$
Output: a full public key $PK_U$ and a certificate $Cert_U$ for the user $id_U$
(4) ***Signcrypt***$(params, m, id_S, PK_S, SK_S, Cert_S, id_R, PK_R) \rightarrow \sigma$
Input: $params$, a message $m$, the sender's identity $id_S$, public key $PK_S$, private key $SK_S$ and certificate $Cert_S$, and the receiver's identity $id_R$ and public key $PK_R$
Output: a ciphertext $\sigma$
(5) ***Designcrypt***$(params, \sigma, id_R, SK_R, Cert_R, id_S, PK_S) \rightarrow m$
Input: $params$, a ciphertext $\sigma$, the receiver's identity $id_R$, private key $SK_R$ and certificate $Cert_R$, and the sender's identity $id_S$ and public key $PK_S$
Output: a message $m$ or an error symbol $\perp$ if $\sigma$ is an invalid ciphertext

**Fig. 1.** Functional description of certificate-based signcryption

**Definition 3.** A certificate-based signcryption scheme $\Pi$ = (***Setup***, ***KeyPairGen***, ***Certify***, ***Signcrypt***, ***Designcrypt***) is said to be correct if for any message $m$, $m$ = ***Designcrypt***($params$, ***Signcrypt***($params$, $m$, $id_S$, $PK_S$, $SK_S$, $Cert_S$, $id_R$, $PK_R$), $id_R$, $SK_R$, $Cert_R$, $id_S$, $PK_S$), where $params$ are obtained from the system setup algorithm ***Setup***, ($SK_S$, $PK_S$, $Cert_S$) and ($SK_R$, $PK_R$, $Cert_R$) are respectively generated according to the the specifications of the key-pair generation algorithm ***KeyPairGen*** and the certification algorithm ***Certify***.

As introduced in [20, 22], the security model for certificate-based signcryption includes two types of adversaries: Type-I and Type-II. A Type-I adversary (denoted by $A_I$) simulates a user who wants to gain some information about a message sent to him from its encryption without a certificate. A Type-II adversary (denoted by $A_{II}$) simulates a malicious certifier in possession of the master key who wants to attack a target user without the knowledge of this user's private key.

To simulate potential attacking scenarios, we will use the following six oracles which can be accessed by the adversaries. We assume that the game simulator keeps a history of "query-answer" while interacting with the adversaries. The six oracles are described as follows:

(1) $O^{CreateUser\text{-}I}(id_U)$: This oracle is only queried by the Type-I adversary $A_I$. On input an identity $id_U$, if $id_U$ has already been created, the game simulator responds with the public key $PK_U$ associated with the identity $id_U$. Otherwise, the game simulator generates a set of private key $SK_U$, public key $PK_U$ and certificate $Cert_U$ for the identity $id_U$, and then returns $PK_U$ as the output. In this case, $id_U$ is said to be created. For simplicity, we assume that other oracles only respond to an identity which has been created.

(2) $O^{CreateUser\text{-}II}(id_U)$: This oracle is only queried by the Type-II adversary $A_{II}$. On input an identity $id_U$, if $id_U$ has already been created, the game simulator responds with the public key $PK_U$ associated with the identity $id_U$. Otherwise, the game simulator first generates a private key $SK_U$ and a partial public key $PPK_U$ for the identity $id_U$, and then outputs $PPK_U$ to $A_{II}$. Different from $O^{CreateUser\text{-}I}$, this oracle requires $A_{II}$ to help the game simulator to create a new user. As $A_{II}$ simulates a malicious certifier who will generate a full public key and a certificate for any user by itself, it is possible that the game simulator is not aware of the secret(s) used by $A_{II}$ to generate the public key and the certificate of a user. Therefore, when creating a new user, $A_{II}$ should submit the secret(s) to the game simulator. Under the help of $A_{II}$, the game simulator generates a full public key $PK_U$ and a certificate $Cert_U$ for the identity $id_U$. In this case, $id_U$ is said to be created. Similarly, we assume that other oracles only respond to an identity which has been created.

(3) $O^{RequestPrivateKey}(id_U)$: On input an identity $id_U$, the game simulator outputs the private key $SK_U$ associated with the identity $id_U$.

(4) $O^{RequestCertificate}(id_U)$: On input an identity $id_U$, the game simulator outputs the certificate $Cert_U$ associated with the identity $id_U$.

(5) $O^{Signcrypt}(m, id_S, id_R)$: On input a message $m$, a sender's identity $id_S$ and a receiver's identity $id_R$, the game simulator responds with the result of ***Signcrypt***$(params, m, id_S, PK_S, SK_S, Cert_S, id_R, PK_R)$. Note that we disallow queries where $id_S = id_R$.

(6) $O^{Designcrypt}(\sigma, id_S, id_R)$: On input a ciphertext $\sigma$, a sender's identity $id_S$ and a receiver's identity $id_R$, the challenger responds with the result of ***Designcrypt***$(params, \sigma, id_R, SK_R, Cert_R, id_S, PK_S)$. Again, we disallow queries where $id_S = id_R$.

A certificate-based signcryption scheme should satisfy both confidentiality (indistinguishability against adaptive chosen-ciphertext attacks (IND-CBSC-CCA2)) and unforgeability (existential unforgeability against adaptive chosen-messages attacks (EUF-CBSC-CMA)).

For the confidentiality, we consider the following two different adversarial games "IND-CBSC-CCA2 Game-I" and "IND-CBSC-CCA2 Game-II". IND-CBSC-CCA2 Game-I is the game played between the Type-I adversary $A_I$ and a game simulator, in which *state* represents some state information, *Oracles-I* means that the adversary $A_I$ can adaptively query the oracles $\{O^{CreateUser\text{-}I}, O^{RequestPrivateKey}, O^{RequestCertificate}, O^{Signcrypt}, O^{Designcrypt}\}$ with the following constraints: (1) The identity $id_R^*$ cannot be submitted to the oracle $O^{RequestCertificate}$; (2) $(\sigma^*, id_S^*, id_R^*)$ cannot be submitted to the oracle $O^{Designcrypt}$. IND-CBSC-CCA2 Game-II is the game played between the Type-II adversary $A_{II}$ and a game simulator, in which *state* represents some state information, *Oracles-II* means that the adversary $A_{II}$ can adaptively query the oracles $\{O^{CreateUser\text{-}II}, O^{RequestPrivateKey}, O^{Signcrypt}, O^{Designcrypt}\}$ with the following

constraints: (1) The identity $id_R^*$ cannot be submitted to the oracle $O^{RequestPrivateKey}$; (2) $(\sigma^*, id_S^*, id_R^*)$ cannot be submitted to the oracle $O^{Designcrypt}$.

**IND-CBSC-CCA2 Game-I:**

1. $(params, msk) \leftarrow_R \textbf{\textit{Setup}}(k)$
2. $(m_0, m_1, id_S^*, id_R^*, state) \leftarrow_R A_I^{Oracles-I}(k, params)$
3. $\sigma^* \leftarrow_R \textbf{\textit{Signcrypt}}(params, m_b, id_S^*, PK_S^*, SK_S^*, id_R^*, PK_R^*)$
4. $b' \leftarrow_R A_I^{Oracles-I}(\sigma^*, state)$
5. Output $b'$

**IND-CBSC-CCA2 Game-II:**

1. $(params, msk) \leftarrow_R \textbf{\textit{Setup}}(k)$
2. $(m_0, m_1, id_S^*, id_R^*, state) \leftarrow_R A_{II}^{Oracles-II}(k, params, msk)$
3. $\sigma^* \leftarrow_R \textbf{\textit{Signcrypt}}(params, m_b, id_S^*, PK_S^*, SK_S^*, id_R^*, PK_R^*)$
4. $b' \leftarrow_R A_{II}^{Oracles-II}(\sigma^*, state)$
5. Output $b'$

In both two games, we say that an adversary wins the game if $b = b'$. The adversary's advantage in winning the game is defined to be $Adv(A_X) = 2|\Pr[b = b'] - 1/2|$, where $X$ is either I or II.

**Definition 4.** A certificate-based signcryption scheme is said to be IND-CBSC-CCA2 secure if no probabilistic polynomial-time adversary has non-negligible advantage in both the games IND-CBSC-CCA2 Game-I and IND-CBSC-CCA2 Game-II.

For the unforgeability, we consider the following two different adversarial games "EUF-CBSC-CMA Game-I" and "EUF-CBSC-CMA Game-II". EUF-CBSC-CMA Game-I is the game played between the Type-I adversary $A_I$ and a game simulator, in which *Oracles-III* means that the adversary $A_I$ can adaptively query the oracles $\{O^{CreateUser-I}, O^{RequestPrivateKey}, O^{RequestCertificate}, O^{Signcrypt}, O^{Designcrypt}\}$ with the following constraints: (1) The identity $id_S^*$ cannot be submitted to the oracle $O^{RequestCertificate}$; (2) $(\sigma^*, id_S^*, id_R^*)$ is not produced by the oracle $O^{Signcrypt}$. EUF-CBSC-CMA Game-II is the game played between the Type-II adversary $A_{II}$ and a game simulator, in which *Oracles-IV* means that the adversary $A_{II}$ can adaptively query the oracles $\{O^{CreateUser-II}, O^{RequestPrivateKey}, O^{Signcrypt}, O^{Designcrypt}\}$ with the following constraints: (1) The identity $id_S^*$ cannot be submitted to the oracle $O^{RequestPrivateKey}$; (2) $(\sigma^*, id_S^*, id_R^*)$ is not produced by the oracle $O^{Signcrypt}$.

**EUF-CBSC-CMA Game-I:**

1. $(params, msk) \leftarrow_R \textbf{\textit{Setup}}(k)$
2. $(\sigma^*, id_S^*, id_R^*) \leftarrow_R A_I^{Oracles-III}(k, params)$
3. Output $(\sigma^*, id_S^*, id_R^*)$

**EUF-CBSC-CMA Game-II:**

1. $(params, msk) \leftarrow_R \textbf{\textit{Setup}}(k)$
2. $(\sigma^*, id_S^*, id_R^*) \leftarrow_R A_{II}^{Oracles-IV}(k, params, msk)$
3. Output $(\sigma^*, id_S^*, id_R^*)$

In both two games, we say that an adversary wins the game if it outputs a valid forgery $(\sigma^*, id_S^*, id_R^*)$, namely that the result of ***Designcrypt***$(params, \sigma^*, id_R^*, SK_R^*, Cert_R^*, id_S^*, PK_S^*)$ is not the symbol $\perp$. The adversary's advantage is defined to be the probability that it wins the game.

**Definition 5.** A certificate-based signcryption scheme is said to be EUF-CBSC-CMA secure if no probabilistic polynomial-time adversary has non-negligible advantage in both the adversarial games EUF-CBSC-CMA Game-I and EUF-CBSC-CMA Game-II.

## 4. Description of the Proposed Certificate-Based Signcryption Scheme

We now present the proposed certificate-based signcryption scheme. As mentioned previously, our scheme is based on the Schnorr signature scheme [23, 24] and the enhanced ElGamal public key encryption scheme proposed by Fujisaki and Okamoto [25]. A formal description of the scheme is as follows:

(1) ***Setup***$(k)$: The certifier performs as follows: choose a group $G$ of $k$-bit prime order $p$ and a random generator $g \in G$; choose a random value $\alpha \in Z_p^*$ and compute $g_1 = g^\alpha$; choose three cryptographic hash functions $H_1$: $\{0,1\}^* \times G \times G \to Z_p^*$, $H_2$: $\{0,1\}^{l_m} \times G \times \{0,1\}^* \times G \times G \to Z_p^*$ and $H_3$: $G \to \{0,1\}^{l_m}$, where $l_m$ denotes the bit-length of a plaintext; output a list of public parameters $params = \{G, p, g, g_1, l_m, H_1, H_2, H_3\}$ and a master key $msk = \alpha$.

(2) ***KeyPairGen***$(params)$: A user with identity $id_U$ chooses a random value $x \in Z_p^*$ as his private key $SK_U$ and computes his partial public key $PPK_U = g^x$.

(3) ***Certify***$(params, msk, id_U, PPK_U)$: The certifier performs as follows: set $PK_U^{(1)} = PPK_U$; choose a random value $y \in Z_p^*$ and compute $PK_U^{(2)} = g^y$; compute $Cert_U = y + \alpha H_1(id_U, PK_U^{(1)}, PK_U^{(2)})$; output the user $id_U$'s public key $PK_U = (PK_U^{(1)}, PK_U^{(2)})$ and certificate $Cert_U$.

(4) ***Signcrypt***$(params, m, id_S, PK_S, SK_S, Cert_S, id_R, PK_R)$: To send a message $m \in \{0,1\}^{l_m}$ to a user $id_R$, the sender $id_S$ does the following: choose a random value $r \in Z_p^*$ and compute $R = g^r$; compute $u = r(SK_S + Cert_S + h)^{-1}$, where $h = H_2(m, R, id_S, PK_S^{(1)}, PK_S^{(2)})$; compute $v = (PK_R^{(1)} \cdot PK_R^{(2)} \cdot g_1^{H_1(id_R, PK_R^{(1)}, PK_R^{(2)})})^r$ and then $c = m \oplus H_3(v)$; output the ciphertext $\sigma = (h, u, c)$.

(5) ***Designcrypt***$(params, \sigma, id_R, SK_R, Cert_R, id_S, PK_S)$: To designcrypt a ciphertext $\sigma$ from the sender $id_S$, the receiver $id_R$ does the following: parse the ciphertext $\sigma$ as $(h, u, c)$; compute $R' = (PK_S^{(1)} \cdot PK_S^{(2)} \cdot g_1^{H_1(id_S, PK_S^{(1)}, PK_S^{(2)})} \cdot g^h)^u$ and then $v' = (R')^{SK_R + Cert_R}$; compute $m' = c \oplus H_3(v')$; check whether $h = H_2(m', R', id_S, PK_S^{(1)}, PK_S^{(2)})$. If it does, output the message $m'$, otherwise output an invalid symbol $\perp$.

## 5. Analysis of the Proposed Certificate-Based Signcryption Scheme

### 5.1 Correctness

**Theorem 1.** The proposed certificate-based signcryption scheme is correct.

Proof. This theorem can be proved by the following equations:

$$R' = (PK_S^{(1)} \cdot PK_S^{(2)} \cdot g_1^{H_1(id_S, PK_S^{(1)}, PK_S^{(2)})} \cdot g^h)^u = (g^{SK_S + Cert_S + h})^{r(SK_S + Cert_S + h)^{-1}} = g^r = R,$$

$$v' = (R')^{SK_R + Cert_R} = g^{r(SK_R + Cert_R)} = (PK_R^{(1)} \cdot PK_R^{(2)} \cdot g_1^{H_1(id_R, PK_R^{(1)}, PK_R^{(2)})})^r = v.$$

## 5.2 Security

We show that our scheme achieves the IND-CBSC-CCA2 security under the GDH assumption and the EUF-CBSC-CMA security under the GDL assumption in the random oracle model.

**Theorem 2.** In the random oracle model, the proposed certificate-based signcryption scheme is IND-CBSC-CCA2 secure under the GDH assumption.

This theorem can be proved by combining the following two lemmas.

**Lemma 1.** Suppose that $H_1 \sim H_3$ are random oracles and $A_I$ is a Type-I adversary against the IND-CBCS-CCA2 security of the proposed scheme with advantage $\varepsilon$ and running time $\tau$. Assume also that $A_I$ makes at most $q_{cu}$ queries to the oracle $O^{CreateUser\text{-}I}$, $q_{pri}$ queries to the oracle $O^{RequestPrivateKey}$, $q_{cer}$ queries to the oracle $O^{RequestCertificate}$, $q_{sc}$ queries to the oracle $O^{Signcrypt}$, $q_{dsc}$ queries to the oracle $O^{Designcrypt}$ and $q_i$ queries to the random oracles $H_i$ ($1 \le i \le 3$) respectively. Then there exists an algorithm $A_{GDH}$ to solve the GDH problem in the group $G$ with advantage

$$\varepsilon' \ge \frac{\varepsilon}{q_{cu}(q_3 + q_{sc})}(1 - q_{sc}\frac{q_2 + q_3 + 2q_{sc}}{2^k})(1 - \frac{q_{dsc}}{2^k}) \quad (3)$$

and running time $\tau' \le \tau + (q_1 + q_2 + q_3 + q_{cer} + q_{pri})O(1) + q_{cu}(3\tau_{exp} + O(1)) + q_{sc}(4\tau_{exp} + O(1)) + q_{dsc}(4\tau_{exp} + \tau_{DDH} + O(1))$, where $\tau_{exp}$ and $\tau_{DDH}$ respectively denote the time for computing an exponentiation in $G$ and the one for a call to the DDH oracle.

Proof. Assume that the algorithm $A_{GDH}$ is given a random GDH problem instance ($G$, $p$, $g$, $g^a$, $g^b$, $O^{DDH}$). Its goal is to compute $g^{ab}$ by interacting with $A_I$ as follows:

At the beginning of the game, the algorithm $A_{GDH}$ randomly chooses an index $\theta \in [1, q_{cu}]$ and sets $g_1 = g^a$. It then starts IND-CBCS-CCA2 Game-I by supplying the adversary $A_I$ with the public parameters $params = \{G, p, g, g_1, l_m, H_1, H_2, H_3\}$, where $H_1 \sim H_3$ are random oracles controlled by $A_{GDH}$. Note that the certifier's master key is the value $a$ which is unknown to the algorithm $A_{GDH}$.

During the query-answer phase, the adversary $A_I$ can adaptively make queries to the oracles $H_1$, $H_2$, $H_3$, $O^{CreateUser\text{-}I}$, $O^{RequestPrivateKey}$, $O^{RequestCertificate}$, $O^{Signcrypt}$ and $O^{Designcrypt}$. The algorithm $A_{GDH}$ responds as follows:

*$H_1$ queries*: $A_{GDH}$ maintains a list $H_1$List of tuples $<id_i, PK_i^{(1)}, PK_i^{(2)}, h_1>$. On receiving such a query on $(id_i, PK_i^{(1)}, PK_i^{(2)})$, $A_{GDH}$ first checks if $H_1$List contains a tuple $<id_i, PK_i^{(1)}, PK_i^{(2)}, h_1>$. If it does, $A_{GDH}$ outputs $h_1$ to $A_I$ directly. Otherwise, it outputs a random value $h_1 \in Z_p^*$ to $A_I$ and inserts a new tuple $<id_i, PK_i^{(1)}, PK_i^{(2)}, h_1>$ into $H_1$List.

*$H_2$ queries*: $A_{GDH}$ maintains a list $H_2$List of tuples $<m, R, id_i, PK_i^{(1)}, PK_i^{(2)}, h_2>$. On receiving such a query on $(m, R, id_i, PK_i^{(1)}, PK_i^{(2)})$, $A_{GDH}$ first checks if $H_2$List contains a tuple $<m, R, id_i, PK_i^{(1)}, PK_i^{(2)}, h_2>$. If it does, $A_{GDH}$ outputs $h_2$ to $A_I$ directly. Otherwise, it outputs a random value $h_2 \in Z_p^*$ to $A_I$ and inserts a new tuple $<m, R, id_i, PK_i^{(1)}, PK_i^{(2)}, h_2>$ into $H_2$List.

*$H_3$ queries*: $A_{GDH}$ maintains a list $H_3$List of tuples $<v, h_3>$. On receiving such a query on $v$, $A_{GDH}$ first checks if $H_3$List contains a tuple $<v, h_3>$. If it does, $A_{GDH}$ outputs $h_3$ to $A_I$ directly. Otherwise, it outputs a random value $v \in \{0,1\}^{l_m}$ to $A_I$ and inserts a new tuple $<v, h_3>$ into $H_3$List.

$O^{CreateUser\text{-}I}$ *queries*: $A_{GDH}$ maintains a list UserList of tuples $<id_i, SK_i, PK_i, Cert_i>$. On receiving such a query on $id_i$, $A_{GDH}$ performs as follows: (1) If UserList contains a tuple $<id_i, SK_i, PK_i, Cert_i>$, output $PK_i$ to $A_I$ directly; (2) Otherwise, if $id_i$ is the $\theta$-th distinct identity submitted to this oracle, choose two random values $x_\theta, y_\theta \in Z_p^*$, set $SK_\theta = x_\theta$ and compute $PK_\theta = (g^{x_\theta}, g^{y_\theta})$, insert a new tuple $<id_\theta, SK_\theta, PK_\theta, \perp>$ into UserList and output $PK_\theta$ to $A_I$; (3) Otherwise, randomly choose $x_i, t_i, e_i \in Z_p^*$, set $SK_i = x_i$ and $Cert_i = t_i$, compute $PK_i = (PK_i^{(1)}, PK_i^{(2)}) = (g^{x_i}, g^{t_i} g_1^{-e_i})$, insert $<id_i, PK_i^{(1)}, PK_i^{(2)}, e_i>$ and $<id_i, SK_i, PK_i, Cert_i>$ into $H_1$List and UserList respectively, and output $PK_i$ to $A_I$.

$O^{RequestPrivateKey}$ *queries*: On receiving such a query on $id_i$, $A_{GDH}$ retrieves a tuple of the form $<id_i, SK_i, PK_i, Cert_i>$ from the list UserList and returns $SK_i$ to $A_I$.

$O^{RequestCertificate}$ *queries*: On receiving such a query on $id_i$, $A_{GDH}$ aborts if $id_i = id_\theta$. Otherwise, it retrieves a tuple of the form $<id_i, SK_i, PK_i, Cert_i>$ from UserList and returns $Cert_i$ to $A_I$.

$O^{Signcrypt}$ *queries*: On receiving such a query on $(m, id_S, id_R)$, $A_{GDH}$ does the following: (1) If $id_S = id_\theta$, $A_{GDH}$ randomly chooses $u, h_2 \in Z_p^*$, $h_3 \in \{0,1\}^{l_m}$, runs the simulation algorithm for the random oracle $H_1$ to get $h_1 = H_1(id_S, PK_S^{(1)}, PK_S^{(2)})$, and computes $R = (PK_{id_\theta}^{(1)} \cdot PK_{id_\theta}^{(2)} \cdot g_1^{h_1} \cdot g^{h_2})^u$, $v = R^{SK_R + Cert_R}$, $c = m \oplus h_3$. Then, $A_{GDH}$ inserts $<m, R, id_\theta, PK_\theta^{(1)}, PK_\theta^{(2)}, h_2>$ and $<v, h_3>$ into $H_2$List and $H_3$List respectively and returns $\sigma = (h_2, u, c)$ as the ciphertext to $A_I$. Note that $A_{GDH}$ fails if $H_2$List or $H_3$List is already defined in the corresponding value but this only happens with probability smaller than $(q_2 + q_3 + 2q_{sc})/2^k$. (2) Otherwise, $A_{GDH}$ can answer the query according to the specification of the algorithm *Signcrypt* since it knows the sender $id_S$'s private key and certificate.

$O^{Designcrypt}$ *queries*: On receiving such a query on $(\sigma = (h, u, c), id_S, id_R)$, $A_{GDH}$ does the following: (1) If $id_R = id_\theta$, $A_{GDH}$ first runs the simulation algorithm for the random oracle $H_1$ to get $h_1 = H_1(id_R, PK_R^{(1)}, PK_R^{(2)})$ and $h_1' = H_1(id_S, PK_S^{(1)}, PK_S^{(2)})$, and then checks if there exist a tuple $<m, R, id_S, PK_S^{(1)}, PK_S^{(2)}, h>$ in $H_2$List and a tuple $<v, h_3>$ in $H_3$List such that $R = (PK_S^{(1)} \cdot PK_S^{(2)} \cdot g_1^{h_1'} \cdot g^h)^u$, $m = c \oplus h_3$ and $O^{DDH}(g, R, PK_R^{(1)} \cdot PK_R^{(2)} \cdot g_1^{h_1}, v) = 1$. If such two tuples exist, $A_{GDH}$ returns $m$ to $A_I$ as the designcryption of $\sigma$; otherwise, it rejects $\sigma$. Note that a valid ciphertext is rejected with probability smaller than $q_{dsc}/2^k$ across the whole game. (2) Otherwise, $A_{GDH}$ designcrypts $\sigma$ in the normal way since it knows the receiver $id_R$'s private key and certificate.

At the challenge phase, $A_I$ outputs two distinct messages $m_0$ and $m_1$ of equal length, a sender identity $id_S^*$ and a receiver identity $id_R^*$, on which it wants to be challenged. If $id_R^* \neq id_\theta$, then $A_{GDH}$ aborts. Otherwise, $A_{GDH}$ randomly chooses $c^* \in \{0,1\}^{l_m}$ and $u^*, h_2^* \in Z_p^*$, sets $R^* = g^b$, and outputs $\sigma^* = (h_2^*, u^*, c^*)$ as the challenge ciphertext to $A_I$. Observe that the decryption of $c^*$ is $c^* \oplus H_3((PK_\theta^{(1)} \cdot PK_\theta^{(2)} \cdot g_1^{H_1(id_\theta, PK_\theta^{(1)}, PK_\theta^{(2)})})^b)$.

At the guess phase, $A_I$ outputs a bit which is ignored by $A_{GDH}$. Note that $A_I$ cannot recognize that the challenge ciphertext $\sigma^*$ is an invalid ciphertext unless it queries $H_3((PK_\theta^{(1)} \cdot PK_\theta^{(2)} \cdot g_1^{H_1(id_\theta, PK_\theta^{(1)}, PK_\theta^{(2)})})^b)$. It is clear that a successful $A_I$ is very likely to query $H_3((PK_\theta^{(1)} \cdot PK_\theta^{(2)} \cdot g_1^{H_1(id_\theta, PK_\theta^{(1)}, PK_\theta^{(2)})})^b)$ if the simulation is indistinguishable from a real attack environment. To produce a result, $A_{GDH}$ randomly chooses a tuple $<v, h_3>$ from $H_3$List and outputs

$$T = \left[ \frac{v}{(R^*)^{x_\theta} \cdot (R^*)^{y_\theta}} \right]^{\frac{1}{H_1(id_\theta, PK_\theta^{(1)}, PK_\theta^{(2)})}} \tag{4}$$

as the solution to the given GDH problem. Obviously, if $v = (PK_\theta^{(1)} \cdot PK_\theta^{(2)} \cdot g_1^{H_1(id_\theta, PK_\theta^{(1)}, PK_\theta^{(2)})})^b$, then we have $T = g^{ab}$.

This completes the simulation. We now estimate the advantage of $A_{GDH}$ in solving the given GDH problem. From the above construction, the simulation fails if any of the following events occurs: (1) $E_1$: $A_I$ does not choose $id_\theta$ as the challenge receiver identity $id_R^*$; (2) $E_2$: $A_I$ queries $O^{RequestCertificate}$ on the identity $id_\theta$; (3) $E_3$: $A_{GDH}$ aborts in answer $A_I$'s query to $O^{Signcrypt}$ because of a collision on $H_2$ or $H_3$; (4) $E_4$: $A_{GDH}$ rejects a valid ciphertext at some point of the game. We clearly have that $\Pr[\neg E_1] = 1/q_{cu}$ and $\neg E_1$ implies $\neg E_2$. We also already observed that $\Pr[E_3] \le q_{sc}(q_2 + q_3 + 2q_{sc})/2^k$ and $\Pr[E_4] \le q_{dsc}/2^k$. Thus, we have that

$$\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4] \ge \frac{1}{q_{cu}}(1 - q_{sc}\frac{q_2 + q_3 + 2q_{sc}}{2^k})(1 - \frac{q_{dsc}}{2^k}) . \tag{5}$$

Since $A_{GDH}$ selects the correct tuple from $H_3$List with probability $1/(q_3 + q_{sc})$, we have that the advantage of $A_{GDH}$ in solving the GDH problem is

$$\varepsilon' \ge \frac{\varepsilon}{q_{cu}(q_3 + q_{sc})}(1 - q_{sc}\frac{q_2 + q_3 + 2q_{sc}}{2^k})(1 - \frac{q_{dsc}}{2^k}) . \tag{6}$$

The time complexity of the algorithm $A_{GDH}$ is dominated by the exponentiations and the calls to the DDH oracle in the oracle simulations. From the above description of $A_{GDH}$, it is easy to see that the time complexity of $A_{GDH}$ is bound by $\tau' \le \tau + (q_1 + q_2 + q_3 + q_{cer} + q_{pri})O(1) + q_{cu}(3\tau_{exp} + O(1)) + q_{sc}(4\tau_{exp} + O(1)) + q_{dsc}(4\tau_{exp} + \tau_{DDH} + O(1))$ , where $\tau_{exp}$ and $\tau_{DDH}$ respectively denote the time for computing an exponentiation in $G$ and the one for a call to the DDH oracle.

**Lemma 2.** Suppose that $H_1 \sim H_3$ are random oracles and $A_{II}$ is a Type-II adversary against the IND-CBCS-CCA2 security of the proposed scheme with advantage $\varepsilon$ and running time $\tau$. Assume also that $A_{II}$ makes at most $q_{cu}$ queries to the oracle $O^{CreateUser\text{-}II}$, $q_{pri}$ queries to the oracle $O^{RequesrPrivateKey}$, $q_{sc}$ queries to the oracle $O^{Signcrypt}$, $q_{dsc}$ queries to the oracle $O^{Desigcrypt}$ and $q_i$ queries to the random oracles $H_i$ $(1 \le i \le 3)$. Then there exists an algorithm $A_{GDH}$ to solve the GDH problem in the group $G$ with advantage

$$\varepsilon' \ge \frac{\varepsilon}{q_{cu}(q_3 + q_{sc})}(1 - q_{sc}\frac{q_2 + q_3 + 2q_{sc}}{2^k})(1 - \frac{q_{dsc}}{2^k}) \tag{7}$$

and running time $\tau' \le \tau + (q_1 + q_2 + q_3 + q_{pri})O(1) + q_{cu}(2\tau_{exp} + O(1)) + q_{sc}(4\tau_{exp} + O(1)) + q_{dsc}(4\tau_{exp} + \tau_{DDH} + O(1))$, where $\tau_{exp}$ and $\tau_{DDH}$ respectively denote the time for computing an exponentiation in $G$ and the one for a call to the DDH oracle.

Proof. Assume that $A_{GDH}$ is given a random GDH problem instance $(G, p, g, g^a, g^b, O^{DDH})$. Its goal is to compute $g^{ab}$ by interacting with $A_{II}$ as follows:

At the beginning of the game, $A_{GDH}$ first randomly chooses $\alpha \in Z_p^*$ and an index $\theta \in [1, q_{cu}]$. It then computes $g_1 = g^{\alpha}$ and starts IND-CBSC-CCA2 Game-II by supplying $A_{II}$ with the master key $msk = \alpha$ and the public parameters $params = \{G, p, g, g_1, l_m, H_1, H_2, H_3\}$, where $H_1 \sim H_3$ are random oracles controlled by $A_{GDH}$.

During the question-answer phase, $A_{II}$ can adaptively make queries to the oracles $H_1$, $H_2$, $H_3$, $O^{CreateUser\text{-}II}$, $O^{RequestPrivateKey}$, $O^{Signcrypt}$ and $O^{Designcrypt}$. $A_{GDH}$ answers $A_{II}$'s queries to $H_1$, $H_2$, $H_3$, $O^{Signcrypt}$ and $O^{Designcrypt}$ as in the proof of Lemma 1 and handles other queries as follows:

$O^{CreateUser\text{-}II}$ *queries*: $A_{GDH}$ maintains a list UserList of tuples $<id_i, SK_i, PK_i, Cert_i, y_i>$. On receiving such a query on $id_i$, $A_{GDH}$ performs as follows: (1) If UserList contains a tuple $<id_i, SK_i, PK_i, Cert_i, y_i>$, output $PK_i$ to $A_{II}$ directly. (2) Otherwise, if $id_i$ is the $\theta$-th distinct identity submitted to this oracle, set $PK_{\theta}^{(1)} = g^a$ and output $PK_{\theta}^{(1)}$ to $A_{II}$. After receiving a value $y_{\theta} \in Z_p^*$ from $A_{II}$, compute $PK_{\theta}^{(2)} = g^{y_{\theta}}$, run the simulation algorithm for the random oracle $H_1$ to get a hash value $h_1 = H_1(id_{\theta}, PK_{\theta}^{(1)}, PK_{\theta}^{(2)})$, compute $Cert_{\theta} = y_{\theta} + \alpha h_1$ and insert a new tuple $<id_{\theta}, \perp, PK_{\theta}, Cert_{\theta}, y_{\theta}>$ into UserList. (3) Otherwise, randomly choose $x_i \in Z_p^*$, set $SK_i = x_i$, compute $PK_i^{(1)} = g^{x_i}$ and output $PK_i^{(1)}$ to $A_{II}$. After receiving a value $y_i \in Z_p^*$ from $A_{II}$, compute $PK_i^{(2)} = g^{y_i}$, run the simulation algorithm for the random oracle $H_1$ to get a hash value $h_1 = H_1(id_i, PK_i^{(1)}, PK_i^{(2)})$, compute $Cert_i = y_i + \alpha h_1$ and insert a new tuple $<id_i, SK_i, PK_i, Cert_i, y_i>$ into UserList.

$O^{RequestPrivateKey}$ *queries*: On receiving such a query on $id_i$, $A_{GDH}$ aborts if $id_i = id_{\theta}$. Otherwise, it retrieves a tuple of the form $<id_i, SK_i, PK_i, Cert_i, y_i>$ from UserList and returns $SK_i$ to $A_{II}$.

At the challenge phase, $A_{II}$ outputs two distinct messages $m_0$ and $m_1$ of equal length, a sender identity $id_S^*$ and a receiver identity $id_R^*$, on which it wants to be challenged. If $id_R^* \neq id_{\theta}$, $A_{GDH}$ aborts. Otherwise, $A_{GDH}$ randomly chooses $c^* \in \{0,1\}^{l_m}$ and $u^*, h_2^* \in Z_p^*$, sets $R^* = g^b$, and outputs $\sigma^* = (h_2^*, u^*, c^*)$ to $A_{II}$ as the challenge ciphertext. Observe that the decryption of $c^*$ is $c^* \oplus H_3((PK_{\theta}^{(1)} \cdot PK_{\theta}^{(2)} \cdot g_1^{H_1(id_{\theta}, PK_{\theta}^{(1)}, PK_{\theta}^{(2)})})^b)$.

At the guess phase, $A_{II}$ outputs a bit which is ignored by $A_{GDH}$. Note that $A_{II}$ cannot recognize that the challenge ciphertext $\sigma^*$ is not a valid ciphertext unless it queries $H_3((PK_{\theta}^{(1)} \cdot PK_{\theta}^{(2)} \cdot g_1^{H_1(id_{\theta}, PK_{\theta}^{(1)}, PK_{\theta}^{(2)})})^b)$. It is clear that a successful $A_{II}$ is very likely to query $H_3((PK_{\theta}^{(1)} \cdot PK_{\theta}^{(2)} \cdot g_1^{H_1(id_{\theta}, PK_{\theta}^{(1)}, PK_{\theta}^{(2)})})^b)$ if the simulation is indistinguishable from a real attack environment. To produce a result, $A_{GDH}$ randomly chooses a tuple $<v, h_3>$ from $H_3$List and outputs

$$T = \frac{v}{(R^*)^{y_{\theta}} \cdot (R^*)^{\alpha H_1(id_{\theta}, PK_{\theta}^{(1)}, PK_{\theta}^{(2)})}} \tag{8}$$

as the solution to the given GDH problem. Obviously, if $v = (PK_{\theta}^{(1)} \cdot PK_{\theta}^{(2)} \cdot g_1^{H_1(id_{\theta}, PK_{\theta}^{(1)}, PK_{\theta}^{(2)})})^b$, then we have $T = g^{ab}$.

We now estimate the advantage of $A_{GDH}$ in solving the given GDH problem.

From the above construction, the simulation fails if any of the following events occurs: (1) $E_1$: $A_{II}$ does not choose $id_{\theta}$ as the challenge receiver identity $id_R^*$; (2) $E_2$: $A_{II}$ queries $O^{RequestPrivateKey}$ on the identity $id_{\theta}$; (3) $E_3$: $A_{GDH}$ aborts in answer $A_{II}$'s $O^{Signcrypt}$ query because of

a collision on $H_2$ or $H_3$;(4) $E_4$: $A_{GDH}$ rejects a valid ciphertext at some point of the game. We clearly have that $\Pr[\neg E_1] = 1/q_{cu}$ and $\neg E_1$ implies $\neg E_2$. We also already observed that $\Pr[E_3] \leq q_{sc}(q_2 + q_3 + 2q_{sc})/2^k$ and $\Pr[E_4] \leq q_{dsc}/2^k$. Thus, we have that

$$\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4] \geq \frac{1}{q_{cu}}(1 - q_{sc}\frac{q_2 + q_3 + 2q_{sc}}{2^k})(1 - \frac{q_{dsc}}{2^k}) . \quad (9)$$

Since $A_{GDH}$ selects the correct tuple from $H_3$List with probability $1/(q_3 + q_{sc})$, the advantage of $A_{GDH}$ in solving the GDH problem is

$$\varepsilon' \geq \frac{\varepsilon}{q_{cu}(q_3 + q_{sc})}(1 - q_{sc}\frac{q_2 + q_3 + 2q_{sc}}{2^k})(1 - \frac{q_{dsc}}{2^k}) . \quad (10)$$

The time complexity of the algorithm $A_{GDH}$ is dominated by the exponentiations and the calls to the DDH oracle in the oracle simulations. From the above description of $A_{GDH}$, it is easy to see that the time complexity of $A_{GDH}$ is bound by $\tau' \leq \tau + (q_1 + q_2 + q_3 + q_{pri})O(1) + q_{cu}(2\tau_{exp} + O(1)) + q_{sc}(4\tau_{exp} + O(1)) + q_{dsc}(4\tau_{exp} + \tau_{DDH} + O(1))$, where $\tau_{exp}$ and $\tau_{DDH}$ respectively denote the time for computing an exponentiation in $G$ and the one for a call to the DDH oracle.

**Theorem 3.** In the random oracle model, the proposed certificate-based signcryption scheme is EUF-CBSC-CMA secure under the GDL assumption.

This theorem can be proved by combining the following two lemmas.

**Lemma 3.** Suppose that $H_1 \sim H_3$ are random oracles and $A_I$ is a Type-I adversary against the EUF-CBSC-CMA security of the proposed scheme who makes at most $q_{cu}$ queries to the oracle $O^{CreateUser\text{-}I}$, $q_{pri}$ queries to the oracle $O^{RequestPrivateKey}$, $q_{cer}$ queries to the oracle $O^{RequestCertificate}$, $q_{sc}$ queries to the oracle $O^{Signcrypt}$, $q_{dsc}$ queries to the oracle $O^{Designcrypt}$ and $q_i$ queries to the random oracles $H_i$ ($1 \leq i \leq 3$). Assume also that $A_I$ produces a forgery with probability $\varepsilon \geq 10(q_{sc} + 1)(q_{sc} + q_2)/2^k$ within a time $\tau$. Then there exists an algorithm $A_{GDL}$ to solve the GDL problem in the group $G$ with advantage $\varepsilon' \geq 1/9$ and running time $\tau' \leq 23q_{cu}q_2[\tau + (q_1 + q_2 + q_3 + q_{cer} + q_{pri})O(1) + q_{cu}(2\tau_{exp} + O(1)) + q_{sc}(4\tau_{exp} + O(1)) + q_{dsc}(4\tau_{exp} + \tau_{rDDH} + O(1))][\varepsilon(1 - q_{dsc}/2^k)(1 - q_{sc}(q_2 + q_3 + 2q_{sc})/2^k)]^{-1}$, where $\tau_{exp}$ and $\tau_{rDDH}$ respectively denote the time for computing an exponentiation in $G$ and the one for a call to the restricted DDH oracle.

Proof. Assume that $A_{GDL}$ is given a random GDL problem instance $(G, p, g, g^a, O^{rDDH})$. Its goal is to compute $a$ by interacting with $A_I$ as follows:

At the beginning of the game, $A_{GDL}$ first randomly chooses $\alpha \in Z_p^*$ and an index $\theta \in [1, q_{cu}]$. It then computes $g_1 = g^\alpha$ and starts EUF-CBSC-CMA Game-I by supplying $A_I$ with the public parameters $params = \{G, p, g, g_1, l_m, H_1, H_2, H_3\}$, where $H_1 \sim H_3$ are random oracles controlled by $A_{GDL}$.

During the query-answer phase, $A_I$ can adaptively make queries to the oracles $H_1$, $H_2$, $H_3$, $O^{CreateUser\text{-}I}$, $O^{RequestPrivateKey}$, $O^{RequestCertificate}$, $O^{Signcrypt}$ and $O^{Designcrypt}$. $A_{GDL}$ answers $A_I$'s queries to $H_1$, $H_2$, $H_3$, $O^{RequestPrivateKey}$, $O^{RequestCertificate}$ and $O^{Signcrypt}$ in the same way as the proof of Lemma 1 and handles other queries as follows:

$O^{CreateUser\text{-}I}$ *queries*: $A_{GDL}$ maintains a list UserList of tuples $<id_i, SK_i, PK_i, Cert_i>$. On receiving such a query on $id_i$, $A_{GDL}$ performs as follows: (1) If UserList contains a tuple $<id_i, SK_i, PK_i, Cert_i>$, output $PK_i$ to $A_I$. (2) Otherwise, if $id_i$ is the $\theta$-th distinct identity submitted to this oracle, randomly choose $x_\theta \in Z_p^*$, set $SK_\theta = x_\theta$ and $PK_\theta = (g^{x_\theta}, g^a)$, insert $<id_\theta, SK_\theta, PK_\theta,$

$\perp$> into UserList and output $PK_\theta$ to $A_I$. (3) Otherwise, randomly choose $x_i, y_i \in Z_p^*$, set $SK_i = x_i$ and $PK_i = (PK_i^{(1)}, PK_i^{(2)}) = (g^{x_i}, g^{y_i})$; run the simulation algorithm for the random oracle $H_1$ to get a hash value $h_1 = H_1(id_i, PK_i^{(1)}, PK_i^{(2)})$ and compute $Cert_i = y_i + \alpha h_1$; insert <$id_i$, $SK_i$, $PK_i$, $Cert_i$> into UserList and output $PK_i$ to $A_I$.

$O^{Designcrypt}$ *queries*: On receiving such a query on ($\sigma$ = ($h$, $u$, $c$), $id_S$, $id_R$), $A_{GDL}$ does the following: (1) If $id_R = id_\theta$, $A_{GDL}$ first runs the simulation algorithm for the random oracle $H_1$ to get $h_1 = H_1(id_R, PK_R^{(1)}, PK_R^{(2)})$ and $h_1' = H_1(id_S, PK_S^{(1)}, PK_S^{(2)})$, and then checks if there exist a tuple <$m$, $R$, $id_S$, $PK_S^{(1)}$, $PK_S^{(2)}$, $h$> in $H_2$List and a tuple <$v$, $h_3$> in $H_3$List such that $R = (PK_S^{(1)} \cdot PK_S^{(2)} \cdot g_1^{h_1'} \cdot g^h)^u$, $m = c \oplus h_3$ and $O^{rDDH}(g, g^a, R, v \cdot R^{-SK_R - \alpha h_1}) = 1$. If such two tuples exist, $A_{GDL}$ returns $m$ to $A_I$ as the designcryption of $\sigma$; otherwise, it rejects $\sigma$. Note that a valid ciphertext is rejected with probability smaller than $q_{dsc}/2^k$ across the whole game. (2) Otherwise, $A_{GDL}$ can answer the query according to the specification of the algorithm *Signcrypt* since it knows the sender $id_R$'s private key and certificate.

Finally, $A_I$ outputs a valid ciphertext $\sigma^* = (h^*, u^*, c^*)$ from $id_S^*$ to $id_R^*$. If $id_S^* \neq id_\theta$, then $A_{GDL}$ aborts. Otherwise, having the knowledge of $id_R^*$'s private key and certificate, $A_{GDL}$ can designcrypt $\sigma^*$ to obtain $m^*$ and $R^*$. Then, $A_{GDL}$ uses the oracle replay technique [28] to generate one more valid ciphertext $\sigma^{*\prime} = (h^{*\prime}, u^{*\prime}, c^{*\prime})$ from $\sigma^*$ such that $h^* \neq h^{*\prime}$ and $u^* \neq u^{*\prime}$. Since $\sigma^*$ and $\sigma^{*\prime}$ are both valid ciphertexts for the same message and randomness, we obtain the following relations:

$$(PK_\theta^{(1)} \cdot PK_\theta^{(2)} \cdot g_1^{h_1} \cdot g^{h^*})^{u^*} = R^* = (PK_\theta^{(1)} \cdot PK_\theta^{(2)} \cdot g_1^{h_1} \cdot g^{h^{*\prime}})^{u^{*\prime}} \tag{11}$$

where $h_1 = H_1(id_\theta, PK_\theta^{(1)}, PK_\theta^{(2)})$. Then, we have

$$(g^{x_\theta} \cdot g^a \cdot g^{\alpha h_1} \cdot g^{h^*})^{u^*} = (g^{x_\theta} \cdot g^a \cdot g^{\alpha h_1} \cdot g^{h^{*\prime}})^{u^{*\prime}}, \tag{12}$$

$$g^{a(u^{*\prime} - u^*)} = g^{x_\theta (u^* - u^{*\prime})} \cdot g^{\alpha h_1 (u^* - u^{*\prime})} \cdot g^{h^* u^* - h^{*\prime} u^{*\prime}}, \tag{13}$$

$$g^a = g^{[(x_\theta + \alpha h_1)(u^* - u^{*\prime}) + (h^* u^* - h_2^{*\prime} u^{*\prime})](u^{*\prime} - u^*)^{-1}}. \tag{14}$$

Therefore, $A_{GDL}$ can compute

$$a = [(x_\theta + \alpha h_1)(u^* - u^{*\prime}) + (h^* u^* - h^{*\prime} u^{*\prime})](u^{*\prime} - u^*)^{-1} \tag{15}$$

as the solution to the given GDL problem.

From the Lemma 4 in [28], if $A_I$ produces a forgery with probability $\varepsilon \geq 10(q_{sc} + 1)(q_{sc} + q_2)/2^k$, then $A_{GDL}$ can use the oracle replay technique to generate one more valid ciphertext with advantage $\varepsilon' \geq 1/9$. Since $A_{GDL}$ can resolve the given GDL problem after it generates one more valid ciphertext from the valid ciphertext forged by $A_I$, we get that the advantage of $A_{GDL}$ in solving the GDL problem is $\varepsilon' \geq 1/9$.

Also from the Lemma 4 in [28], the time complexity of $A_{GDL}$ in solving the given GDL problem is bound by $\tau' \leq 23 q_{cu} q_2 [\tau + (q_1 + q_2 + q_3 + q_{cer} + q_{pri}) O(1) + q_{cu}(2\tau_{exp} + O(1)) +$

$q_{sc}(4\tau_{exp} + O(1)) + q_{dsc}(4\tau_{exp} + \tau_{rDDH} + O(1))][\varepsilon(1 - q_{dsc}/2^k)(1 - q_{sc}(q_2 + q_3 + 2q_{sc})/2^k)]^{-1}$, where $\tau_{exp}$ and $\tau_{rDDH}$ respectively denote the time for computing an exponentiation in $G$ and the one for a call to the restricted DDH oracle.

**Lemma 4.** Suppose that $H_1 \sim H_3$ are random oracles and $A_{II}$ is a Type-II adversary against the EUF-CBSC-CMA security of the proposed scheme that makes at most $q_{cu}$ queries to the oracle $O^{CreateUser\text{-}II}$, $q_{pri}$ queries to the oracle $O^{RequestPrivateKey}$, $q_{sc}$ queries to the oracle $O^{Signcrypt}$, $q_{dsc}$ queries to the oracle $O^{Designcrypt}$ and $q_i$ queries to the random oracles $H_i$ $(1 \le i \le 3)$. Assume also that $A_{II}$ produces a forgery with probability $\varepsilon \ge 10(q_{sc}+1)(q_{sc}+q_2)/2^k$ within a time $\tau$. Then there exists an algorithm $A_{GDL}$ to solve the GDL problem in the group $G$ with advantage $\varepsilon' \ge 1/9$ and running time $\tau' \le 23q_{cu}q_2[\tau + (q_1 + q_2 + q_3 + q_{pri})O(1) + q_{cu}(2\tau_{exp} + O(1)) + q_{sc}(4\tau_{exp} + O(1)) + q_{dsc}(4\tau_{exp} + \tau_{rDDH} + O(1))][\varepsilon(1 - q_{dsc}/2^k)(1 - q_{sc}(q_2 + q_3 + 2q_{sc})/2^k)]^{-1}$, where $\tau_{exp}$ and $\tau_{rDDH}$ respectively denote the time for computing an exponentiation in $G$ and the one for a call to the restricted DDH oracle.

Proof. Assume that $A_{GDL}$ is given a random GDL problem instance $(G, p, g, g^a, O^{rDDH})$. Its goal is to compute $a$ by interacting with $A_{II}$ as follows:

At the beginning of the game, $A_{GDL}$ first randomly chooses $\alpha \in Z_p^*$ and an index $\theta \in [1, q_{cu}]$. It then computes $g_1 = g^\alpha$ and starts EUF-CBSC-CMA Game-II by supplying $A_{II}$ with the master key $msk = \alpha$ and the public parameters $params = \{G, p, g, g_1, l_m, H_1, H_2, H_3\}$, where $H_1 \sim H_3$ are random oracles controlled by $A_{GDL}$.

During the query-answer phase, $A_{II}$ can adaptively make queries to the oracles $H_1$, $H_2$, $H_3$, $O^{CreateUser\text{-}II}$, $O^{RequestPrivateKey}$, $O^{Signcrypt}$ and $O^{Designcrypt}$. $A_{GDL}$ answers $A_{II}$'s queries to $H_1$, $H_2$, $H_3$, $O^{CreateUser\text{-}II}$, $O^{RequestPrivateKey}$ and $O^{Signcrypt}$ in the same way as the proof of Lemma 2 and and handles other queries as follows:

$O^{Designcrypt}$ *queries*: On receiving such a query on $(\sigma = (h, s, c), id_S, id_R)$, $A_{GDL}$ does the following: (1) If $id_R = id_\theta$, $A_{GDL}$ first runs the simulation algorithm for the random oracle $H_1$ to get $h_1 = H_1(id_R, PK_R^{(1)}, PK_R^{(2)})$ and $h_1' = H_1(id_S, PK_S^{(1)}, PK_S^{(2)})$, and then checks if there exist a tuple $<m, R, id_S, PK_S^{(1)}, PK_S^{(2)}, h>$ in $H_2$List and a tuple $<v, h_3>$ in $H_3$List such that $R = (PK_S^{(1)} \cdot PK_S^{(2)} \cdot g_1^{h_1'} \cdot g^h)^u$, $m = c \oplus h_3$ and $O^{rDDH}(g, g^a, R, v \cdot R^{-Cert_R}) = 1$. If such two tuples exist, $A_{GDL}$ returns $m$ to $A_{II}$ as the designcryption of $\sigma$; otherwise, it rejects $\sigma$. Note that a valid ciphertext is rejected with probability smaller than $q_{dsc}/2^k$ across the whole game. (2) Otherwise, $A_{GDL}$ designcrypts $\sigma$ in the normal way since it knows the receiver $id_R$'s private key and certificate.

Finally, $A_{II}$ outputs a valid ciphertext $\sigma^* = (h^*, u^*, c^*)$ from $id_S^*$ to $id_R^*$. If $id_S^* \ne id_\theta$, then $A_{GDL}$ aborts. Otherwise, having the knowledge of the user $id_R^*$'s private key and certificate, $A_{GDL}$ can designcrypt $\sigma^*$ to obtain $m^*$ and $R^*$. Then, $A_{GDL}$ uses the oracle replay technique [28] to generate one more valid ciphertext $\sigma^{*\prime} = (h^{*\prime}, u^{*\prime}, c^{*\prime})$ from $\sigma^*$ such that $h^* \ne h^{*\prime}$ and $u^* \ne u^{*\prime}$. Since $\sigma^*$ and $\sigma^{*\prime}$ are both valid ciphertexts for the same message and randomness, we obtain the following relations:

$$(PK_\theta^{(1)} \cdot PK_\theta^{(2)} \cdot g_1^{h_1} \cdot g^{h^*})^{u^*} = R^* = (PK_\theta^{(1)} \cdot PK_\theta^{(2)} \cdot g_1^{h_1} \cdot g^{h^{*\prime}})^{u^{*\prime}} \quad (16)$$

where $h_1 = H_1(id_\theta, PK_\theta^{(1)}, PK_\theta^{(2)})$. Then, we have

$$(g^{a}\cdot g^{y_\theta}\cdot g^{\alpha h_1}\cdot g^{h^*})^{u^*}=(g^{a}\cdot g^{y_\theta}\cdot g^{\alpha h_1}\cdot g^{h^{*\prime}})^{u^{*\prime}}, \tag{17}$$

$$g^{a(u^{*\prime}-u^*)}=g^{y_\theta(u^*-u^{*\prime})}\cdot g^{\alpha h_1(u^*-u^{*\prime})}\cdot g^{h^*u^*-h^{*\prime}u^{*\prime}}, \tag{18}$$

$$g^{a}=g^{[(y_\theta+\alpha h_1)(u^*-u^{*\prime})+(h^*u^*-h^{*\prime}u^{*\prime})](u^{*\prime}-u^*)^{-1}}. \tag{19}$$

Therefore, $A_{GDL}$ can compute

$$a=[(y_\theta+\alpha h_1)(u^*-u^{*\prime})+(h^*u^*-h^{*\prime}u^{*\prime})](u^{*\prime}-u^*)^{-1} \tag{20}$$

as the solution to the given GDL problem.

From the Lemma 4 in [28], if $A_{II}$ produces a forgery with probability $\varepsilon\geq 10(q_{sc}+1)(q_{sc}+q_2)/2^k$, then $A_{GDL}$ can use the oracle replay technique to generate one more valid ciphertext with advantage $\varepsilon'\geq 1/9$. Since $A_{GDL}$ can resolve the given GDL problem after it generates one more valid ciphertext from the valid ciphertext forged by $A_{II}$, we get that the advantage of $A_{GDL}$ in solving the GDL problem is $\varepsilon'\geq 1/9$.

Also from the Lemma 4 in [28], the time complexity of $A_{GDL}$ in solving the given GDL problem is bound by $\tau' \leq 23q_{cu}q_2[\tau+(q_1+q_2+q_3+q_{pri})O(1)+q_{cu}(2\tau_{exp}+O(1))+q_{sc}(4\tau_{exp}+O(1))+q_{dsc}(4\tau_{exp}+\tau_{rDDH}+O(1))][\varepsilon(1-q_{dsc}/2^k)(1-q_{sc}(q_2+q_3+2q_{sc})/2^k)]^{-1}$, where $\tau_{exp}$ and $\tau_{rDDH}$ respectively denote the time for computing an exponentiation in $G$ and the one for a call to the restricted DDH oracle.

### 5.3 Performance Comparison

We next make a comparison of our proposed scheme and the previous certificate-based signcryption schemes.

The details of the compared schemes are listed in **Table 1**, where we compare the schemes on computation complexity of signcryption and designcryption. We mainly consider four atomic operations: pairing, exponentiation in $G_T$, exponentiation in $G$ and hash. Here $G$ is an additive or multiplicative cyclic group, $G_T$ is the target group in the setting of bilinear pairing, *i.e.*, the bilinear pairing is $e$: $G\times G\rightarrow G_T$. For simplicity, we denote these operations by $Pa$, $Exp_{G_T}$, $Exp_G$ and $Ha$ respectively.

**Table 1.** Comparison of the certificate-based signcryption schemes

| Schemes | Signcryption cost | Designcryption cost |
|---|---|---|
| Ours | $4Exp_G+3Ha$ | $4Exp_G+3Ha$ |
| [18] | $1Pa+1Exp_{G_T}+4Exp_G+3Ha$ | $3Pa+1Exp_{G_T}+1Exp_G+3Ha$ |
| [20] | $1Pa+5Exp_G+4Ha$ | $4Pa+2Exp_G+3Ha$ |
| [22] | $1Pa+3Exp_{G_T}+3Exp_G+3Ha$ | $3Pa+2Exp_{G_T}+3Ha$ |

From **Table 1**, we can see that both the signcryption and designcryption algorithms in our scheme need computing four exponentiations and three hashes. In any other existing paring-based certificate-based signcryption scheme, the signcryption algorithm requires computing at least one bilinear pairing, five exponentiations and three hashes while the designcryption algorithm requires computing at least three pairings, two exponentiations and three hashes. Actually, the computation performance of our scheme can be further optimized

when $g_1^{H_1(id_U, PK_U^{(1)}, PK_U^{(2)})}$ can be pre-computed. Such a pre-computation enables us to additionally reduce one exponentiation and one hash computation in both the signcryption algorithm and the designcryption algorithm.

To give a much clearer comparison and also show that our scheme is more suitable for the computation-limited devices, we make a concrete time analysis of the compared schemes. The results are given in **Table 2**. Here, we estimate the computation time of the compared schemes on a standard MICA2 sensor node. According to the experiment results in [29, 30], for 80-bit security, one pairing computation takes 1.90s and one exponentiation in the group $G$ takes 0.32s on a MICA2 sensor node. However, the exponentiation in the target group $G_T$ takes more time than the exponentiation in the group $G$ because of the fact that it is computed in a field much bigger than the field in which $G$ is defined. In usual implementations of pairing, one exponentiation in $G_T$ costs about equal to four exponentiations in $G$ [31]. Considering that the overheads of hash operations and arithmetic operations in $Z_p^*$ are very small compared to the expensive pairing and exponentiation operations, we ignore these costs in the time analysis.

**Table 2.** Computation time of the compared schemes on MICA2 sensors

| Schemes | Signcryption cost (s) | Designcryption cost (s) |
|---|---|---|
| Ours | 1.28 | 1.28 |
| [18] | 4.46 | 7.3 |
| [20] | 3.5 | 8.24 |
| [22] | 6.7 | 7.96 |

From the comparison in **Table 2**, we can conclude that our scheme enjoys obvious advantage in the computation time and is more suitable to be employed in the computation-limited environments.

## 6. Conclusion

In this paper, we presented a new certificate-based signcryption scheme and proved its security under the gap Diffie-Hellman assumption and the gap discrete logarithm assumption. As our proposed scheme does not require any costly pairing operation, it is more efficient than the previous certificate-based signcryption schemes which have to be based on the bilinear pairings. However, a limitation of our scheme is that its security can only be achieved in the random oracle model. So, it would be interesting to construct paring-free certificate-based signcryption in the standard model.

## References

[1] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. of Advances in Cryptology - Crypto 1984*, pp. 47-53, August 19-22, 1984. Article (CrossRef Link).

[2] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proc. of Advances in Cryptology - Asiacrypt 2003*, pp. 452-473, November 30-December 4, 2003. Article (CrossRef Link).

[3] C. Gentry, "Certificate-based encryption and the certificate revocation problem," in *Proc. of Advances in Cryptology - Eurocrypt 2003*, pp. 272-293, May 4-8, 2003. Article (CrossRef Link).

[4] C. Sur, C. D. Jung and K. H. Rhee, "Multi-receiver certificate-based encryption and application to public key broadcast encryption," in *Proc. of 2007 ECSIS Symposium on Bio-inspired, Learning,*

*and Intelligent Systems for Security*, pp. 35-40, August 9-10, 2007. Article (CrossRef Link).
[5] D. Galindo, P. Morillo and C. Ràfols, "Improved certificate-based encryption in the standard model," *Journal of Systems and Software*, vol. 81, no. 7, pp. 1218-1226, July, 2008. Article (CrossRef Link).
[6] J. K. Liu and J. Zhou, "Efficient certificate-based encryption in the standard model," in *Proc. of 6th Int. Conf. on Security and Cryptography for Networks*, pp. 144-155, September 10-12, 2008. Article (CrossRef Link).
[7] Y. Lu, J. Li and J. Xiao, "Constructing efficient certificate-based encryption with paring," *Journal of Computers*, vol. 4, no. 1, pp. 19-26, January, 2009. Article (CrossRef Link).
[8] Z. Shao, "Enhanced certificate-based encryption from pairings," *Computers and Electrical Engineering*, vol. 37, no. 2, pp. 136-146, March, 2011. Article (CrossRef Link).
[9] Y. Lu and J. Li, "Constructing pairing-free certificate-based encryption," *International Journal of Innovative Computing, Information and Control*, vol. 9, no. 11, pp. 4509-4518, November, 2013. Article (CrossRef Link).
[10] J. Yao, J. Li and Y. Zhang, "Certificate-based encryption scheme without pairing," *KSII Transactions on Internet and Information Systems*, vol. 7, no. 6, pp. 1480-1491, June, 2013. Article (CrossRef Link).
[11] B. G. Kang, J. H. Park and S. G. Hahn, "A certificate-based signature scheme," in *Proc. of Topics in Cryptology - CT-RSA 2004*, pp. 99-111, February 23-27, 2004. Article (CrossRef Link).
[12] M. H. Au, J. K. Liu, W. Susilo and T. H. Yuen, "Certificate based (linkable) ring signature," in *Proc. of 3rd Information Security Practice and Experience Conference*, pp.79-92, May 7-9, 2007. Article (CrossRef Link).
[13] J. Li, X. Huang, Y. Mu, W. Susilo and Q. Wu, "Certificate-based signature: security model and efficient construction," in *Proc. of 4th European PKI Workshop Theory and Practice*, pp. 110-125, June 28-30, 2007. Article (CrossRef Link).
[14] J. Li, X. Huang, Y. Mu, W. Susilo and Q. Wu, "Constructions of certificate-based signature secure against key replacement attacks," *Journal of Computer Security*, vol. 18, no. 3, pp. 421-449, August, 2010. Article (CrossRef Link).
[15] J. K. Liu, J. Baek, W. Susilo, and J. Zhou, "Certificate based signature schemes without pairings or random oracles," in *Proc. of 11th Information Security conference*, pp. 285-297, September 15-18, 2008. Article (CrossRef Link).
[16] W. Wu, Y. Mu, W. Susilo, X. Huang, "Certificate-based signatures, revisited," *Journal of Universal Computer Science*, vol. 15, no. 8, pp. 1659-1684, 2009. Article (CrossRef Link).
[17] Y. Zheng, "Digital signcryption or how to achieve cost (signature & encryption) << cost (signature) + cost (encryption)," in *Proc. of Advances in Cryptology - Crypto 1997*, pp. 165-179, August 17-21, 1997. Article (CrossRef Link).
[18] F. Li, X. Xin and Y. Hu, "Efficient certificate-based signcryption scheme from bilinear pairings," *International Journal of Computers and Applications*, vol. 30, no. 2, pp. 129-133, March, 2008. Article (CrossRef Link).
[19] L. Chen and J. Malone-Lee, "Improved identity-based signcryption," in *Proc. of 8th Int. Workshop on Theory and Practice in Public Key Cryptography*, pp. 362-379, January 23-26, 2005. Article (CrossRef Link).
[20] M. Luo, Y. Wen and H. Zhao, "A certificate-based signcryption scheme," in *Proc. of 2008 Int. Conf. on Computer Science and Information Technology*, pp. 17-23, August 29 - September 2, 2008. Article (CrossRef Link).
[21] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in *Proc. of* 1st *ACM Conf. on Communications and Computer Security*, pp. 62-73, November 3-5, 1993. Article (CrossRef Link).
[22] J. Li, X. Huang, M. Hon and Y. Zhang, "Certificate-based signcryption with enhanced security features," *Computers & Mathematics with Applications*, vol. 64, no. 6, pp. 1587-1601, September, 2012. Article (CrossRef Link).
[23] C. P. Schnorr, "Efficient identifications and signatures for smart cards," in *Proc. of Advances in Cryptology - Crypto 1989*, pp. 239-252, August 20-24, 1989. Article (CrossRef Link).

[24] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161-174, March, 1991. Article (CrossRef Link).
[25] E. Fujisaki and T. Okamoto, "How to enhance the security of public-key encryption at minimum cost," in *Proc. of 2nd Int. Workshop on Theory and Practice in Public Key Cryptography*, pp. 53-68, March 1-3, 1999. Article (CrossRef Link).
[26] T. Okamoto and D. Pointcheval, "The gap-problems: a new class of problems for the security of cryptographic schemes," in *Proc. of 4th Int. Workshop on Theory and Practice in Public Key Cryptography*, pp. 104-118, February 13-15, 2001. Article (CrossRef Link).
[27] J. Baek, R. Steinfeld and Y. Zheng, "Formal proofs for the security of signcryption," *Journal of Cryptology*, vol. 20, no. 2, pp. 203-235, 2007. Article (CrossRef Link).
[28] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, no. 3, pp. 361-396, 2000. Article (CrossRef Link).
[29] L. B. Oliveira, D. F. Aranha, C. P. L. Gouvêa, M. Scott, D. F. Câmara, J. López and R. Dahab, "TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks," *Computer Communications*, vol. 34, no. 3, pp. 485-493, 2011. Article (CrossRef Link).
[30] D. F. Aranha, R. Dahab, J. López, and L. B. Oliveira, "Efficient implementation of elliptic curve cryptography in wireless sensors," *Advances in Mathematics of Communications*, vol. 4, no. 2, pp. 169-187, 2010. Article (CrossRef Link).
[31] L. Chen, P. Morrissey and N. P. Smart, "Pairings in trusted computing," in *Proc. of 2nd International Conference on Pairing-based Cryptography*, pp. 1-17, September 1-3, 2008. Article (CrossRef Link).

**Yang Lu** received the Ph.D. degree from PLA University of Science and Technology in 2009. He has been working in HoHai University from 2003. Currently, he is an Assistant Professor in College of Computer and Information Engineering. He has published more than 30 papers in international conferences and journals. His research interest includes information security and cryptography.

**Jiguo Li** received the Ph.D. degree from Harbin Institute of Technology in 2003. He has been working in HoHai University from 2003. Currently, he is a Professor in College of Computer and Information Engineering. His major research interests include information security and cryptography, network security, wireless security and trusted computing etc. He has published more than 70 scientific papers and two books. He has served as a PC member of several international conferences and the reviewer of some international journals and conferences.