8-2010

An efficient signcryption scheme with key privacy and its extension to ring signcryption

Chung Ki LI

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

Duncan S. WONG

Xiaotie DENG

Sherman S. M. CHOW

Journal of Computer Security 18 (2010) 451–473
DOI 10.3233/JCS-2009-0374
IOS Press

An efficient signcryption scheme with key privacy and its extension to ring signcryption *

Chung Ki Li [a], Guomin Yang [a], Duncan S. Wong [a], Xiaotie Deng [a] and Sherman S.M. Chow [b]

[a] *Department of Computer Science, City University of Hong Kong, China*
E-mails: {travisli, csyanggm, duncan, deng}@cs.cityu.edu.hk
[b] *Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, NY 10012, USA*
E-mail: schow@cs.nyu.edu

In *Information Processing Letters* (2006), Tan pointed out that the anonymous signcryption scheme proposed by Yang, Wong and Deng (YWD) in ISC 2005 provides neither confidentiality nor anonymity. However, no discussion has been made on how a secure scheme can be made and there is no secure scheme available to date. In this paper, we propose a modification of YWD scheme which resolves the security issues of the original scheme without sacrificing its high efficiency and simple design. Indeed, we show that our scheme achieves confidentiality, existential unforgeability and anonymity with more precise reduction bounds. We also give a variation of our scheme and extend it to a ring signcryption scheme by using the technique due to Boneh, Gentry, Lynn and Shacham.

Keywords: Signcryption, ring signcryption, privacy, anonymity

1. Introduction

Signcryption, introduced by Zheng in 1997 [25], is a cryptographic primitive targeting to provide unforgeability and confidentiality simultaneously as typical signature-then-encryption technique does but with less computational complexity and lower communication cost. Due to these advantages, signcryption is suitable for many applications which require secure and authenticated message delivery using resource limited devices.

There have been many signcryption schemes proposed after Zheng's publication (e.g. [2,9–11,13–16,20,24]). In 2002, Baek et al. [3] first formally defined the security notions of signcryption, which are similar to the traditional semantic security against adaptive chosen ciphertext attack (IND-CCA2) [17] and existential unforgeability against adaptive chosen message attack (EUF-CMA) [12]. The notion of *in-*

* A preliminary version appears in the *Proc. of the 4th European PKI Workshop – Theory and Practice (EuroPKI 2007)*. The work was supported by a grant from CityU (Project No. 7002001).

sider security[1] was first defined by An et al. [1]. The notion allows an adversary to not only access the public keys of both sender and receiver but also know the sender's private key. For example, a signcryption scheme is said to be "insider secure" if the adversary cannot compromise the confidentiality of a ciphertext even the adversary knows the sender's private key. Similar notion has later been extended to other security properties for signcryption [9,13,14]. Those properties include unforgeability, anonymity, etc.

In [9], Boyen proposed a new set of security models for signcryption schemes (under the identity-based setting [19]). In particular, a new requirement called *ciphertext anonymity* was proposed. It requires that a ciphertext should appear anonymous to anyone except the actual recipient. It hides the identities of both the sender and the recipient of the ciphertext. This notion can be viewed as an extension of *key privacy* introduced by Bellare et al. [4] for public key encryption. A signcryption scheme with ciphertext anonymity or key privacy protects the identities of both sender and recipient from being known from a ciphertext.

In [14], Libert and Quisquater proposed a signcryption scheme with ciphertext anonymity. However, [21] and [23] independently demonstrated that it is neither semantically secure nor anonymous under chosen plaintext attack. In [23], Yang, Wong and Deng also proposed an improvement (hereinafter referred as the YWD scheme) based on [14]. However, a recent result by Tan [22] showed that the YWD scheme is not semantically secure and does not satisfy ciphertext anonymity, under insider chosen-ciphertext attack. However, no improved scheme was proposed by Tan.

Our contributions. It is still unknown if the YWD scheme can be improved to a secure one, while maintaining the advantages of the original scheme being highly efficient and simple. In this paper, we propose a modification of the YWD scheme. The modified scheme not only solves the security issues of the original scheme, but also maintains its efficiency. In particular, we show that our scheme achieves confidentiality, existential unforgeability and anonymity with more precise reduction bounds.

To consider anonymity one step further, we define the notion of ring signcryption. In a ring signcryption scheme, after decrypting the ciphertext, the recipient will obtain a ring signature instead of a standard signature, and the recipient cannot tell which party inside the ring generated the ciphertext. We also give a variation of our improved signcryption scheme and show that it can be extended to a ring signcryption scheme by using the technique of [7].

Organization. We give the definition and security models of a signcryption scheme with ciphertext anonymity (or key privacy) in Section 2. It is then followed by the review and discussion of YWD signcryption scheme and Tan's attacks in Section 3. This leads us to the description of our method for solving the security issues of

[1]The original paper of An et al. [1] only presents the insider attack against the integrity of a signcryption. The idea has later been extended to confidentiality and other security properties [9,14].

the YWD scheme. Our construction and its security analysis are given in Section 4. Based on our construction, we define and construct a ring signcryption scheme in Section 5. We conclude the paper in Section 6.

2. The definition and security models of signcryption with key privacy

A signcryption scheme is a quadruple of probabilistic polynomial time (PPT) algorithms (**Keygen**, **Signcrypt**, **De-signcrypt**, **Verify**).

$(sk, pk) \leftarrow$ **Keygen**(1^k) is the key generation algorithm which takes a security parameter $k \in \mathbb{N}$ and generates a private/public key pair (sk, pk).

$\sigma \leftarrow$ **Signcrypt**$(1^k, m, sk_{\mathrm{U}}, pk_{\mathrm{R}})$ takes k, a message m, a private key sk_{U} and a public key pk_{R}, outputs a ciphertext σ. m is drawn from a message space M which is defined as $\{0,1\}^n$ where n is some polynomial in k.

$(m, s, pk_{\mathrm{U}})/\texttt{reject} \leftarrow$ **De-signcrypt**$(1^k, \sigma, sk_{\mathrm{R}})$ takes k, σ and a private key sk_{R}, outputs either a triple (m, s, pk_{U}) where $m \in M$, s is a signature and pk_{U} is a public key, or `reject` which indicates the failure of de-signcryption.

$\texttt{true}/\texttt{false} \leftarrow$ **Verify**$(1^k, m, s, pk_{\mathrm{U}})$ takes k, $m \in M$, a signature s and a public key pk_{U}, outputs `true` for a valid signature or `false` for an invalid signature.

For simplicity, we omit the notation of 1^k from the inputs of **Signcrypt**, **De-signcrypt** and **Verify** in the rest of this paper. Note that the specification above requires the corresponding signcryption scheme to support the "unwrapping" option introduced in [15]. The "unwrapping" option allows the receiver of a ciphertext to release the message and derive the embedded sender's signature from the ciphertext for public verification. Early schemes such as [25] do not support the "unwrapping" option and therefore not satisfy this definition.

Definition 2.1 (Completeness). For any $m \in M$, $(sk_{\mathrm{U}}, pk_{\mathrm{U}}) \leftarrow$ **Keygen**(1^k) and $(sk_{\mathrm{R}}, pk_{\mathrm{R}}) \leftarrow$ **Keygen**(1^k) such that $sk_{\mathrm{U}} \neq sk_{\mathrm{R}}$, we have

$$(m, s, pk_{\mathrm{U}}) \leftarrow \textbf{De-signcrypt}(\textbf{Signcrypt}(m, sk_{\mathrm{U}}, pk_{\mathrm{R}}), sk_{\mathrm{R}})$$

and $\texttt{true} \leftarrow$ **Verify**(m, s, pk_{U}).

Informally, we consider a secure signcryption scheme with key privacy to be semantically secure against adaptive chosen ciphertext attack, existentially unforgeable against chosen message attack, and anonymous in the sense that a ciphertext should contain no information in the clear that identifies the author or the recipient of the message and yet be decipherable by the intended recipient. We capture these notions in the following definitions. They are similar to those defined by Libert and Quisquater in [14].

Definition 2.2 (Confidentiality). A signcryption scheme is semantically secure against insider chosen ciphertext attack (SC-IND-CCA) if no PPT adversary has a non-negligible advantage in the following game:

1. The challenger runs **Keygen** to generate a key pair $(sk_{\mathrm{U}}, pk_{\mathrm{U}})$. sk_{U} is kept secret while pk_{U} is given to adversary $\mathcal{A}$.
2. In the first stage, $\mathcal{A}$ makes a number of queries to the following oracles:
   (a) Signcryption oracle: $\mathcal{A}$ prepares a message $m \in M$ and a public key pk_{R}, and queries the signcryption oracle (simulated by the challenger) for the result of **Signcrypt**$(m, sk_{\mathrm{U}}, pk_{\mathrm{R}})$. The result is returned if $pk_{\mathrm{R}} \neq pk_{\mathrm{U}}$ and pk_{R} is valid in the sense that pk_{R} is in the range of **Keygen** with respect to the security parameter. Otherwise, a symbol "$\perp$" is returned for rejection.
   (b) De-signcryption oracle: $\mathcal{A}$ produces a ciphertext σ and queries for the result of **De-signcrypt** $(\sigma, sk_{\mathrm{U}})$. The result is made of a message, a signature and the sender's public key if the de-signcryption is successful and the signature is valid under the recovered sender's public key. Otherwise, a symbol "$\perp$" is returned for rejection.

   These queries can be asked adaptively: each query may depend on the answers of previous ones.
3. $\mathcal{A}$ produces two plaintexts $m_0, m_1 \in M$ of equal length and a valid private key sk_{S} such that sk_{S} is in the range of **Keygen** with respect to the security parameter. The challenger flips a coin $\check{b} \overset{R}{\leftarrow} \{0,1\}$ and computes a signcryption $\sigma^* = \textbf{Signcrypt}(m_{\check{b}}, sk_{\mathrm{S}}, pk_{\mathrm{U}})$ of $m_{\check{b}}$ with the sender's private key sk_{S} under the receiver's public key pk_{U}. σ^* is sent to $\mathcal{A}$ as a challenge ciphertext.
4. $\mathcal{A}$ makes a number of new queries as in the first stage with the restriction that it cannot query the de-signcryption oracle with σ^*.
5. At the end of the game, $\mathcal{A}$ outputs a bit b' and wins if $b' = \check{b}$.

$\mathcal{A}$'s advantage is defined as $Adv^{ind-cca}(\mathcal{A}) = \Pr[b' = \check{b}] - \frac{1}{2}$ and the probability that $b' = \check{b}$ is called the probability that $\mathcal{A}$ wins the game.

The definition above captures the advantage of an active adversary over an eavesdropper. That is, the adversary knows and has the full control of the signing key. This also gives us insider-security for confidentiality [1,9,14].

Definition 2.3 (Unforgeability). A signcryption scheme is existentially unforgeable against chosen-message insider attack (SC-EUF-CMA) if no PPT forger has a non-negligible advantage in the following game:

1. The challenger runs **Keygen** to generate a key pair $(sk_{\mathrm{U}}, pk_{\mathrm{U}})$. sk_{U} is kept secret while pk_{U} is given to forger $\mathcal{F}$.
2. The forger $\mathcal{F}$ adaptively makes a number of queries to the signcryption oracle and the de-signcryption oracle as in the confidentiality game.

3. $\mathcal{F}$ produces a ciphertext σ and a valid key pair $(sk_{\mathrm{R}}, pk_{\mathrm{R}})$ in the sense that the key pair is in the range of **Keygen** and wins the game if
   (a) **De-signcrypt**$(\sigma, sk_{\mathrm{R}})$ returns a tuple (m, s, pk_{U}) such that $\texttt{true} \leftarrow$ **Verify**(m, s, pk_{U}), and
   (b) σ is not the output of the signcryption oracle.

We allow the forger to have the full control of the de-signcryption key pair $(sk_{\mathrm{R}}, pk_{\mathrm{R}})$. This also captures the notion of insider-security for unforgeability.

Definition 2.4 (Ciphertext anonymity). A signcryption scheme is ciphertext anonymous against chosen-ciphertext insider attack (SC-ANON-CCA) if no PPT distinguisher has a non-negligible advantage in the following game:

1. The challenger generates two distinct public key pairs $(sk_{\mathrm{R},0}, pk_{\mathrm{R},0})$ and $(sk_{\mathrm{R},1}, pk_{\mathrm{R},1})$ using **Keygen**, and gives $pk_{\mathrm{R},0}$ and $pk_{\mathrm{R},1}$ to the distinguisher $\mathcal{D}$.
2. In the first stage, $\mathcal{D}$ adaptively makes a number of queries in the form of **Signcrypt**$(m, sk_{\mathrm{R},c}, pk_{\mathrm{R}})$ or **De-signcrypt**$(\sigma, sk_{\mathrm{R},c})$, for $c = 0$ or $c = 1$. pk_{R} is some arbitrary but valid recipient key such that $pk_{\mathrm{R}} \neq pk_{\mathrm{R},c}$.
3. After completing the first stage, $\mathcal{D}$ outputs two valid and distinct private keys $sk_{S,0}$ and $sk_{S,1}$, and a plaintext $m \in M$.
4. The challenger then flips two coins $b, b' \overset{R}{\leftarrow} \{0, 1\}$ and computes a challenge ciphertext $\sigma =$ **Signcrypt**$(m, sk_{S,b}, pk_{\mathrm{R},b'})$ and sends it to $\mathcal{D}$.
5. $\mathcal{D}$ adaptively makes a number of new queries as above with the restriction that it is not allowed to ask the de-signcryption oracle of the challenge ciphertext σ.
6. At the end of the game, $\mathcal{D}$ outputs bits d, d' and wins the game if $(d, d') = (b, b')$.

$\mathcal{D}$'s advantage is defined as $Adv^{anon-cca}(\mathcal{D}) = \Pr[(d, d') = (b, b')] - \frac{1}{4}$.

The ciphertext anonymity definition above follows that of Libert and Quisquater in [14], Definition 4, which is considered to be an extension of the "key privacy" notion of public key encryption [4]. We only consider this definition for key privacy in this paper rather than also considering an additional one called key invisibility [14], Definition 5. We believe that the definition above is more intuitive. With only a few differences, one can also consider it as a non-identity based version of Boyen's definition [9] of ciphertext anonymity in the identity-based setting.

3. Preliminaries

Bilinear map. Let k be a system-wide security parameter. Let q be a k-bit prime. Let $\mathbb{G}_1$ be an additive cyclic group of order q and $\mathbb{G}_2$ be a multiplicative cyclic group of the same order. Let P be a generator of $\mathbb{G}_1$. A bilinear map is defined as $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties:

1. *Bilinear*: For all $U, V \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}$, we have $e(aU, bV) = e(U, V)^{ab}$.
2. *Non-degenerate*: $e(P, P) \neq 1$.
3. *Computable*: there is an efficient algorithm to compute $e(U, V)$ for any U, $V \in \mathbb{G}_1$.

Modified pairings [6] obtained from the Weil or the Tate pairing provide admissible maps of this kind.

The Computational Diffie–Hellman Problem. The Computational Diffie–Hellman Problem (CDH) in $\mathbb{G}_1$ is to compute abP from $\langle P, aP, bP\rangle$ where a, b are random in $\mathbb{Z}_q$.

3.1. YWD signcryption scheme [23]

Suppose each element in $\mathbb{G}_1$ can be represented distinctly using l bits. Let $H_1 : \{0,1\}^{n+2l} \to \mathbb{G}_1$, $H_2 : \mathbb{G}_1^3 \to \{0,1\}^l$ and $H_3 : \mathbb{G}_1^3 \to \{0,1\}^{n+l}$ be cryptographic hash functions where n denotes the length of a plaintext in binary representation and is in some polynomial of k. The scheme is described as follows:

Keygen: A private key is generated by picking a random $x_u \leftarrow \mathbb{Z}_q$ and the corresponding public key is computed as $Y_u = x_u P$. In the following, the sender and the receiver are denoted by $u = S$ and $u = R$, and their private/public key pairs are denoted by (x_S, Y_S) and (x_R, Y_R), respectively.

Signcrypt: To signcrypt a message $m \in \{0,1\}^n$ for receiver R, sender S carries out the following steps:

1. Pick a random $r \leftarrow \mathbb{Z}_q$ and compute $U = rP$.
2. Compute $V = x_S H_1(m, U, Y_R)$.
3. Compute $W = V \oplus H_2(U, Y_R, rY_R)$ and $Z = (m \| Y_S) \oplus H_3(U, Y_R, rY_R)$.

The ciphertext is $\sigma = (U, W, Z)$.

De-signcrypt: When a ciphertext $\sigma = (U, W, Z)$ is received, receiver R performs the following steps:

1. Compute $V = W \oplus H_2(U, Y_R, x_R U)$.
2. Compute $(m \| Y_S) = Z \oplus H_3(U, Y_R, x_R U)$.
3. If $Y_S \notin \mathbb{G}_1$, outputs `reject`. Otherwise, compute $H = H_1(m, U, Y_R)$ and check if $e(Y_S, H) = e(P, V)$.
4. If the equation holds, output $\langle m, (U, Y_R, V), Y_S\rangle$; otherwise, output `reject`.

Verify: For a message–signature pair $(m, (U, Y_R, V))$ and a signer's public key Y_S, the algorithm checks if $e(Y_S, H_1(m, U, Y_R)) = e(P, V)$. If the condition holds, it outputs `true`. Otherwise, it outputs `false`.

3.2. Tan's attacks against the YWD scheme

Adaptive chosen ciphertext attack. Based on the SC-IND-CCA game stated in Definition 2.2, Tan's attack [22] is carried out as follows.

Given the receiver's public key Y_{R}, the adversary $\mathcal{A}$ first chooses a sender's private key x_{S} and two same length messages m_0 and m_1, and sends them to the challenger. Then the challenger randomly picks $b \in \{0, 1\}$ and computes the challenge ciphertext of m_b as $C^* = (U^*, W^*, Z^*)$. After receiving C^*, $\mathcal{A}$ makes a guess of b to be 0 and computes a new ciphertext with a random message $\bar{m}$ which has the same length as m_0 and a random $\bar{x}_{\mathrm{S}} \leftarrow \mathbb{Z}_q$. $\mathcal{A}$ computes the ciphertext $\bar{C}$ as follows:

- $\bar{Y_{\mathrm{S}}} = \bar{x}_S P$,
- $V^* = x_S H_1(m_0, U^*, Y_{\mathrm{R}})$,
- $\bar{V} = \bar{x}_S H_1(\bar{m}, U^*, Y_{\mathrm{R}})$,
- $\bar{W} = (\bar{V} \oplus V^*) \oplus W^*$,
- $\bar{Z} = ((m_0 \oplus \bar{m}) \| (\bar{Y_{\mathrm{S}}} \oplus Y_{\mathrm{S}})) \oplus Z^*$.

At last, $\mathcal{A}$ sends $\bar{C} = (U^*, \bar{W}, \bar{Z})$ to the de-signcryption oracle which computes the following:

- $\hat{m} \| \hat{Y_{\mathrm{S}}} = \bar{Z} \oplus H_3(U^*, Y_{\mathrm{R}}, x_R U^*)$, here $\hat{Y_{\mathrm{S}}} = \bar{Y_{\mathrm{S}}}$,
- $\hat{V} = \bar{W} \oplus H_2(U^*, Y_{\mathrm{R}}, x_R U^*)$,
- $H = H_1(\hat{m}, U^*, Y_{\mathrm{R}})$.

If $\mathcal{A}$'s guess is correct, the de-signcryption oracle returns $\bar{m}$ with probability 1. Otherwise, the de-signcryption oracle returns "$\perp$" with overwhelming probability. A similar attack can be carried out to break the Ciphertext Anonymity, readers may refer to [22] for the details of the attack.

Discussions. The YWD scheme is not secure against Tan's attacks because the component V can easily be reconstructed with the sender's secret. Since the inputs of H_1 do not involve any secret value, the adversary under the notion of "insider security" can easily make use of the malleability property of W and Z to construct a related but different ciphertext. In the next section, we proposed an efficient solution of this problem.

4. Simple and efficient signcryption with key privacy

Our scheme is based on Boneh–Lynn–Shacham's (BLS) short signature [8] and ElGamal encryption. The recipient can decrypt without knowing who the sender is. Both the sender's public key and the related signature are recovered from the ciphertext such that the recipient can verify its authenticity.

4.1. Our construction

Suppose each element in $\mathbb{G}_1$ can uniquely be represented using an l-bit long binary string. Let $H_1 : \{0,1\}^n \times \mathbb{G}_1^3 \to \mathbb{G}_1$ and $H_2 : \mathbb{G}_1^3 \to \{0,1\}^{n+2l}$ be hash functions where n denotes the length of a plaintext and is in some polynomial of k. For security analysis, all hash functions are viewed as random oracles [5]. Our proposed scheme is described as follows.

Keygen. The same as that of the YWD scheme.

Signcrypt. To signcrypt a message $m \in \{0,1\}^n$ for receiver R, sender S carries out the following steps:

1. Pick a random $r \leftarrow \mathbb{Z}_q$ and compute $U = rP$.
2. Compute $V = x_S H_1(m, U, Y_R, rY_R)$ and $Z = (m \| Y_S \| V) \oplus H_2(U, Y_R, rY_R)$.

The ciphertext is $\sigma = (U, Z)$.

De-signcrypt. When a ciphertext $\sigma = (U, Z)$ is received, receiver R performs the following steps:

1. Compute $D = x_R U$.
2. Compute $(m \| Y_S \| V) = Z \oplus H_2(U, Y_R, D)$.
3. If $Y_S \notin \mathbb{G}_1$, output `reject`. Otherwise, compute $H = H_1(m, U, Y_R, D)$ and check if (P, Y_S, H, V) is a Diffie–Hellman tuple by $e(Y_S, H) \stackrel{?}{=} e(P, V)$.
4. If the check passes, output $\langle m, (U, Y_R, D, V), Y_S \rangle$; otherwise, output `reject`.

Verify. For a message–signature pair $(m, (U, Y_R, D, V))$ and a signer's public key Y_S, the algorithm checks if $e(Y_S, H_1(m, U, Y_R, D)) = e(P, V)$. If the condition holds, it outputs `true`. Otherwise, it outputs `false`.

Discussions. In our scheme, the signature V is signed on the value D which cannot be computed without the recipient's private key. Even with the knowledge of the sender's private key, the adversary cannot reconstruct V if D remains unknown. On the other hand, the signcryption oracle does not offer much help since no adaptive choice is available to make a signature of D. These give us semantic security under adaptive insider's chosen-ciphertext attack. The adversary needs to know the value of D to come up with a valid ciphertext, which renders the decryption oracle "useless" and explains why we get chosen-ciphertext security from chosen-plaintext secure ElGamal encryption.

4.2. Security analysis

Theorem 4.1. *Let k be a security parameter. Under the random oracle model, if there exists a PPT algorithm which can break the SC-IND-CCA security of the proposed signcryption scheme with advantage at least $\rho(k)$, then there exists a PPT algorithm which can solve the Computational Diffie–Hellman Problem with probability at least $2(1 - p2^{-poly(k)})\rho(k)$ where poly$(\cdot)$ is some polynomial and p is the maximum number of de-signcryption queries made in the model of SC-IND-CCA.*

Proof. For contradiction, we assume that there exists an adversary $\mathcal{A}$ who wins the game given in Definition 2.2 with non-negligible advantage. In the following, we construct an algorithm $\mathcal{B}$ to solve the CDH problem in $\mathbb{G}_1$ with the help of a DDH solver due to the bilinear pairing.

Suppose $\mathcal{B}$ is given a random instance of the CDH problem $(P, aP, bP) \in \mathbb{G}_1^3$, $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine to find the solution abP. $\mathcal{B}$ sets up a simulated environment of SC-IND-CCA model for $\mathcal{A}$ as follows:

$\mathcal{B}$ gives bP to $\mathcal{A}$ as the challenging public key Y_u.

$\mathcal{B}$ maintains two lists L1 and L2 for simulating hash oracles H_1 and H_2, respectively. When a hash query $H_1(m, P_1, P_2, P_3)$ is received, where $m \in \{0,1\}^n$ and $P_1, P_2, P_3 \in \mathbb{G}_1$, $\mathcal{B}$ checks if the query tuple (m, P_1, P_2, P_3) is already in L1. If it exists, the existing result in L1 is returned. If it does not exist but $e(P_1, P_2) = e(P, P_3)$ and $(P_1, P_2, \top)$ is in L1, where "$\top$" is a special symbol, then $\mathcal{B}$ replaces "$\top$" in the entry with P_3 and returns the existing result in the entry. For all other cases, $\mathcal{B}$ randomly chooses $t \leftarrow \mathbb{Z}_q$ and returns tP to $\mathcal{A}$. The query tuple and return value are then saved in L1. For enabling the retrieval of t possibly at some later time of the simulation, the value of t is also stored in L1 along the entry. Hash queries to H_2 are handled similarly.

For a signcryption query on a message m with a receiver's public key Y_R, $\mathcal{B}$ checks if $Y_R \in \mathbb{G}_1$. If it is incorrect or $Y_R = Y_u$, $\mathcal{B}$ returns a symbol "$\perp$" for rejection. Otherwise, $\mathcal{B}$ randomly picks $r \leftarrow \mathbb{Z}_q$, computes $U = rP$ and simulates $H_1(m, U, Y_R, rY_R)$ described as above. Suppose $H_1(m, U, Y_R, rY_R)$ is set to $t'P$. $\mathcal{B}$ then simulates $H_2(U, Y_R, rY_R)$, and computes the ciphertext $\sigma = (U, Z)$ where $Z = (m \| Y_u \| t'(Y_u)) \oplus H_2(U, Y_R, rY_R)$.

De-signcryption query on $\sigma = (U, Z)$ is answered as follows:

1. $\mathcal{B}$ looks for a tuple of the form (U, Y_u, λ) in L2 such that $e(P, \lambda) = e(U, Y_u)$ or $\lambda = \top$.
   - If the tuple (U, Y_u, λ) is not in L2, $\mathcal{B}$ adds a new entry into L2 by storing $(U, Y_u, \top)$ as the query tuple and a value randomly drawn from the range of H_2 as the oracle return. The special symbol "$\top$" is used as a marker for denoting that the real value should be the solution of the CDH problem instance (U, Y_u). This step ensures that the value of $H_2(U, Y_u, \lambda)$ is *fixed* before σ is de-signcrypted.

- If the tuple (U, Y_u, λ) is already in L2, the existing result will be used as the value of $H_2(U, Y_u, \lambda)$.

2. $\mathcal{B}$ computes $m\|Y_S\|V = Z \oplus H_2(U, Y_u, \lambda)$ and looks for a tuple of the form (m, U, Y_u, λ) in L1 such that $e(P, \lambda) = e(U, Y_u)$ or $\lambda = \top$.

   - If the tuple (m, U, Y_u, λ) is not in L1, $\mathcal{B}$ adds a new entry into L1 by storing (m, U, Y_u, λ) as the query tuple and setting $r'P$ as the oracle return, where $r' \leftarrow \mathbb{Z}_q$. Note that λ can be a value such that $e(P, \lambda) = e(U, Y_u)$. This is because, the value may have been obtained from L2 above.
   - If the tuple (m, U, Y_u, λ) is in L1 and $e(P, \lambda) = e(U, Y_u)$, then besides using the existing result of L1 as the value for $H_1(m, U, Y_u, \lambda)$, the value of λ should also be used to update the corresponding entry in L2.
   - If the tuple $(m, U, Y_u, \top)$ is in L1, then the existing result in L1 will be used as the value for $H_1(m, U, Y_u, \top)$. Also if the value of λ can be obtained from L2, then the entry $(m, U, Y_u, \top)$ in L1 should also be updated to (m, U, Y_u, λ).

3. $\mathcal{B}$ checks if $e(P, V) = e(H_1(m, U, Y_u, \lambda), Y_S)$ holds.

   - If the equation holds and $e(P, \lambda) = e(U, Y_u)$, $(m, (U, Y_u, \lambda, V), Y_S)$ are returned as the message–signature pair and the sender's public key.
   - If the equation holds but $\lambda = \top$, then $\mathcal{B}$ halts with failure.
   - Otherwise, the symbol "$\perp$" is returned for rejection.

After completing the first stage of the game, $\mathcal{A}$ chooses two n-bit plaintexts m_0 and m_1 together with a sender's private key x_S^*, and requests $\mathcal{B}$ for a challenge ciphertext built under the receiver's challenging public key Y_u. Suppose the associated signer's public key is $Y_S^* = x_S^* P$.

$\mathcal{B}$ sets the challenge ciphertext to $\sigma^* = (U^*, Z^*)$ where $U^* = aP$ and Z^* is randomly drawn from $\{0,1\}^n \times \mathbb{G}_1^2$. $\mathcal{B}$ also randomly picks $\check{b} \leftarrow 1/0$, and updates L1 by adding in $(m_{\check{b}}, aP, Y_u, \top)$, randomly picking $t^* \leftarrow \mathbb{Z}_q$, and setting the result of $H_1(m_{\check{b}}, aP, Y_u, \top)$ to t^*P. Note that this entry will only be added in L1 if it is not in L1 yet. Similarly, L2 will also be updated with $(aP, Y_u, \top)$ and the value of $H_2(aP, Y_u, \top)$ is set to $Z^* \oplus (m_{\check{b}}\|Y_S^*\|t^*Y_S^*)$. Let $V^* = t^*Y_S^*$.

After that, $\mathcal{B}$ answers $\mathcal{A}$'s queries as in the first stage. If $\mathcal{A}$ queries H_1 or H_2 with (aP, Y_u, λ) such that $e(aP, Y_u) = e(\lambda, P)$, then $\mathcal{B}$ outputs λ and halts. If $\mathcal{A}$ halts without making this query, $\mathcal{B}$ outputs a random point in $\mathbb{G}_1$ and halts.

Analysis. The running time of $\mathcal{B}$ is in polynomial of $\mathcal{A}$'s running time. To see that the simulated game is computationally indistinguishable from a real game, we note that H_1, H_2 and signcryption oracle are simulated perfectly. For de-signcryption queries, except the following case, are carried out perfectly too.

The exceptional case is at step 3 of the de-signcryption oracle simulation above when $(U, Y_u, \top)$ is in L2 and $(m, U, Y_u, \top)$ is in L1, while $e(P, V) = e(H_1(m, U, Y_u, \top), Y_S)$ where $m\|Y_S\|V = Z \oplus H_2(U, Y_u, \top)$ (i.e. the equation at

step 3 holds). This case implies that $\mathcal{A}$ has never queried H_1 on (m, U, Y_u, λ) nor H_2 on (U, Y_u, λ) such that $e(P, \lambda) = e(U, Y_u)$, while $Z_3 = W_3 \oplus V$ where $Z = (Z_1, Z_2, Z_3) \in \{0,1\}^n \times \mathbb{G}_1^2$ and $(W_1, W_2, W_3) = H_2(U, Y_u, \top) \in \{0,1\}^n \times \mathbb{G}_1^2$. Also note that the de-signcryption oracle would never leak any of $H_1(m, U, Y_u, \top)$ and W_3 to $\mathcal{A}$ at step 3 of the de-signcryption oracle simulation above, since $\mathcal{B}$ either fails or rejects. Therefore, we have

$$\Pr[Z_3 = W_3 \oplus V] \leqslant p/|\mathbb{G}_1| = p2^{-poly(k)},$$

where p is the maximum number of de-signcryption queries made by $\mathcal{A}$ and $poly(\cdot)$ is some polynomial function. Hence with probability at least $1 - p2^{-poly(k)}$, $\mathcal{B}$ does not fail and carries out the simulation perfectly.

Let $\mathbf{E}$ be the event that (aP, Y_u, aY_u) is queried on H_1 or H_2. $\bar{\mathbf{E}}$ denotes the event that (aP, Y_u, aY_u) is not queried on H_1 or H_2. Note that $\mathcal{B}$ solves the CDH problem instance in event $\mathbf{E}$.

We claim that for event $\bar{\mathbf{E}}$, $\mathcal{A}$ does not have any advantage in winning the game over random guessing. Let $V_{\tilde{b}} = x_S^* H_1(m_{\tilde{b}}, aP, Y_u, aY_u)$. Then $\sigma^* = (aP, Z^*)$ is the signcryption of $m_{\tilde{b}}$ if we have

$$(m_{\tilde{b}} \| Y_S^* \| V_{\tilde{b}}) = Z^* \oplus H_2(aP, Y_u, aY_u).$$

If we focus on the output portion of $H_2(aP, Y_u, aY_u)$ that is corresponding to $V_{\tilde{b}}$, we can see that, in event $\bar{\mathbf{E}}$, although $(aP, Y_u, \top)$ is in L2 and $(m_{\tilde{b}}, aP, Y_u, \top)$ is in L1, as argued above, $\mathcal{B}$ does not leak any information of these values to $\mathcal{A}$. Also due to the randomness assumption of H_1 and H_2, $\mathcal{A}$ does not have any advantage in determining the oracle return of $H_1(m_{\tilde{b}}, aP, Y_u, aY_u)$ and the output portion of $H_2(aP, Y_u, aY_u)$ corresponding to $V_{\tilde{b}}$ in the equation above. Hence, $\Pr[\mathcal{A} \text{ wins the game } |\bar{\mathbf{E}}] = \frac{1}{2}$. From the assumption,

$$\Pr[\mathcal{A} \text{ wins the game}] = \frac{1}{2} + \rho(k) \leqslant \Pr[\mathbf{E}] + \frac{1}{2}(1 - \Pr[\mathbf{E}]),$$

where ρ is $\mathcal{A}$'s non-negligible advantage in winning the game defined in Definition 2.2 and k is the system-wide security parameter. Therefore, $\Pr[\mathbf{E}] \geqslant 2\rho(k)$. We have $\Pr[\mathbf{E} \wedge \mathcal{B} \text{ does not fail}] \geqslant 2(1 - p2^{-poly(k)})\rho(k)$. □

Theorem 4.2. *Let k be the security parameter. Under the random oracle model, if there exists a PPT algorithm which can break the SC-EUF-CMA security of the proposed scheme with advantage at least $\rho(k)$, then there exists a PPT algorithm which can solve the Computational Diffie–Hellman Problem with probability at least $(1-p2^{-poly(k)})(1-q_1 q_S 2^{-poly(k)})\rho(k)$ where $poly(\cdot)$ is some polynomial, and p, q_1, q_S are the maximum number of de-signcryption, H_1 and signcryption queries made in the model of SC-EUF-CMA.*

Proof. We prove it also by contradiction, we assume that there exists a forger $\mathcal{F}$ who can win the game stated in Definition 2.3. In the following, we construct an algorithm $\mathcal{B}$ that can solve the CDH problem in $\mathbb{G}_1$.

Suppose $\mathcal{B}$ is given a random instance of the CDH problem $(P, aP, bP) \in \mathbb{G}_1^3$, $\mathcal{B}$ runs $\mathcal{F}$ as a subroutine to find abP. $\mathcal{B}$ sets up a simulated environment of SC-EUF-CMA as follows:

$\mathcal{B}$ gives bP to $\mathcal{F}$ as the challenge public key Y_u.

$\mathcal{B}$ maintains two lists L1 and L2 to simulate the hash oracles H_1 and H_2, respectively. In each entry of the lists, it keeps the query and the corresponding return of the oracle. Hash oracles H_2 is simulated as in the proof of Theorem 4.1.

When a hash query $H_1(m, P_1, P_2, P_3)$ is asked by $\mathcal{F}$, $\mathcal{B}$ first checks if the query tuple (m, P_1, P_2, P_3) is already in L1. If it exists, the existing result in L1 is returned. If it does not exist but $e(P_1, P_2) = e(P, P_3)$ and $(P_1, P_2, \top)$ is in L1, where "$\top$" is a special symbol, then $\mathcal{B}$ replaces "$\top$" in the entry with P_3 and returns the existing result in the entry. For all other cases, $\mathcal{B}$ randomly chooses $t \leftarrow \mathbb{Z}_q$ and returns $t(aP)$ to $\mathcal{F}$. The query tuple and return value are then saved in L1. For enabling the retrieval of t possibly in some later time of the simulation, the value is also saved in L1.

For a signcryption query on a message m with a receiver's public key Y_{R} both chosen by $\mathcal{F}$, $\mathcal{B}$ first checks if $Y_{\mathrm{R}} \in \mathbb{G}_1$. If it is incorrect or $Y_{\mathrm{R}} = Y_u$, $\mathcal{B}$ returns the symbol "$\perp$" for rejection. Otherwise, $\mathcal{B}$ picks a random $r \leftarrow \mathbb{Z}_q$ and computes $U = rP$. If the tuple $m, U, Y_{\mathrm{R}}, rY_{\mathrm{R}}$ is already in L1 and the corresponding hash value is in the form of taP for some $t \in \mathbb{Z}_q$, $\mathcal{B}$ fails and aborts. Note that, if such a case happens, $m, U, Y_{\mathrm{R}}, rY_{\mathrm{R}}$ can only occur in an H_1 hash query before. Otherwise, $\mathcal{B}$ selects a random $t' \leftarrow \mathbb{Z}_q$ and returns $t'P$ as the value of $H_1(m, U, Y_{\mathrm{R}}, rY_{\mathrm{R}})$. The query tuple, oracle return and the value of t' are then saved in L1. After obtaining t' such that $t'P = H_1(m, U, Y_{\mathrm{R}}, rY_{\mathrm{R}})$, $\mathcal{B}$ computes $V = t'(Y_u)$ which is equal to $bH_1(m, U, Y_{\mathrm{R}}, rY_{\mathrm{R}})$. $\mathcal{B}$ then simulates H_2 as in the proof of Theorem 4.1 for obtaining $H_2(U, Y_{\mathrm{R}}, rY_{\mathrm{R}})$, and computes the result ciphertext $\sigma = (U, Z)$ where $Z = (m \| Y_u \| t'(Y_u)) \oplus H_2(U, Y_{\mathrm{R}}, rY_{\mathrm{R}})$.

When $\mathcal{F}$ performs a De-signcrypt(σ, sk_u) query, where $\sigma = (U, Z)$, the following steps are carried out:

1. $\mathcal{B}$ looks for a tuple of the form (U, Y_u, λ) in L2 such that $e(P, \lambda) = e(U, Y_u)$ or $\lambda = \top$.
   - If the tuple (U, Y_u, λ) is not in L2, $\mathcal{B}$ adds a new entry into L2 by storing $(U, Y_u, \top)$ as the query tuple and a value randomly drawn from the range of H_2 as the oracle return. The special symbol "$\top$" is used as a marker for denoting that the real value should be the solution of the CDH problem instance (U, Y_u). This step ensures that the value of $H_2(U, Y_u, \lambda)$ is *fixed* before σ is de-signcrypted.
   - If the tuple (U, Y_u, λ) is already in L2, the existing result will be used as the value of $H_2(U, Y_u, \lambda)$.

2. $\mathcal{B}$ computes $m\|Y_S\|V = Z \oplus H_2(U, Y_u, \lambda)$ and looks for a tuple of the form (m, U, Y_u, λ) in L1 such that $e(P, \lambda) = e(U, Y_u)$ or $\lambda = \top$.
   - If the tuple (m, U, Y_u, λ) is not in L1, $\mathcal{B}$ adds a new entry into L1 by storing (m, U, Y_u, λ) as the query tuple and setting $t'(aP)$ as the oracle return, where $t' \leftarrow \mathbb{Z}_q$. Note that λ can be a value such that $e(P, \lambda) = e(U, Y_u)$. This is because, the value may have been obtained from L2 above.
   - If the tuple (m, U, Y_u, λ) is in L1 and $e(P, \lambda) = e(U, Y_u)$, then besides using the existing result of L1 as the value for $H_1(m, U, Y_u, \lambda)$, the value of λ should also be used to update the corresponding entry in L2.
   - If the tuple $(m, U, Y_u, \top)$ is in L1, then the existing result in L1 will be used as the value for $H_1(m, U, Y_u, \top)$. Also if the value of λ can be obtained from L2, then the entry $(m, U, Y_u, \top)$ in L1 should also be updated to (m, U, Y_u, λ).
3. $\mathcal{B}$ checks if $e(P, V) = e(H_1(m, U, Y_u, \lambda), Y_S)$ holds.
   - If the equation holds and $e(P, \lambda) = e(U, Y_u)$, $(m, (U, Y_u, \lambda, V), Y_S)$ are returned as the message–signature pair and the sender's public key.
   - If the equation holds but $\lambda = \top$, then $\mathcal{B}$ halts with failure.
   - Otherwise, the symbol "$\perp$" is returned for rejection.

When $\mathcal{F}$ produces a ciphertext $\sigma = (U, Z)$ and a receiver's key pair (x_R, Y_R), $\mathcal{B}$ de-signcrypts the ciphertext as the simulation of de-signcrypt query above. If the forgery is valid, which means $e(Y_u, H_1(m, U, Y_R, \lambda)) = e(P, V) = e(bP, taP)$, $\mathcal{B}$ gets $V = tabP$ and solves the CDH problem from computing $abP = t^{-1}V$. Then $\mathcal{B}$ outputs abP and halts.

Analysis. From the simulation of the de-signcrypt query above, we can see that there must be an entry in L1 for $H_1(m, U, Y_R, \lambda)$. We also claim that the corresponding oracle return in the entry must be in the form $t(aP)$ for some $t \in \mathbb{Z}_q$, which can be retrieved from L1. Notice that if $H_1(m, U, Y_R, \lambda)$ is equal to $t'P$, which is generated in a signcryption query, the values of Z would also have been determined in that signcryption query, which contradicts the restriction of the game defined in Definition 2.3.

Obviously, The running time of $\mathcal{B}$ is also in polynomial of $\mathcal{F}$'s running time. As proven in Theorem 1, the simulated game is also computationally indistinguishable from a real game if $\mathcal{B}$ does not fail during the simulation. There are two cases in which $\mathcal{B}$ may fail: the first case is that in a signcryption query, $\mathcal{B}$ happens to choose $r \in \mathbb{Z}_q$ such that $m, U = rP, Y_R, rY_R$ is queried in an H_1 query, and it happens with probability at most $q_1 q_S / |\mathbb{G}_1|$, so $\mathcal{B}$ does not fail with probability at lease $1 - q_1 q_S 2^{-poly(k)}$. Also, as in Theorem 1, $\mathcal{B}$ does not fail in de-signcryption queries with probability at lease $1 - p2^{-poly(k)}$.

So, if $\mathcal{B}$ does not fail and $\mathcal{F}$ can win the game, then $\mathcal{B}$ can solve the CDH problem. Hence, $\Pr[\mathcal{F}$ wins the game $\wedge$ $\mathcal{B}$ does not fail$] \geqslant (1 - p2^{-poly(k)})(1 - q_1 q_S 2^{-poly(k)})\rho(k)$. □

Theorem 4.3. *Let k be a security parameter. Under the random oracle model, if there exists a PPT algorithm which can break the SC-ANON-CCA security of the proposed signcryption scheme with advantage at least $\rho(k)$, then there exists a PPT algorithm which can solve the Computational Diffie–Hellman Problem with probability at least $\frac{4}{3}(1 - p2^{-poly(k)})\rho(k)$ where poly$(\cdot)$ is some polynomial and ρ is the maximum number of de-signcryption queries made in the model of SC-ANON-CCA.*

Proof. The proof follows that of Theorem 1. Suppose $\mathcal{B}$ is given (aP, cP) as a random instance of the CDH problem, $\mathcal{B}$ runs $\mathcal{D}$ to find the solution acP.

$\mathcal{B}$ picks two random elements $x, y \in \mathbb{Z}_q$ and sets the two challenge public keys as $pk_{\mathrm{R},0} = x(cP)$ and $pk_{\mathrm{R},1} = y(cP)$. $\mathcal{B}$ then simulates all the hash queries, signcryption queries and de-signcryption queries as in the proof of Theorem 1.

After the completion of the first stage, $\mathcal{D}$ chooses two private keys $sk_{S,0}, sk_{S,1}$ and a plaintext $m \in \{0,1\}^n$ and requests a challenge ciphertext built under $sk_{S,b}$ and $pk_{\mathrm{R},b'}$ where $b, b' \overset{R}{\leftarrow} \{0,1\}$.

$\mathcal{B}$ sets the challenge ciphertext to $\sigma' = (U', Z')$ where $U' = aP$ and Z' is randomly drawn from $\{0,1\}^n \times \mathbb{G}_1^2$. $\mathcal{B}$ updates L1 by adding in $(m, aP, pk_{\mathrm{R},0}, \top)$ and $(m, aP, pk_{\mathrm{R},1}, \top)$, randomly picking $t', t'' \in \mathbb{Z}_q$ and setting $t'P, t''P$ as the result of $H_1(m, aP, pk_{\mathrm{R},0}, \top)$ and $H_1(m, aP, pk_{\mathrm{R},1}, \top)$ respectively. Note that those entries will only be added in L1 if they are not in L1 yet. Similarly, L2 will also be updated with $(aP, pk_{\mathrm{R},0}, \top)$ and $(aP, pk_{\mathrm{R},1}, \top)$, the value of $H_2(aP, pk_{\mathrm{R},0}, \top)$ and $H_2(aP, pk_{\mathrm{R},1}, \top)$ are set to $Z' \oplus (m \| Y_s' \| t'Y_s')$ and $Z' \oplus (m || Y_s' || t''Y_s')$ respectively, where $Y_s' = pk_{S,b}$.

$\mathcal{B}$ answers $\mathcal{D}$'s queries as in the first stage. If $\mathcal{D}$ queries H_1 or H_2 with $(aP, pk_{\mathrm{R},0}, \lambda)$ such that $e(aP, pk_{\mathrm{R},0}) = e(P, \lambda)$, $\mathcal{B}$ halts and outputs $x^{-1}\lambda$; If $\mathcal{D}$ queries H_1 or H_2 with $(aP, pk_{\mathrm{R},1}, \lambda)$ such that $e(aP, pk_{\mathrm{R},1}) = e(P, \lambda)$, $\mathcal{B}$ halts and outputs $y^{-1}\lambda$. $\mathcal{B}$ output a random point in $\mathbb{G}_1$ and halts, if $\mathcal{D}$ halts without making those queries.

Analysis. Obviously, the running time of $\mathcal{B}$ is in polynomial of $\mathcal{D}$'s running time. The simulated game is computationally indistinguishable from a real game with the failure probability $p2^{-poly(k)}$ as proven in Theorem 1. In the following, we analyze $\mathcal{B}$'s success rate.

Let $\mathbf{E}$ be the event that $(aP, pk_{\mathrm{R},0}, a(pk_{\mathrm{R},0}))$ or $(aP, pk_{\mathrm{R},1}, a(pk_{\mathrm{R},1}))$ has been queried on H_1 or H_2. $\bar{\mathbf{E}}$ denotes event $\mathbf{E}$ does not happen. Note that $\mathcal{B}$ solves the CDH problem instance in event $\mathbf{E}$.

We claim that for event $\bar{\mathbf{E}}$, $\mathcal{D}$ does not have any advantage in winning the game over random guessing: Let $V_{(b,b')} = sk_{S,b} H_1(m, aP, pk_{\mathrm{R},b'}, apk_{\mathrm{R},b'})$. Then $\sigma' = (aP, Z')$ is the signcryption of m under $sk_{S,b}$ and $pk_{\mathrm{R},b'}$ if we have

$$m \| sk_{S,b} P \| V_{b,b'} = Z' \oplus H_2(aP, pk_{\mathrm{R},b'}, apk_{\mathrm{R},b'}).$$

In event $\bar{\mathbf{E}}$, since H_1 and H_2 are not queried with $(aP, pk_{\mathrm{R},0}, a(pk_{\mathrm{R},0}))$ or $(aP, pk_{\mathrm{R},1}, a(pk_{\mathrm{R},1}))$, due to the random oracle assumption, $\mathcal{D}$ does not have any

Table 1
Efficiency comparison for signcryption schemes with key privacy

Scheme	Security	Efficiency: $\mathbb{G}_1$ Mul, Signcrypt	$\mathbb{G}_2$ Exp, MapToPoint, De-signcrypt	Pairing verify
Boyen ID-SC [9]	C, N, A	3, 1, 2, 1	2, 0, 3, 4	1, 0, 1, 2
LQ [14]	–, N, –	3, 0, 1, 0	1, 0, 1, 2	0, 0, 1, 2
YWD [23]	–, N, –	3, 0, 1, 0	1, 0, 1, 2	0, 0, 1, 2
This paper	C, N, A	3, 0, 1, 0	1, 0, 1, 2	0, 0, 1, 2

advantage in determining the oracle returns of H_1 and H_2 on these query tuples. Hence, $\Pr[\mathcal{D} \text{ wins the game } |\bar{\mathbf{E}}] = \frac{1}{4}$. From the assumption,

$$\Pr[\mathcal{D} \text{ wins the game}] = \frac{1}{4} + \rho(k) \leqslant \Pr[\mathbf{E}] + \frac{1}{4}(1 - \Pr[\mathbf{E}]),$$

where ρ is $\mathcal{D}$'s non-negligible advantage in winning the game defined in Definition 2.4 and k is the system-wide security parameter. Therefore, $\Pr[\mathbf{E}] \geqslant \frac{4}{3}\rho(k)$.

We have $\Pr[\mathbf{E} \wedge \mathcal{B} \text{ does not fail}] \geqslant \frac{4}{3}(1 - p2^{-poly(k)})\rho(k)$. □

4.3. Performance

We compare our scheme with other signcryption schemes with key privacy in Table 1. We consider the costly operations which include point scalar multiplication on $\mathbb{G}_1$ ($\mathbb{G}_1$ Mul), exponentiation on $\mathbb{G}_2$ ($\mathbb{G}_2$ Exp), Map-To-Point hash operation [6] (MapToPoint) and pairing operation (Pairing).

In Table 1, C denotes confidentiality, N denotes non-repudiation and A denotes ciphertext anonymity. We can see that our scheme solves the security issues of the LQ and YWD schemes but maintaining their high efficiency.

5. Variation and extension

Rivest, Shamir and Tauman defined the notion of ring signature in [18]. A ring signature allows a party to sign messages on behalf of a group without leaking its identity. To consider key privacy for signcryption schemes one step further, we define and construct a ring signcryption scheme in this paper. In a ring signcryption scheme, after decrypting the ciphertext, the recipient will obtain a ring signature instead of a standard signature, and the recipient cannot tell which party inside the ring generated the ciphertext.

Before defining ring signcryption, we first give a variation of our signcryption scheme. This variant is the basis of our ring signcryption construction.

5.1. Signcryption variation

Keygen. The same as the original scheme.

Signcrypt. To signcrypt a message $m \in \{0,1\}^n$ for receiver R, sender S carries out the following steps:

1. Pick a random $r \leftarrow \mathbb{Z}_q$ and compute $U = rP$.
2. Compute $V = x_S^{-1} H_1(m, U, Y_R, rY_R)$ and $Z = (m \| Y_S \| V) \oplus H_2(U, Y_R, rY_R)$.

The ciphertext is $\sigma = (U, Z)$.

De-signcrypt. When a ciphertext $\sigma = (U, Z)$ is received, receiver R performs the following steps:

1. Compute $D = x_R U$.
2. Compute $(m \| Y_S \| V) = Z \oplus H_2(U, Y_R, D)$.
3. If $Y_S \notin \mathbb{G}_1$, output `reject`. Otherwise, compute $H = H_1(m, U, Y_R, D)$ and check $e(Y_S, V) \stackrel{?}{=} e(P, H)$.
4. If the check passes, output $\langle m, (U, Y_R, D, V), Y_S \rangle$; otherwise, output `reject`.

Verify. For a message–signature pair $(m, (U, Y_R, D, V))$ and a signer's public key Y_S, the algorithm checks if $e(Y_S, V) = e(P, H_1(m, U, Y_R, D))$. If the condition holds, it outputs `true`. Otherwise, it outputs `false`.

The differences between the variation and the original scheme are in the generation of V and the signature verification. In the variation, V is generated as $x_S^{-1} H_1(m, U, Y_R, rY_R)$ rather than $x_S H_1(m, U, Y_R, rY_R)$, and the signature verification becomes $e(Y_S, V) = e(P, H)$ rather than $e(Y_S, H) = e(P, V)$.

Note that we do not give a security proof here for the variation as we only make use of it as an intermediate step towards the construction of our final ring signcryption scheme. In addition, we can see that the complete security proofs of the ring signcryption scheme below will have also covered the confidentiality (Definition 2.2) and unforgeability (Definition 2.3) of this variation, when we consider the case that the ring size is equal to one.

In the next section, we construct a ring signcryption scheme based on this variation.

5.2. Ring signcryption

A ring signcryption scheme is a quadruple of probabilistic polynomial time (PPT) algorithms (**Keygen**, **Signcrypt**, **De-signcrypt**, **Verify**).

$(sk, pk) \leftarrow$ **Keygen**(1^k) is the key generation algorithm which takes a security parameter $k \in \mathbb{N}$ and generates a private/public key pair (sk, pk).

$\sigma \leftarrow$ **Signcrypt**$(m, pk_S = (pk_S^1, pk_S^2, \ldots, pk_S^j), sk_S^i, pk_R)$ takes a message m drawn from the message space M, a group of distinct public keys $pk_S = (pk_S^1, pk_S^2, \ldots, pk_S^j)$ and a private key sk_S^i whose corresponding public key $pk_S^i \in pk_S$, and a public key pk_R such that $pk_R \notin pk_S$, outputs a ciphertext σ.

$(m, s, pk_S = (pk_S^1, pk_S^2, \ldots, pk_S^j))/\texttt{reject} \leftarrow$ **De-signcrypt**(σ, sk_R) takes σ and a private key sk_R, outputs either a triple $(m, s, pk_S = (pk_S^1, pk_S^2, \ldots, pk_S^j))$ where $m \in M$, s is a ring signature and pk_S is a group of distinct public keys, or `reject` which indicates the failure of de-signcryption.

$\texttt{true/false} \leftarrow$ **Verify**$(m, s, pk_S = (pk_S^1, pk_S^2, \ldots, pk_S^j))$ takes $m \in M$, a signature s and a group of distinct public keys $pk_S = (pk_S^1, pk_S^2, \ldots, pk_S^j)$, outputs `true` for a valid ring signature or `false` for an invalid ring signature.

Our proposed ring signcryption scheme

Keygen. The same as the original signcryption scheme.

Signcrypt. To signcrypt a message $m \in \{0,1\}^n$ for receiver R, sender $\hat{S}$ carries out the following steps:

1. Choose a random index $1 \leqslant i \leqslant j$ where j is the size of the sender group, and choose $j-1$ distinct public keys $pk_S^1, \ldots, pk_S^{i-1}, pk_S^{i+1}, \ldots, pk_S^j$. Set the sender group as $pk_S = (pk_S^1, \ldots, pk_S^{i-1}, pk_{\hat{S}}, pk_S^{i+1}, \ldots, pk_S^j)$.
2. Pick a random $r \leftarrow \mathbb{Z}_q$ and compute $U = rP$. Choose random $a_\omega \leftarrow \mathbb{Z}_q$ for $1 \leqslant \omega \leqslant j$ and $\omega \neq i$.
3. Compute $H = H_1(m, U, pk_R, rpk_R)$ and set $s_i = x_{\hat{S}}^{-1}(H - \sum_{\omega \neq i} a_\omega pk_S^\omega)$. For $\omega \neq i$, set $s_\omega = a_\omega P$.
4. Set $V = (s_1, s_2, \ldots, s_j)$ and compute $Z = (m \| V) \oplus H_2(U, pk_R, rpk_R)$.

The ciphertext is $\sigma = (pk_S, U, Z)$.

De-signcrypt. When a ciphertext $\sigma = (pk_S, U, Z)$ is received, receiver R performs the following steps:

1. Compute $D = x_R U$.
2. Compute $(m \| V) = Z \oplus H_2(U, pk_R, D)$.
3. Compute $H = H_1(m, U, pk_R, D)$ and check $\prod_{\omega=1}^{j} e(s_\omega, pk_S^\omega) \stackrel{?}{=} e(P, H)$.
4. If the check passes, output $\langle m, (U, pk_R, D, V), pk_S \rangle$; otherwise, output `reject`.

Verify. For a message–signature pair $(m, (U, pk_R, D, V))$ and a group of j public keys $pk_S = (pk_S^1, \ldots, pk_S^j)$, the algorithm checks if $\prod_{\omega=1}^{j} e(s_\omega, pk_S^\omega) = e(P, H)$. If the condition holds, it outputs `true`. Otherwise, it outputs `false`.

5.3. Security analysis

We also consider confidentiality, unforgeability and key privacy for ring signcryption. However, we separate sender anonymity and receiver anonymity here, for the

former, we consider a notion similar to the signer ambiguity for ring signature [18], while for the latter, we define it similarly to Definition 2.4.

Definition 5.1 (RSC confidentiality). A ring signcryption scheme is semantically secure against insider chosen ciphertext attack (RSC-IND-CCA) if no PPT adversary has a non-negligible advantage in the following game:

1. The challenger runs **Keygen** to generate a key pair $(sk_{\mathrm{U}}, pk_{\mathrm{U}})$. sk_{U} is kept secret while pk_{U} is given to adversary $\mathcal{A}$.
2. In the first stage, $\mathcal{A}$ makes a number of queries to the following oracles:
   (a) Signcryption oracle: $\mathcal{A}$ prepares a message $m \in M$, a polynomial j which indicates the size of the signer ring, and a number i which indicates the position for pk_{U}, and a public key pk_{R}, and queries the signcryption oracle (simulated by the challenger). If $pk_{\mathrm{R}} \neq pk_{\mathrm{U}}$, the challenger runs **Keygen** to generate $j-1$ key pairs, and runs $\sigma \leftarrow$ **Signcrypt**$(m, pk_{\mathrm{S}} = (pk_{\mathrm{S}}^1, pk_{\mathrm{S}}^2, \ldots, pk_{\mathrm{S}}^j), sk_{\mathrm{S}}^i, pk_{\mathrm{R}})$ where $(pk_{\mathrm{S}}^i, sk_{\mathrm{S}}^i) = (pk_{\mathrm{U}}, sk_{\mathrm{U}})$ to generate a ciphertext σ and returns it to $\mathcal{A}$. Otherwise, a symbol "$\perp$" is returned for rejection.
   (b) De-signcryption oracle: $\mathcal{A}$ produces a ciphertext σ and queries for the result of **De-signcrypt**$(\sigma, sk_{\mathrm{U}})$. The result is made of a message, a ring signature and a group of public keys if the de-signcryption is successful. Otherwise, a symbol "$\perp$" is returned for rejection.

   These queries can be asked adaptively: each query may depend on the answers of previous ones.
3. $\mathcal{A}$ produces two plaintexts $m_0, m_1 \in M$ of equal length, $j-1$ valid public keys, a valid public/private key pair $(pk_{\hat{S}}, sk_{\hat{S}})$ and an index i which indicates the position of $pk_{\hat{S}}$ in $pk_{\mathrm{S}} = (pk_{\mathrm{S}}^1, \ldots, pk_{\mathrm{S}}^j)$. The challenger flips a coin $\check{b} \overset{R}{\leftarrow} \{0,1\}$ and computes a signcryption $\sigma^* =$ **Signcrypt**$(m_{\check{b}}, pk_{\mathrm{S}}, sk_{\hat{S}}, pk_{\mathrm{U}})$ of $m_{\check{b}}$. σ^* is sent to $\mathcal{A}$ as a challenge ciphertext.
4. $\mathcal{A}$ makes a number of new queries as in the first stage with the restriction that it cannot query the de-signcryption oracle with σ^*.
5. At the end of the game, $\mathcal{A}$ outputs a bit b' and wins if $b' = \check{b}$.

$\mathcal{A}$'s advantage is defined as $Adv^{rsc-ind-cca}(\mathcal{A}) = \Pr[b' = \check{b}] - \frac{1}{2}$ and the probability that $b' = \check{b}$ is called the probability that $\mathcal{A}$ wins the game.

Definition 5.2 (RSC unforgeability). A ring signcryption scheme is existentially unforgeable against chosen-message insider attack (RSC-EUF-CMA) if no PPT forger has a non-negligible advantage in the following game for any polynomial j:

1. The challenger runs **Keygen** to generate j key pairs $(sk_1, pk_1), \ldots, (sk_j, pk_j)$. The public keys are given to forger $\mathcal{F}$.

2. The forger $\mathcal{F}$ has access to a ring signcryption oracle: $\mathcal{F}$ provides a message m, a valid receiver's public key pk_{R} such that $pk_{\mathrm{R}} \neq pk_i$ $(1 \leqslant i \leqslant j)$, and an index ω, the challenger generates a ciphertext $\sigma \leftarrow$ **Signcrypt**$(m, pk_{\mathrm{S}} = (pk_1, pk_2, \ldots, pk_j), sk_\omega, pk_{\mathrm{R}})$ and returns it to $\mathcal{F}$.
3. $\mathcal{F}$ produces a ciphertext σ and a valid key pair $(sk_{\mathrm{R}}, pk_{\mathrm{R}})$ and wins the game if
   (a) **De-signcrypt**$(\sigma, sk_{\mathrm{R}})$ returns a tuple $(m, s, (pk_1, pk_2, \ldots, pk_j))$ such that `true` $\leftarrow$ **Verify**$(m, s, (pk_1, pk_2, \ldots, pk_j))$, and
   (b) σ is not the output of the signcryption oracle.

Definition 5.3 (RSC perfect sender anonymity). Consider the following game against any unbounded adversary $\mathcal{A}$:

> $\mathcal{A}$ runs **Keygen** to generate $j+1$ key pairs $(sk_1, pk_1), \ldots, (sk_j, pk_j), (sk_{\mathrm{R}}, pk_{\mathrm{R}})$ where j is any polynomial of k, and gives them to the challenger, together with a message m. The challenger randomly selects $1 \leqslant i \leqslant j$ and generates $\sigma \leftarrow$ **Signcrypt**$(m, pk_{\mathrm{S}} = (pk_1, pk_2, \ldots, pk_j), sk_i, pk_{\mathrm{R}})$.

The adversary's goal is to guess the real signer among $pk_{\mathrm{S}} = (pk_1, pk_2, \ldots, pk_j)$. A ring signcryption scheme is sender anonymous if the probability that $\mathcal{A}$ guesses correctly is at most $1/j$.

Definition 5.4 (RSC receiver anonymity). A ring signcryption scheme has receiver anonymity if no PPT distinguisher has a non-negligible advantage in the following game:

1. The challenger generates two distinct public key pairs $(sk_{\mathrm{R},0}, pk_{\mathrm{R},0})$ and $(sk_{\mathrm{R},1}, pk_{\mathrm{R},1})$ using **Keygen**, and gives $pk_{\mathrm{R},0}$ and $pk_{\mathrm{R},1}$ to the distinguisher $\mathcal{D}$.
2. In the first stage, $\mathcal{D}$ adaptively makes a number of ring signcrypt and de-signcrypt as in the RSC Confidentiality game, with either $sk_{\mathrm{R},0}$ or $sk_{\mathrm{R},1}$ involved.
3. After completing the first stage, $\mathcal{D}$ outputs a message m, $j-1$ valid public keys, a valid public/private key pair $(pk_{\hat{S}}, sk_{\hat{S}})$ and an index i which indicates the position of $pk_{\hat{S}}$ in $pk_{\mathrm{S}} = (pk_{\mathrm{S}}^1, \ldots, pk_{\mathrm{S}}^j)$.
4. The challenger then flips a coin $b' \overset{R}{\leftarrow} \{0, 1\}$ and computes a challenge ciphertext $\sigma =$ **Signcrypt**$(m, pk_{\mathrm{S}} = (pk_1, \ldots, pk_j), sk_i, pk_{\mathrm{R},b'})$ and sends it to $\mathcal{D}$.
5. $\mathcal{D}$ adaptively makes a number of new queries as above with the restriction that it is not allowed to ask the de-signcryption oracle of the challenge ciphertext σ.
6. At the end of the game, $\mathcal{D}$ outputs a bit d' and wins the game if $d' = b'$.

$\mathcal{D}$'s advantage is defined as $Adv^{r-anon}(\mathcal{D}) = \Pr[d' = b'] - \frac{1}{2}$.

Theorem 5.5. *Our ring signcryption scheme achieves perfect sender anonymity.*

The proof follows the same probability argument as in the proof of Theorem 5.1 of [7].

Theorem 5.6. *If there exists a PPT algorithm which can break the RSC-IND-CCA security of the proposed ring signcryption scheme with advantage $\rho(k)$, then there exists a PPT algorithm which can solve the Computational Diffie–Hellman Problem with probability at least $2(1 - p2^{-poly(k)})\rho(k)$ where $poly(\cdot)$ is some polynomial and p is the maximum number of de-signcryption queries made in the model of RSC-IND-CCA.*

Proof. Assume there exists an adversary $\mathcal{A}$, who makes at most p de-signcryption queries, wins the game in Definition 5.1 with non-negligible probability $\rho(k)$, we construct another algorithm $\mathcal{B}$ who solve the CDH problem with non-negligible probability.

$\mathcal{B}$ is given P, aP, bP and aims to compute abP, where $a, b \overset{R}{\leftarrow} \mathbb{Z}_q$. $\mathcal{B}$ sets $pk_{\mathrm{U}} = aP$ and gives it to $\mathcal{A}$.

Hash queries are handled similarly as in the proof of Theorem 4.1, except that for a hash query on H_1, $\mathcal{B}$ picks random $t \overset{R}{\leftarrow} \mathbb{Z}_q$, and returns taP.

When $\mathcal{A}$ asks a ring signcrypt query with m, j, i, pk_{R}, $\mathcal{B}$ picks $x_1, \ldots, x_{i-1}$, $x_{i+1}, \ldots, x_j$ at random from $\mathbb{Z}_q$ and sets $x_i = 1$. Then $\mathcal{B}$ sets $pk_{\mathrm{S}}^i = x_i aP$. $\mathcal{B}$ picks random $r \overset{R}{\leftarrow} \mathbb{Z}_q$, and computes $U = rP$. $\mathcal{B}$ simulates the H_1 oracle, the hash returned is taP for some t. Then $\mathcal{B}$ chooses random $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_j \overset{R}{\leftarrow} \mathbb{Z}_q$ and computes $a_i = t - \sum_{\omega \neq i} a_\omega x_\omega$, and computes $V = a_1 P, \ldots, a_j P$. $\mathcal{B}$ then simulates $H_2(U, pk_{\mathrm{R}}, rpk_{\mathrm{R}})$ and computes the ciphertext (pk_{S}, U, Z) where $Z = m \| V \oplus H_2(U, pk_{\mathrm{R}}, rpk_{\mathrm{R}})$. The de-signcryption queries are handled similarly as in the proof of Theorem 4.1.

After completing the first stage, $\mathcal{A}$ gives $m_0, m_1, (pk_{\mathrm{S}}^1, \ldots, pk_{\mathrm{S}}^{i-1}, pk_{\mathrm{S}}^{i+1}, \ldots, pk_{\mathrm{S}}^j), pk_{\hat{S}}, sk_{\hat{S}}, i$ to $\mathcal{B}$. $\mathcal{B}$ sets $U^* = bP$, and selects random $Z^* \overset{R}{\leftarrow} \{0,1\}^n \times \mathbb{G}_1^j$. $\mathcal{B}$ also randomly picks $\check{b} \overset{R}{\leftarrow} \{0,1\}$, and sets $H_1(m_{\check{b}}, bP, pk_{\mathrm{U}}, \top) = t^* aP$ for random $t^* \in \mathbb{Z}_q$. Here $\top$ denotes the (unknown) value of abP. Then $\mathcal{B}$ generates V^* according to the signcrypt algorithm and sets $H_2(bP, pk_{\mathrm{U}}, \top) = Z^* \oplus m_{\check{b}} \| V^*$. The challenging ciphertext is $\sigma^* = (pk_{\mathrm{S}}, U^*, Z^*)$.

After that, $\mathcal{B}$ answers $\mathcal{A}$'s queries as in the first stage. If $\mathcal{A}$ queries H_1 or H_2 with $(bP, pk_{\mathrm{U}}, \lambda)$ such that $e(bP, pk_{\mathrm{U}}) = e(P, \lambda)$, $\mathcal{B}$ outputs λ and halts. If $\mathcal{A}$ halts without making such a query, $\mathcal{B}$ outputs a random point in $\mathbb{G}_1$ and halts.

By following a similar security analysis as in the proof of Theorem 4.1, $\mathcal{B}$ breaks the CDH assumption with probability at least $2(1 - p2^{-poly(k)})\rho(k)$. □

Theorem 5.7. *If there exists a PPT algorithm which can break the RSC-EUF-CMA security of the proposed ring signcryption scheme with advantage $\rho(k)$, then there exists a PPT algorithm which can solve the Computational Diffie–Hellman Problem with probability at least $\frac{\rho(k)}{e(q_{\mathrm{S}}+1)}$ where q_{S} is the maximum number of ring signcrypt queries made in the model of RSC-EUF-CMA, and e is the base of the natural logarithm.*

Proof. Assume there is an adversary $\mathcal{A}$, which takes at most q_S ring signcrypt queries, wins the game in Definition 5.2 with non-negligible advantage $\rho(k)$, we construct another algorithm $\mathcal{B}$ which solves the CDH problem with non-negligible probability. $\mathcal{B}$ is given P, abP, aP and aims to compute bP, where $a, b \overset{R}{\leftarrow} \mathbb{Z}_q$.

Initially, $\mathcal{B}$ picks $x_2, \ldots, x_j$ at random from $\mathbb{Z}_q$ and sets $x_1 = 1$. $\mathcal{B}$ sets $pk_i = x_i aP$ for $1 \leqslant i \leqslant j$.

Hash queries are handled similarly as in the proof of Theorem 4.1, except that for a hash query on H_1, $\mathcal{B}$ flips a coin that shows 1 with probability p and 0 otherwise (p shall be determined later), then $\mathcal{B}$ pick random $t \overset{R}{\leftarrow} \mathbb{Z}_q$, and if the coin shows 0, $\mathcal{B}$ returns $tabP$, otherwise, it returns taP.

When $\mathcal{A}$ asks a ring signcrypt query with m, pk_R, i, $\mathcal{B}$ picks random $r \overset{R}{\leftarrow} \mathbb{Z}_q$, and computes $U = rP$. $\mathcal{B}$ simulates the H_1 oracle, if the coin flipped for this hash query shows 0, $\mathcal{B}$ fails and aborts, otherwise, the hash returned is taP for some t. Then $\mathcal{B}$ chooses random $a_2, \ldots, a_j \overset{R}{\leftarrow} \mathbb{Z}_q$ and computes $a_1 = t - \sum_{2 \leqslant \omega \leqslant j} a_\omega x_\omega$, and computes $V = a_1 P, \ldots, a_j P$. $\mathcal{B}$ then simulates $H_2(U, pk_R, rpk_R)$ and computes the ciphertext $(pk_S = (pk_1, \ldots, pk_j), U, Z)$ where $Z = m \| V \oplus H_2(U, pk_R, rpk_R)$.

Eventually $\mathcal{A}$ outputs a forgery $\sigma^* = (pk_S, U^*, Z^*)$ and a key pair (sk_R^*, pk_R^*). $\mathcal{B}$ decrypts the ciphertext by following the de-signcrypt algorithm. If the decryption is successful, $\mathcal{B}$ gets $m \| V = (s_1, \ldots, s_j)$. If the coin flipped for the H_1 query shows 1, $\mathcal{B}$ fails and aborts, otherwise, $\mathcal{B}$ outputs $t^{-1}(s_1 + s_2 x_2 + \cdots + s_j x_j)$.

Analysis. Due to the perfect sender anonymity of our ring signcryption scheme, the simulation is perfect if $\mathcal{B}$ does not abort the game. Also, given a valid forgery, the H_1 query with respect to $\mathcal{A}$'s forgery must not appear in any signcrypt query, since otherwise, (pk_S, U^*, Z^*) is the same as the ciphertext returned by that signcrypt query. $\mathcal{B}$'s success probability is $\rho(k) p^{q_S}(1 - p)$ which is maximized when $p = q_S/(q_S + 1)$, giving a bound of $\frac{\rho(k)}{e(q_S+1)}$. □

Theorem 5.8. *If there exists a PPT algorithm which can break the RSC receiver anonymity of the proposed ring signcryption scheme with advantage $\rho(k)$, then there exists a PPT algorithm which can solve the Computational Diffie–Hellman Problem with probability at least $2(1 - p2^{-poly(k)})\rho(k)$ where poly$(\cdot)$ is some polynomial and p is the maximum number of de-signcryption queries made in the model of RSC receiver anonymity.*

Proof. The proof follows that of Theorem 5.6. $\mathcal{B}$ is given P, aP, bP and aims to compute abP.

$\mathcal{B}$ randomly selects $x, y \overset{R}{\leftarrow} \mathbb{Z}_q$, and sets the two challenging public keys as $pk_{R,0} = xbP$, $pk_{R,1} = ybP$. Hash queries, ring signcrypt queries and de-signcrypt queries are simulated as in the proof of Theorem 5.6.

After completing the first stage, $\mathcal{A}$ gives $m, (pk_S^1, \ldots, pk_S^{i-1}, pk_S^{i+1}, \ldots, pk_S^j)$, $pk_{\hat{S}}, sk_{\hat{S}}, i$ to $\mathcal{B}$. $\mathcal{B}$ sets the challenging ciphertext to $pk_S = (pk_S^1, \ldots, pk_S^j), U^*, Z^*$, where $U^* = aP$ and Z^* is selected at random from $\{0,1\}^n \times \mathbb{G}_1^j$. $\mathcal{B}$ also randomly picks $b' \overset{R}{\leftarrow} \{0,1\}$, sets $H_1(m, aP, pk_{R,b'}, \top) = taP$ for random $t \overset{R}{\leftarrow} \mathbb{Z}_q$. Then $\mathcal{B}$ generates V^* according to the signcrypt algorithm and sets $H_2(aP, pk_{R,b'}, \top) = Z^* \oplus m \| V^*$. The challenging ciphertext is $\sigma^* = (pk_S, U^*, Z^*)$.

$\mathcal{B}$ answers $\mathcal{A}$'s queries as in the first stage. If $\mathcal{A}$ queries H_1 or H_2 on $(aP, pk_{R,0}, \lambda)$ such that $e(P, \lambda) = e(aP, pk_{R,0})$, $\mathcal{B}$ outputs $x^{-1}\lambda$ and halts. If $\mathcal{A}$ queries H_1 or H_2 on $(aP, pk_{R,1}, \lambda)$ such that $e(P, \lambda) = e(aP, pk_{R,1})$, $\mathcal{B}$ outputs $y^{-1}\lambda$ and halts. If $\mathcal{A}$ halts without making such queries, $\mathcal{B}$ outputs a random point in $\mathbb{G}_1$ and halts.

By following a similar security analysis as in the proof of Theorem 4.3, $\mathcal{B}$ breaks the CDH assumption with probability at least $2(1 - p2^{-poly(k)})\rho(k)$. □

6. Conclusion

We propose a solution to solve the security weaknesses of the YWD signcryption scheme. Our scheme not only achieves confidentiality, unforgeability and ciphertext anonymity under the Computational Diffie–Hellman assumption, but also is very efficient. We also define the notion of ring signcryption, and give an efficient construction based on our basic signcryption scheme.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments.

References

[1] J.H. An, Y. Dodis and T. Rabin, On the security of joint signature and encryption, in: *Proc. EUROCRYPT 2002*, LNCS, Vol. 2332, Springer, 2002, pp. 83–107.

[2] F. Bao and R.H. Deng, A signcryption scheme with signature directly verifiable by public key, in: *PKC'98*, LNCS, Vol. 1431, Springer, 1998, pp. 55–59.

[3] J. Beak, R. Steinfeld and Y. Zheng, Formal proofs for the security of signcryption, in: *PKC'02*, LNCS, Vol. 2274, Springer, 2002, pp. 80–98.

[4] M. Bellare, A. Boldyreva, A. Desai and D. Pointcheval, Key-privacy in public-key encryption, in: *Proc. ASIACRYPT 2001*, LNCS, Vol. 2248, Springer, 2001, pp. 566–582.

[5] M. Bellare and P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, in: *First ACM Conference on Computer and Communications Security*, ACM, Fairfax, 1993, pp. 62–73.

[6] D. Boneh and M. Franklin, Identity based encryption from the Weil pairing, in: *Proc. CRYPTO 2001*, LNCS, Vol. 2139, Springer, 2001, pp. 213–229.

[7] D. Boneh, C. Gentry, B. Lynn and H. Shacham, Aggregate and verifiably encrypted signatures from bilinear maps, in: *EUROCRYPT 2003*, Springer, 2003, pp. 416–432.

[8] D. Boneh, B. Lynn and H. Shacham, Short signatures from the Weil pairing, in: *Proc. ASIACRYPT 2001*, LNCS, Vol. 2248, Springer, 2001, pp. 514–532.
[9] X. Boyen, Multipurpose identity-based signcryption: A swiss army knife for identity-based cryptography, in: *Proc. CRYPTO 2003*, LNCS, Vol. 2729, Springer, 2003, pp. 383–399.
[10] S. Chow, S. Yiu, L. Hui and K. Chow, Efficient forward and provably secure ID-based signcryption scheme with public verifiability and public ciphertext authenticity, in: *Information Security and Cryptology – ICISC 2003*, LNCS, Vol. 2971, Springer, 2003, pp. 352–369.
[11] C. Gamage, J. Leiwo and Y. Zheng, Encrypted message authentication by firewalls, in: *PKC'99*, LNCS, Vol. 1560, Springer, 1999, pp. 69–81.
[12] S. Goldwasser, S. Micali and R. Rivest, A digital signature scheme secure against adaptive chosen-message attack, *SIAM J. Computing* **17**(2) (1988), 281–308.
[13] B. Libert and J.-J. Quisquater, New identity based signcryption schemes from pairings, in: *IEEE Information Theory Workshop*, IEEE, 2003, pp. 155–158; available at: http://eprint.iacr.org.
[14] B. Libert and J.-J. Quisquater, Efficient signcryption with key privacy from gap Diffie–Hellman groups, in: *PKC'04*, LNCS, Vol. 2947, Springer, 2004, pp. 187–200.
[15] J. Malone-Lee and W. Mao, Two birds one stone: Signcryption using RSA, in: *Topics in Cryptology – Proceedings of CT-RSA 2003*, LNCS, Vol. 2612, Springer, 2003, pp. 211–225.
[16] Y. Mu and V. Varadharajan, Distributed signcryption, in: *INDOCRYPT 2000*, LNCS, Vol. 1977, Springer, 2000, pp. 155–164.
[17] C. Rackoff and D.R. Simon, Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack, in: *Proc. CRYPTO 91*, LNCS, Vol. 576, Springer, 1992, pp. 433–444.
[18] R. Rivest, A. Shamir and Y. Tauman, How to leak a secret, in: *Proc. ASIACRYPT 2001*, LNCS, Vol. 2248, Springer, 2001, pp. 552–565.
[19] A. Shamir, Identity-based cryptosystems and signature schemes, in: *Proc. CRYPTO 84*, Springer, 1984, pp. 47–53.
[20] R. Steinfeld and Y. Zheng, A signcryption scheme based on integer factorization, in: *ISW'00*, LNCS, Vol. 1975, Springer, 2000, pp. 308–322.
[21] C.-H. Tan, On the security of signcryption scheme with key privacy, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **88**(4) (2005), 1093–1095.
[22] C.-H. Tan, Analysis of improved signcryption scheme with key privacy, *Information Processing Letters* **99**(4) (2006), 135–138.
[23] G. Yang, D.S. Wong and X. Deng, Analysis and improvement of a signcryption scheme with key privacy, in: *8th Information Security Conference (ISC'05)*, LNCS, Vol. 3650, 2005, pp. 218–232.
[24] D.H. Yum and P.J. Lee, New signcryption schemes based on KCDSA, in: *Information Security and Cryptology – ICISC 2001*, LNCS, Vol. 2288, Springer, 2002, pp. 305–317.
[25] Y. Zheng, Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption), in: *Proc. CRYPTO 97*, LNCS, Vol. 1294, Springer, 1997, pp. 165–179.