ID based generalized signcryption

Sunder Lal and Prashant Kushwah
Department of Mathematics
Dr B. R. A. (Agra) University
Agra- 282002 (UP) - INDIA
E-mail: sunder_lal2@rediffmail.com, pra.ibs@gmail.com

Abstract: Generalized signcryption is a new cryptographic primitive in which a signcryption scheme can work as an encryption scheme as well as a signature scheme. This paper presents an identity based generalized signcryption scheme based on bilinear pairing and discusses its security for message confidentiality non repudiation and ciphertext authentication.

Keywords: bilinear pairings, identity based cryptography, signcryption, generalized signcryption.

1. Introduction:

Confidentiality and authenticity are two main primitives of cryptography and realized through encryption schemes and digital signature schemes respectively. Logically two primitives are independent. In public key setting, the encryption uses the public key of the receiver whereas signature uses secrete or private key of the sender. To achieve both confidentiality and authenticity we use sign-then-encrypt approach which involves both encryption as well as signature. In 1997, Zheng [13] gave the concept of signcryption, which performs encryption and signature both in a single logical step. Computationally, signcryption is more efficient than 'sign-then-encrypt' approach. The use of signcryption reduces the number of steps, reduces the length of ciphertext and most importantly, it reduces the implementation complexity by combining the two modules of encryption and signature into a single module of signcryption. Zheng's original signcryption scheme is discrete logarithm based. In 1998, Zheng and Imai [14] gave a signcryption scheme based on elliptic curves. The first identity based signcryption scheme was proposed by Malone-Lee [8] in 2002. Since then, several identity based signcryption algorithms have been proposed [2, 4, 5, 7, 8, 9]. However, not all these schemes are supported by formal models and security proofs in the random oracle model. Boyen [2] gave the security notions for signcryption as: message confidentiality, signature non-repudiation, ciphertext unlinkability, ciphertext authentication, and ciphertext anonymity. Among the schemes supported by security proofs in formal security models, Chen and Malone Lee's proposal [4] happens to be most efficient construction; however, it loses ciphertext unlinkability.

All the above signcryption schemes work well when user wants both confidentiality and authenticity. However, not all messages require both confidentiality and authenticity. If only one of the two functionalities is required then the signcryption scheme is not efficient. In this scenario, according to Zheng, signcryption may be replaced with signature/encryption algorithm. Thus, to resolve the problem, we have to use three cryptographic algorithms signcryption, encryption and signature as per need. However it may not be feasible in some applications such as embedded systems and ubiquitous computing. In 2006, Han and Yang [6] proposed the idea to use the same scheme as a signcryption scheme, as an encryption scheme and as a signature scheme as per requirement. They termed the new primitive as **generalized signcryption.** There scheme is based on elliptic curves. Wang et al [12] improved upon the scheme [6] and provided security notions of generalized signcryption

scheme. It is to be noted that none of these schemes is identity based. Here we propose an identity based generalized signcryption (IDGSC) scheme.

In IDGSC, we have three modes signcryption mode, signature-only mode and encryption-only mode. The crucial point here is to identify the three modules. In the identity based cryptography, to sign a message we require the information about specific sender, to encrypt a message we require the information about specific recipient and to signcrypt a message we require the information about both sender and receiver. Thus identity can be used to distinguish the three cases. It is signcryption mode when both specific parties exist. It is signature/encryption mode when one of specific parties exists. We also give the security notions for IDGSC for message confidentiality, signature non-repudiation and ciphertext authentication. The security of our IDGSC scheme relies on the hardness of Bilinear Diffie-Hellman Problem (BDHP).

This paper will organize as follows: before introducing our ID based generalized signcryption scheme, we discuss a generalized signcryption scheme [6] in section 2. In section 3 definition of IDGSC and security notions for IDGSC are given and section 4 give the definition of bilinear pairings and some computational hard problem on which security of our scheme rely. Also we proposed IDGSC scheme in section 4. In section 5 security results are given. Before giving concluding remarks section 6 deals with the efficiency of proposed scheme.

2. A GSC based on ECDSA

First we describe the generalized signcryption scheme introduced by Han and Yang [6] in 2006.

Set up: Let $E(F_q)$ be an elliptic curve defined over F_q, and let P be a point on $E(F_q)$ of prime order p. Then $pP = \mathcal{O}.$ We let $\mathcal{O} = (\mathcal{O}_x, \mathcal{O}_y)$, where $\mathcal{O}$ is the point at infinity on elliptic curve.

Bind is an arbitrary string involving some information about the sender and the receiver. $\{0,1\}^*$ denotes the set of all binary strings and $\{0,1\}^n$ denotes the set of all n-bits binary strings. *Kenc*, *Ksig* denotes suitable binary strings.

The scheme uses four hash functions $H : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_1 : \mathbb{Z}_q^* \rightarrow \{0,1\}^{k_0+k_1}$, $H_2 : \mathbb{Z}_q^* \rightarrow \{0,1\}^{n+\ell}$, and $H_K : \{0,1\}^{n+k_2} \rightarrow \{0,1\}^\ell$. Here k_0 denotes the number of bits required to represents the binary length of the key K of keyed hash function H_K, k_1 is the number of bits in the binary string *Ksig*, k_2 is the number of bits required to represents the elements of $\mathbb{Z}_q$, ℓ is the number of bits required to represent the output of H_K and n is a number of bits of a message unit. For $x, 0 \in \mathbb{Z}_p$ $x\mathcal{O} = \mathcal{O}$ and $0P = \mathcal{O}$. Further $H_1(\mathcal{O}_y) = \mathbf{0}$, $H_2(\mathcal{O}_x) = \mathbf{0}$, $H_{\mathbf{0}}(.) = \mathbf{0}$, where $\mathbf{0}$ denotes suitable binary string of zeros.

Extract: Each user U randomly chooses his private key $d_U \in_R \mathbb{Z}_p^*$, computes the public key $Q_U = d_U P$.

In the signature-only mode (encryption-only mode) receiver (respectively sender) does not exist. We denote this situation by taking $U = \phi$ and $d_\phi = 0$ where 0 is the zero of $\mathbb{Z}_p$. In this situation $Q_\phi = 0P = \mathcal{O}$, where $\mathcal{O}$ is the point at infinity on the elliptic curve.

Let the message $m \in \{0,1\}^n$ be signcrypted by A (Alice, the sender) and unsigncrypted by B (Bob, the receiver).

Signcryption

A uses the algorithm GSC (generalized signcryption) with inputs m, d_A and Q_B to return $\sigma = (c, R, s)$ as follows:

1. Chooses $x \in_R \mathbb{Z}_p^*$,
2. Computes
  (i) $R = xP = (x_1, y_1)$, $r = x_1 \bmod q$
  (ii) $xQ_B = (x_2, y_2)$
  (iii) $H_2(x_2) = Kenc$, $H_1(y_2) = K \| Ksig$
3. If $d_A = 0$, takes $s = 0$; else computes
  (i) $s = x^{-1}(H(m \| Bind \| Ksig) + rd_A) \bmod n$
  (ii) $e = H_K(m \| s)$
4. Computes
    $c = (m \| e) \oplus Kenc$
5. Returns $\sigma = (c, R, s)$.

Unsigncryption

B uses the algorithm UGSC (generalized unsigncryption) with inputs σ, Q_A, and d_B to return m as follows:

1. Recovers r from R
2. Computes
  (i) $d_B R = (x_2, y_2)$
  (ii) $H_2(x_2) = Kenc$, $H_1(y_2) = K \| Ksig$
3. Recovers $(m \| e) = c \oplus Kenc$
4. Computes $e' = H_K(m \| s)$
  If $e \neq e'$, returns $\perp$; else,
  If $s = 0$ returns m; else,
5. Computes
  (i) $u_1 = s^{-1}(H(m \| Bind \| Ksig))$
  (ii) $u_2 = s^{-1}r$
  (iii) $R' = u_1 P + u_2 Q_A$
  If $R' \neq R$, return $\perp$;
  else returns m.

Signature-only mode

Sign

If A wants to only sign m, then A uses the algorithm GSC with inputs m, d_A and $Q_\phi = \mathcal{O}$ to return $\sigma = (m, R, s)$ as follows:

1. Chooses $x \in_R \mathbb{Z}_p^*$,
2. Computes
  (i) $R = xP = (x_1, y_1)$, $r = x_1 \bmod q$
  (ii) $x\mathcal{O} = \mathcal{O} = (\mathcal{O}_x, \mathcal{O}_y)$
  (iii) $H_2(\mathcal{O}_x) = \mathbf{0}$, $H_1(\mathcal{O}_y) = \mathbf{0}$
3. Computes
  (i) $s = x^{-1}(H(m \| \mathbf{0} \| \mathbf{0}) + rd_A) \bmod n$
  (ii) $\mathbf{0} = H_{\mathbf{0}}(m \| s)$
4. Computes
  (i) $m = (m \| \mathbf{0}) \oplus \mathbf{0}$
5. Returns $\sigma = (m, R, s)$.

Verify

Any recipient can use the algorithm UGSC with inputs m, Q_A, and $d_\phi = 0$ to verify the signature of A as follows:

1. Recovers r from R
2. Computes
  (i) $0R = \mathcal{O} = (\mathcal{O}_x, \mathcal{O}_y)$
  (ii) $H_2(\mathcal{O}_x) = \mathbf{0}$, $H_1(\mathcal{O}_y) = \mathbf{0}$
3. $m = m \oplus \mathbf{0}$
4. $\mathbf{0} = H_{\mathbf{0}}(m \| s)$
5. Computes
  (i) $u_1 = s^{-1}(H(m \| \mathbf{0} \| \mathbf{0}))$
  (ii) $u_2 = s^{-1}r$
  (iii) $R' = u_1 P + u_2 Q_A$
  If $R' \neq R$, return $\perp$;
  else returns T.

Encryption-only mode

Encrypt

Any one can encrypt m for Bob by using algortm GSC with inputs m, $d_\phi = 0$ and Q_B to return $\sigma = (c, R)$ as follows:

1. Chooses $x \in_R \mathbb{Z}_p^*$,

2. Computes

  (i) $R = xP$

  (ii) $xQ_B = (x_2, y_2)$

  (iii) $H_2(x_2) = Kenc$, $H_1(y_2) = K \| Ksig$

3. $s = 0$

4. Computes

  (i) $e = H_K(m \| \mathbf{0})$

  (ii) $c = (m \| e) \oplus Kenc$

5. Returns $\sigma = (c, R)$.

Decrypt

B uses the algorithm UGSC with inputs σ, $Q_\phi = \mathcal{O}$, and d_B to return m as follows:

1. Computes

  (i) $d_B R = (x_2, y_2)$

  (ii) $H_2(x_2) = Kenc$, $H_1(y_2) = K \| Ksig$

2. Recovers $(m \| e) = c \oplus Kenc$

3. Computes $e' = H_K(m \| \mathbf{0})$

  If $e \neq e'$, returns $\perp$;

  else returns m.

3. IDGSC and its Security

An Identity based generalized signcryption consists of the following algorithms:

Set Up: On input of a security parameter 1^k the private key generator (PKG) uses this algorithm to produce a pair (**param**, s), where params are global public parameters for the system and s is the master secrete key. The public parameters include P_{Pub}, the public key of PKG, a description of finite message space $\mathcal{M}$, a description of a finite signature space $\mathcal{S}$ and a description of a finite ciphertext space $\mathcal{C}$. Further, there is no need for publicly known **param** to be explicitly provided as input to any other algorithm.

Extract: On input of an identity ID_U and the master key s, PKG uses this algorithm to compute secrete key S_U corresponding to ID_U.

GSC: Suppose Alice (ID_A) wants to send a message m to Bob (ID_B). On input (S_A, ID_B, m), Alice uses this algorithm to produce cipher text c.

UGSC: On receiving c, Bob uses this algorithm with input (ID_A, S_B, c) and obtain m if c is valid ciphertext, and the symbol $\perp$ if c is invalid ciphertext.

The two algorithms GSC and UGSC are such that
    $c = (S_A, ID_B, m)$ iff m= UGSC (ID_A, S_B, c).

Signature-Only mode: If Alice wants only to sign a message m, then the specific receiver Bob does not exist. In this case $ID_B = ID_\phi$, GSC (S_A, ID_ϕ, m) = Sign (S_A, m), and UGSC (ID_A, S_ϕ, c) = Verify (ID_A, m).

Encryption-Only mode: If a user wants to encrypt a message for Bob, then the specific sender Alice does not exist. In this case GSC (S_ϕ, ID_B, m) = Enc (ID_B, m), and UGSC (ID_ϕ, S_B, c) = Dec (S_B, c).

We now discuss the security model for identity based generalized signcryption scheme.

3.1 Message Confidentiality

The accepted notion of security with respect to confidentiality for public key encryption is indistinguishability of encryptions under adaptive chosen ciphertext attack. The notion of security defined in the game below is a natural adaptation of this notion for the generalized signcryption scheme.

Game

Initial: The challenger runs **Setup** (1^k) and gives the resulting **params** to the adversary. It keeps s secrete.

Phase1: The challenger is probed by the adversary who makes the following queries.

- **Sign:** The adversary submits a signer identity and a message to the challenger. The challenger responds with the signature of the signer on the message.
- **Signcrypt:** The adversary submits a sender and receiver identity and a message to the challenger. The challenger responds with the signature of the sender on the message, encrypted under the public key of the receiver.
- **Decrypt:** The adversary submits a ciphertext and a receiver's identity to the challenger. The challenger decrypts the ciphertext under the secrete key of receiver and returns the message.
- **Unsigncrypt:** The adversary submits a ciphertext and a receiver's identity to the challenger. The challenger decrypts the ciphertext under the secrete key of receiver. It then verifies that the resulting decryption is a valid message/signature pair under the public key of the decrypted identity. If so the challenger returns the message, its signature and the identity of the signer, otherwise it returns $\bot$.
- **Extract:** The adversary submits an identity to the challenger. The challenger responds with the secrete key of that identity.

At the end of phase1 the adversary outputs two identity { ID_A , ID_B } and two messages { m_0, m_1 }. The adversary must not have made extraction query on ID_B.

Challenge: The challenger chooses a bit b uniformly at random. It signs m_b under secrete key corresponding to ID_A and encrypts the result under the public key of ID_B to produce c. The challenger returns c to the adversary.

Phase2: The adversary continues to probe the challenger with the same type of queries that it made in the phase1. It is not allowed to extract the private key corresponding to ID_B and it is not allowed to make a decrypt and unsigncrypt query for c under ID_B.

Response: The adversary returns a bit b'. The adversary wins if $b' = b$.

Definition1: Let $\mathcal{A}$ denote an adversary that plays the game above. If the quantity **Adv**[$\mathcal{A}$]$= 2\Pr[b' = b] - 1$ is negligible we say that the scheme is *semantically secure against adaptive chosen ciphertext attack,* or IND-IDGSC-CCA2 secure.

5

Note that above definition deals with insider security since the adversary is assumed to have access to the private key of the sender of a signcrypted message. This means that confidentiality is preserved even if a sender's key is compromised.

3.2 Signature Non-repudiation

Regarding the property of authentication and non-repudiation, the following definitions formalize the inability of any adversary to create a ciphertext containing a message authenticated by some user without knowing the latter's private key. We define the notion of non-repudiation via the following game

Game

Initial: The challenger runs **Setup** (1^k) and gives the resulting **params** to the adversary. It keeps s secrete.

Probing: The challenger is probed by the adversary who makes queries as in the phase1 of the game in section 3.1.

Forge: The adversary returns a recipient identity ID_B with its PKG and a ciphertext c. Let (m, ID_A, σ) be the result of decrypting c under the secrete key corresponding to ID_B. The adversary wins if $ID_A \neq ID_B$; $ID_A \neq ID_\phi$; **Verify** $(m, ID_A, \sigma) = \top$; no extraction query was made on ID_A; no sign query was responded with (m, ID_A, σ) and no signcrypt query $(m, ID_A, ID_{B'})$ was responded to with a ciphertext whose decryption under the private key of $ID_{B'}$ is (m, ID_A, σ).

Definition2: Let $\mathcal{A}$ denote an adversary that plays the game above. If the quantity **Adv**$[\mathcal{A}]$=Pr$[\mathcal{A}$ wins] is negligible we say that the scheme is *existentially unforgeable against insider chosen message attack*, or EUF-IDGSC-CMA secure.

Definition2 allows the adversary access to the secret key of the recipient of the forgery. It is this that gives us insider security.

3.3 Ciphertext Authentication:

Ciphertext authentication provides the guarantee to the recipient that the message was encrypted by the same person who signed it. We define this notion via a game played by a challenger and an adversary.

Game

Initial: The challenger runs **Setup** (1^k) and gives the resulting **params** to the adversary. It keeps s secrete.

Probing: The challenger is probed by the adversary who makes queries as in the phase1 of the game in section 3.1.

Forge: The adversary returns a recipient identity ID_B with its PKG and a ciphertext c. Let (m, ID_A, σ) be the result of decrypting c under the secrete key corresponding to ID_B. The adversary wins if $ID_A \neq ID_B$; $ID_A \neq ID_\phi$; **Verify** $(m, ID_A, \sigma) = \top$; no extraction query was made on ID_A or ID_B; and c did not result from a Signcrypt query with sender ID_A and recipient ID_B.

Definition3: Let $\mathcal{A}$ denote an adversary that plays the game above. If the quantity **Adv** $[\mathcal{A}]$ = Pr $[\mathcal{A}$ wins] is negligible we say that the scheme is *existentially ciphertext-unforgeable against outsider chosen message attacks*, or AUTH-IDGSC-CMA secure.

Here we have an example of outsider security since the adversary is not able to extract the secret key corresponding to ID_B.

4. Proposed IDGSC scheme

Before we present our identity based generalized signcryption scheme, we discuss some preliminaries.

Bilinear Pairings: Let G_1 be an additive group of order q, a prime and G_2 be a multiplicative group of same order q. A function e: $G_1 \times G_1 \to G_2$ is called a **bilinear pairing** if it satisfies the following properties:

(i) $\forall P, Q \in G_1, \forall a, b \in \mathbb{Z}_q^*, \ e(aP, bQ) = e(P,Q)^{ab}$

(ii) For any point $P \in G_1, e(P,Q) = 1$ for all $Q \in G_1$ iff $P = \mathcal{O}$, the identity of G_1.

(iii) There exists an efficient algorithm to compute $e(P,Q), \forall P, Q \in G_1$.

Computational Diffie-Hellman Problem (CDHP): Given P, aP, bP in G_1, for some (unknown) $a, b \in \mathbb{Z}_q^*$, compute abP in G_1.

Bilinear Diffie-Hellman Problem (BDHP): Given P, aP, bP, cP in G_1, for some (unknown) $a, b, c \in \mathbb{Z}_q^*$, compute $e(P,P)^{abc}$ in G_2.

Our proposed IDGSC scheme works as follows:

Setup:

Establishes parameters $G_1, G_2, \ q, \ e: G_1 \times G_1 \to G_2$, $H_0 : \{0,1\}^{k_1} \to G_1$, $H_1 : \{0,1\}^{k_0+n} \to \mathbb{Z}_q^*$, $H_2 : G_2 \to \{0,1\}^{k_0+k_1+n}$, where k_0 is the number of bits required to represent an element of G_1, k_1 is the number of bits required to represent an identity of a user and n is a number of bits of a message unit.

Chooses P, a generator of cyclic group G_1.

Chooses a random s in $\mathbb{Z}_q^*$ and computes the public key of PKG, $P_{Pub} = sP$.

The system parameter **params** are $\langle G_1, G_2, q, e, P, P_{pub}, n, H_0, H_1, H_2 \rangle$. Further we take output of $H_2(1)$ as $(k_0 + k_1 + n)$ bit zero string.

Extract: Extracts private key of the user U with $ID_U \in \{0,1\}^{k_1}$

Computes the public key $Q_U = H_0(ID_U)$ and the private key $S_U = sQ_U$.

For signature-only mode (encryption-only mode) where receiver (sender) does not exist, we use the key pair $(\mathcal{O}, \mathcal{O}) \leftarrow (Q_U, S_U)$ when $U = ID_U = ID_\phi$ where ID_ϕ is a k_1 bits zero string.

GSC (S_A, ID_B, m) : To send a message $m \in \{0,1\}^n$ to Bob (ID_B) in a secure and authenticated way, Alice (ID_A) does the following:

1. Chooses $r \in_R \mathbb{Z}_q^*$

2. Computes

      (i) $X = rP + rQ_A$, where $Q_A = H_0(ID_A)$

      (ii) $Z = rP_{Pub} + (r + h_1)S_A$, where $h_1 = H_1(X \| m)$

      (iii)$\omega = e(rP_{Pub} + rS_A, Q_B)$, where $Q_B = H_0(ID_B)$

      (iv)$y = (Z \| ID_A \| m) \oplus H_2(\omega)$

3. Return $c = (X, y)$.

      Here c is the signcryptext of message m.

UGSC(ID_A, S_B, c): On receiving the signcryptext $c = (X, y)$, Bob

1. Computes

      (i) $Q_A = H_0(ID_A)$

      (ii) $\omega = e(X, S_B)$

      (iii)$Z \| ID_A \| m = y \oplus H_2(\omega)$

      (iv) $h_1 = H_1(X \| m)$

      (v) $e(Z, P)$

      (vi) $e(P_{pub}, X + h_1Q_A)$

2. Returns valid iff $e(Z, P) = e(P_{pub}, X + h_1Q_A)$.

Consistency:

$$e(X, S_B) = e(rP + rQ_A, sQ_B)$$
$$= e(P + Q_A, sQ_B)^r$$
$$= e(sP + sQ_A, Q_B)^r$$
$$= e(P_{Pub} + S_A, Q_B)^r$$

and

$$e(P_{pub}, X + h_1Q_A) = e(sP, rP + rQ_A + h_1Q_A)$$
$$= e(P, rsP + (r + h_1)sQ_A)$$
$$= e(P, rP_{Pub} + (r + h_1)S_A)$$
$$= e(P, Z) = e(Z, P)$$

      Once Bob has recovered m and Z, he can prove to a third party that (Z, X) is a valid signature of Alice on m. Third party can compute $h_1 = H_1(X \| m)$ and then can verify that $e(Z, P) = e(P_{pub}, X + h_1Q_A)$.

Signature-only mode: GSC $(S_A, ID_\phi, m) =$ Sign (S_A, m)

      If Alice only wants to sign $m \in \{0,1\}^n$, then she

1. Chooses $r \in_R \mathbb{Z}_q^*$

2. Computes

      (i) $X = rP + rQ_A$, where $Q_A = H_0(ID_A)$

(ii) $Z = rP_{Pub} + (r + h_1)S_A$, where $h_1 = H_1(X \| m)$

(iii) $1 = e(P_{Pub} + S_A, \mathcal{O})^r$

(iv) $Z \| ID_A \| m = (Z \| ID_A \| m) \oplus H_2(1)$, and

3. Return $\sigma = (Z \| ID_A \| m, X)$.

Here σ is the signature on message m.

UGSC (ID_A, ID_ϕ, σ) = Verify (ID_A, σ)

Any one can verify the signature on m by
computing

(i) $Q_A = H_0(ID_A)$

(ii) $1 = e(X, \mathcal{O})$

(iii) $Z \| ID_A \| m = Z \| ID_A \| m \oplus H_2(1)$

(iv) $h_1 = H_1(X \| m)$

(v) $e(Z, P)$

(vi) $e(P_{pub}, X + h_1 Q_A)$

and concluding that σ is valid iff $e(Z, P) = e(P_{pub}, X + h_1 Q_A)$.

Encryption-only mode: GSC (ID_ϕ, ID_B, m) = Enc (ID_B, m)

If user wants to send a message $m \in \{0,1\}^n$ in a secure manner to Bob then he/she

1. Chooses $r \in_R \mathbb{Z}_q^*$

2. Computes

(i) $X = rP = rP + \mathcal{O}$

(ii) $Z = rP_{Pub} = rP_{Pub} + (r + h_1)\mathcal{O}$, where $h_1 = H_1(X \| m)$

(iii) $\omega = e(rP_{Pub}, Q_B) = e(rP_{Pub} + \mathcal{O}, Q_B)$, where $Q_B = H_0(ID_B)$

(iv) $y = (Z \| ID_\phi \| m) \oplus H_2(\omega)$

3. Return $c = (X, y)$ as the ciphertext of message m.

UGSC (ID_ϕ, d_{ID_B}, c) = Dec (S_B, c)

On receiving ciphertext $c = (X, y)$, Bob

1. Computes

(i) $\omega = e(X, S_B)$

(ii) $Z \| ID_\phi \| m = y \oplus H_2(\omega)$

(iii) $h_1 = H_1(X \| m)$

(iv) $e(Z, P)$

(v) $e(P_{pub}, X + h_1 \mathcal{O})$

2. Accepts m as plaintext iff $e(Z, P) = e(P_{pub}, X + h_1 \mathcal{O})$.

5. Security:

In this section we state the security results for the IDGSC scheme under the definition
of section 4. The proofs are suitable modification in the proofs in [4].

All our security results are based on the Bilinear Diffie-Hellman Problem (BDHP) defined in section 5. Our results assume that the hash functions H_0, H_1 and H_2 in the IDGSC scheme are all random oracles. In each of the results below we assume that the adversary makes q_i queries to H_i for i= 0, 1, 2. The number of sign, signcrypt, decrypt and unsigncrypt queries made by the adversary are denoted by q_s, q_{sc}, q_d and q_u respectively.

5.1 Theorem (Message Confidentiality)

If there is an IND-IDGSC-CCA2 adversary $\mathcal{A}$ of IDGSC that succeeds with probability ϵ, then there is a simulator $\mathcal{B}$ running in polynomial time that solves the BDHP with probability at least

$$\epsilon \cdot \left(1 - \frac{q_s(q_1 + q_s + q_{sc})}{q}\right)\left(1 - \frac{q_{sc}(q_1 + q_s + q_{sc})}{q}\right) \cdot \frac{1}{q_0 q_2}$$

5.2 Theorem (Signature Non-repudiation)

If there is an EUF-IDGSC-CMA adversary $\mathcal{A}$ of IDGSC that succeeds with probability ϵ, then there is a simulator $\mathcal{B}$ running in polynomial time that solves the BDHP with probability at least

$$\epsilon \cdot \left(1 - \frac{q_s(q_1 + q_s + q_{sc})}{q}\right)^2 \cdot \left(1 - \frac{q_{sc}(q_1 + q_s + q_{sc})}{q}\right)^2 \cdot \frac{1}{4q_0^2(q_1 + q_s + q_{sc})^2}$$

5.3 Theorem (Ciphertext Authentication)

If there is an AUTH-IDGSC-CMA adversary $\mathcal{A}$ of IDGSC that succeeds with probability ϵ, then there is a simulator $\mathcal{B}$ running in polynomial time that solves the BDHP with probability at least

$$\epsilon \cdot \left(1 - \frac{q_s(q_1 + q_s + q_{sc})}{q}\right)\left(1 - \frac{q_{sc}(q_1 + q_2 + q_s + 2q_{sc})}{q}\right) \cdot \frac{1}{q_0(q_0 - 1)(q_{sc} + q_u)(q_2 + q_{sc})}$$

5.4 Lemma 1: If there is a EUF-IDGSC-CMA adversary $\mathcal{A}$ of IDGSC in signcryption mode that succeeds with probability with ϵ, then there is a EUF-IDGSC-CMA adversary of IDGSC in signature-only mode that succeeds with probability at least ϵ.

5.5 Lemma 2: If there is an IND-IDGSC-CPA adversary $\mathcal{A}$ of IDGSC in signcryption mode that succeeds with probability with ϵ, then there is an IND-IDGSC-CPA adversary of IDGSC in encryption-only mode that succeeds with probability at least ϵ.

Remark:

Our scheme is a combination of Malone-Lee [8] and Chen and Malone-Lee [4] signcryption schemes. In the signature-only mode it reduces to a modified form of Cha-Cheon signature scheme. In the encryption-only mode it reduces to Boneh-Frankline [1] basic encryption scheme, which is chosen plaintext secure.

Boyen [2] gave three additional security notions ciphertext unlinkability, ciphertext authentication and ciphertext anonymity for identity based signcryption schemes. Like the scheme in [4], this scheme too does not possess the ciphertext unlinkability. Also we use identities of sender and receiver as the identifier of signature-only mode and encryption-only mode. Hence the concept of ciphertext anonymity does not exist.

6. Efficiency and Comparisons:

The idea of generalized signcryption is to reduce implementation complexity (same algorithm for encryption, signature and signcryption). It may not reduce the computational complexity. However, the communication complexity in encryption-only mode and signature-only mode may increase a bit. In our scheme, the signcryption mode is as efficient as [4]. This is an additional advantage of our scheme.

In Table 1 we assess the comparative efficiency of some identity based signcryption schemes. Table 1 summarizes the number of relevant basic operations underlying the some identity based signcryption, namely scalar multiplication (G_1mls.) exponentiation (G_2 exps.) and pairing evaluation (e cps.).

Signcryption schemes	Sign/Encrypt			Decrypt/Verify		
	G_1 mls	G_2 exps	e cps	G_1 mls	G_2 exps	e cps
Malone-Lee [8]	3	—	1	1	—	3
Nalla-Reddy [9]	2	1	1	—	1	3
Libert-Quisquater[7]	2	—	2	1	—	4
Chen-J.M. Lee [4]	3	—	1	1	—	3
X. Boyen [2]	3	1	1	2	—	4
Our scheme	3	—	1	1	—	3

Table 1

Conclusion: In this paper, we proposed an identity based generalized signcryption scheme. To achieve our goal of generalized signcryption we use a variant of Cha-Cheon [3] signature scheme. We also compare the efficiency of our scheme with several signcryption schemes.

Reference:

1. D. Boneh and M. Franklin: Identity–based encryption scheme from Weil pairing. CRYPTO 2001, LNCS # 2139, Springer-Verlag, 2001, 213-229.
2. X. Boyen: Multipurpose Identity based signcryption: A Swiss army knife for identity based cryptography. CRYPTO 2003, LNCS # 2729, Springer-Verlag, 2003, 389-399.
3. J.C. Cha and J.H. Cheon: An identity-based signature from Gap Diffie-Hellman Groups. PKC-2003, LNCS # 2567, Springer-Verlag, 2003, 18-30.
4. L. Chen and J. Malone-Lee: Improved Identity-based signcryption. PKC 2005, LNCS # 3386, Springer-Verlag, 2005, 362-379.
5. S. S. M. Chow, S. M. Yiu, L. C. K. Hui and K. P. Chow: Efficient forward and provably secure ID based signcryption scheme with public verifiability and public cipher text authenticity. ICISC'2003, LNCS # 2971, Springer-Verlag, 2003, 352-369.
6. Y. Han and X. Yang: ECGSC: Elliptic curve based generalized signcryption scheme. Cryptology ePrint Archive, Report 2006/126, 2006, http:/eprint.iacr.org/.
7. B. Libert and J.J. Quisquater: New Identity based signcryption schemes from   pairings. IEEE Information Theory Workshop, Paris (France) 2003.
8. J. Malone-Lee: Identity based signcryption. Cryptology ePrint Archive, Report 2002/098, 2002, http:/eprint.iacr.org/.
9. D. Nalla and K.C. Reddy: Signcryption scheme for identity based cryptosystems. Cryptology ePrint Archive, Report 2003/066, 2003, http:/eprint.iacr.org/.
10. D. Pointcheval and J. Stern: Security arguments for digital signature and blind signature. Journal of cryptology, 13(3): 361-396, 2000.

11. A. Shamir: Identity-based cryptosystems and signature schemes. CRYPTO 84, LNCS # 196, Springer-Verlag, 1984, 47-53.
12. X. Wang, Y. Yang and Y. Han: Provable secure generalized signcryption. Cryptology ePrint Archive, Report 2007/173, 2007, http:/eprint.iacr.org/.
13. Y. Zheng: Digital signcryption or How to Achieve cost (Signature & Encryption) << cost (Signature) + cost (Encryption). CRYPTO'97, LNCS # 1294, Springer-Verlag, 1997, 165-179.
14. Y. Zheng and H. Imai: How to construct efficient signcryption schemes on elliptic curves. Information Processing Letters 68(5), 1998.

Appendix:
Proof of Theorem 5.1:

We will show how an IND-IDGSC-CCA2 adversary $\mathcal{A}$ of IDGSC may be used to construct a simulator $\mathcal{B}$ that solves the BDHP. Let (P, aP, bP, cP) be the instant of the BDHP that we wish to solve.

The simulator runs $\mathcal{A}$ with $\text{Ppub} = bP$. It also creates algorithms to respond to queries made by $\mathcal{A}$ during its attack. To maintain consistency between queries made by $\mathcal{A}$, the simulator keeps the following lists: L_i for $i = 0, 1, 2$ of data for query/response pairs to random oracle H_i; L_{sc} of signcryptions generated by the simulator; L_s of signatures generated by the simulator; L_d of some of the queries made by $\mathcal{A}$ to decrypt oracle and L_u of some of the queries made by $\mathcal{A}$ to unsigncryption oracle. We describe how $\mathcal{B}$ runs phase1 of $\mathcal{A}$'s attack below.

Simulator: $H_0(\text{ID}_U)$

At the beginning of simulator choose i_β uniformly at random from $\{1, \ldots, q_0\}$. We show how to respond to the ith query made by $\mathcal{A}$ below. Note that we assume that $\mathcal{A}$ does not make repeat queries.
— If $i = i_\beta$ then respond with $H_0(\text{ID}_U) = aP$ and set $\text{ID}_\beta = \text{ID}_U$.
— Else chooses x uniformly at random from $\mathbb{Z}_q^*$, compute $Q_U = xP$; compute $S_U = xP_{Pub}$; store $(\text{ID}_U, Q_U, S_U, x)$ in L_0 and respond with Q_U.

Simulator: $H_1(X \| m)$
— If $(X \| m, h_1) \in L_1$ for some h_1, returns h_1.
— Else choose h_1 uniformly at random from $\mathbb{Z}_q^*$; add $(X \| m, h_1)$ to L_1 and returns h_1.

Simulator: $H_2(\omega)$
— If $(\omega, h_2) \in L_2$ for some h_2, return h_2.
— Else choose h_2 uniformly at random from $\{0,1\}^{k_0+k_1+n}$; add (ω, h_2) to L_2 and return h_2.

Simulator: Extract (ID_U)

We will assume that $\mathcal{A}$ makes the query $H_0(\text{ID}_U)$ before it makes the extraction query for ID_U.
— If $\text{ID}_U = \text{ID}_\beta$ abort the simulation.
— Else search L_0 for the entry $(\text{ID}_U, Q_U, S_U, x)$ corresponding to ID_U and return S_U.

Simulator: Sign $(m, \text{ID}_1, \text{ID}_\phi)$

We will assume that $\mathcal{A}$ makes the queries $H_0(\text{ID}_1)$ before it makes a sign query on message m using this identity. We have following cases to consider

13

Case1: $ID_1 \neq ID_\beta$

— Find the entry (ID_1, Q_1, S_1, x) in L_0

— Choose r uniformly at random from $\mathbb{Z}_q^*$ and compute $X = rP + rQ_1$

— Compute $h_1 = H_1(X \| m)$ (where H_1 is the simulator above)

— Compute $Z = rP_{Pub} + (r + h_1)S_1$

— Compute $1 = e(rP_{Pub} + rS_1, \mathcal{O})$

— Compute $Z \| ID_1 \| m = H_2(1) \oplus (Z \| ID_1 \| m)$ (where $H_2(1)$ is a zero string of length $k_0 + k_1 + n$)

— Return $(X, Z \| ID_1 \| m)$

Case2: $ID_1 = ID_\beta$

— Choose r, h_1 uniformly at random from $\mathbb{Z}_q^*$

— Compute $X = rP - h_1 Q_\beta$ and $Z = rP_{Pub}$

— Add $(X \| m, h_1)$ to L_1

— Compute $1 = e(X, \mathcal{O})$

— Compute $Z \| ID_\beta \| m = H_2(1) \oplus (Z \| ID_\beta \| m)$ (where $H_2(1)$ is a zero string of length $k_0 + k_1 + n$)

— Return $(X, Z \| ID_\beta \| m)$

Simulator: Signcrypt (m, ID_1, ID_2)

We will assume that $\mathcal{A}$ makes the queries $H_0(ID_1)$ and $H_0(ID_2)$ before it makes a signcryption query on message m using these identities. We have following cases to consider

Case1: $ID_1 \neq ID_\beta$

— Find the entry (ID_1, Q_1, S_1, x) in L_0

— Choose r uniformly at random from $\mathbb{Z}_q^*$ and compute $X = rP + rQ_1$

— Compute $h_1 = H_1(X \| m)$ (where H_1 is the simulator above)

— Compute $Z = rP_{Pub} + (r + h_1)S_1$

— Compute $Q_2 = H_0(ID_2)$ (where H_0 is the simulator above)

— Compute $\omega = e(rP_{Pub} + rS_1, Q_2)$

— Compute $y = H_2(\omega) \oplus (Z \| ID_1 \| m)$ (where H_2 is the simulator above)

— Return (X, y).

Case2: $ID_1 = ID_\beta$

— If $(m, ID_\beta, X, Z, r, h_1) \in L_s$ for m then

  • Find the entry (ID_2, Q_2, S_2, x) in L_0

  • Compute $\omega = e(X, S_2)$

  • Compute $y = H_2(\omega) \oplus (Z \| ID_\beta \| m)$ (where H_2 is the simulator above)

  • Return (X, y)

— Else choose r, h_1 uniformly at random from $\mathbb{Z}_q^*$

- Compute $X = rP - h_1 Q_\beta$ and $Z = rP_{Pub}$
- Add $(X \| m, h_1)$ to L_1
- Find the entry (ID_2, Q_2, S_2, x) in L_0
- Compute $\omega = e(X, S_2)$
- Compute $y = H_2(\omega) \oplus (Z \| ID_\beta \| m)$ (where H_2 is the simulator above)
- Return (X, y)

Simulator: Decrypt $(X, y) \, ID_2$

We will assume that $\mathcal{A}$ makes the queries $H_0(ID_2)$ before it makes a decryption query for ID_2. We have following cases to consider

Case1: $ID_2 \neq ID_\beta$

— Find the entry (ID_2, Q_2, S_2, x) in L_0

— Compute $\omega = e(X, S_2)$

— If $\omega \notin L_2$, return $\perp$. Else $Z \| ID_1 \| m = y \oplus H_2(\omega)$ (where H_2 is the simulator above)

— If $ID_1 \neq ID_\phi$, return $\perp$. Else $Q_1 = \mathcal{O}$

— If $e(Z, P) \neq e(P_{Pub}, X)$ return $\perp$. Else return m, (X, Z), ID_ϕ.

Case2: $ID_2 = ID_\beta$

Step through the list L_2 with entries (ω, h_2) as follows

— Compute $Z \| ID_1 \| m = y \oplus h_2$

— If $ID_1 \neq ID_\beta$, move to the next element in L_2 and begin again

— If $ID_1 = ID_\phi$ set $Q_1 = \mathcal{O}$

— Check that $\omega = e(Z, aP)$ and if not move to the next element in L_2 and begin again

— Check that $e(Z, P) = e(P_{Pub}, X)$, if so return m, else move to the next element in L_2

— If no message has been returned after stepping through L_2 return $\perp$.

Simulator: Unsigncrypt (X, y), ID_2

We will assume that $\mathcal{A}$ makes the queries $H_0(ID_2)$ before it makes an unsigncryption query for ID_2. We have following cases to consider

Case1: $ID_2 \neq ID_\beta$

— Find the entry (ID_2, Q_2, S_2, x) in L_0

— Compute $\omega = e(X, S_2)$

— If $\omega \notin L_2$, return $\perp$. Else $Z \| ID_1 \| m = y \oplus H_2(\omega)$ (where H_2 is the simulator above)

— If $ID_1 = ID_\phi$, $ID_1 = ID_2$ or $ID_1 \notin L_0$, return $\perp$. Else $Q_1 = H_0(ID_1)$

— If $X \| m \notin L_1$, return $\perp$. Else $h_1 = H_1(X \| m)$ (where H_1 is the simulator above)

— If $e(Z, P) \neq e(P_{Pub}, X + h_1 Q_1)$ return $\perp$. Else return m, (X, Z), ID_1.

Case2: $ID_2 = ID_\beta$

Step through the list L_2 with entries (ω, h_2) as follows

— Compute $Z \| ID_1 \| m = y \oplus h_2$

15

— If $ID_1 = ID_\phi$ or $ID_1 = ID_\beta$, move to the next element in L_2 and begin again

— If $ID_1 \in L_0$ let $Q_1 = H_0(ID_1)$ and find S_1 in L_0, else move to the next element in L_2

— Check that $\omega = e(Z - h_1 S_1, aP)$ and if not move to the next element in L_2 and begin again

— Check that $e(Z, P) = e(P_{Pub}, X + h_1 Q_1)$, if so return m, else move to the next element in L_2

— If no message has been returned after stepping through L_2 return $\perp$.

At the end of phase1 the adversary outputs two identities { ID_A, ID_B } and two messages { m_0, m_1 }. If $ID_B \neq ID_\beta$, $\mathcal{B}$ aborts the simulation. Otherwise it chooses $y^* \in \{0,1\}^{k_0 + k_1 + n}$ and sets $X^* = cP$. It returns the challenge ciphertext $\sigma^* = (X^*, y^*)$ to $\mathcal{A}$. The queries made by $\mathcal{A}$ in phase2 are responded to in the same way as those made by $\mathcal{A}$ in phase1.

At the end of phase2, $\mathcal{A}$ outputs a bit b. The simulator ignores this bit. It chooses some ω at random from L_2 as its guess at the solution of the $BDHP(P, aP, bP, cP)$. We call this event ch_1. Let us now consider how our simulation could fail when it execute phase1 of $\mathcal{A}$'s attack i.e. what events could cause $\mathcal{A}$'s view to differ when run by $\mathcal{B}$ from its view in a real attack. We call such an event error and denote it ER.

It is clear that the simulations for H_0, H_1 and H_2 are indistinguishable from genuine random oracles. Since they are only define at points where they are called by $\mathcal{A}$ or $\mathcal{B}$.

Let us now consider how the simulation for sign could fail. We denote such an event S-ER. The most likely failure will caused by the sign simulator responded to a query of the form case2 (see simulator). The only possibility for introducing an error here defining $H_1(X \| m)$ where it is already defined. Since X takes the values uniformly at random in $\langle P \rangle$, the chance of these event occurring is most $(q_1 + q_s + q_{sc})/q$ for each query. The q_s and q_{sc} comes from the fact that the signing and signcryption simulator adds elements to L_1. Therefore over the whole simulation, the chance of an error introduced in this way is at most

$$\frac{q_s(q_1 + q_s + q_{sc})}{q}.$$

With the similar argument the chance of an error introduced in signcrypt simulator is at most

$$\frac{q_{sc}(q_1 + q_s + q_{sc})}{q}$$

we denote an error in the signcryption simulator by SC-ER.

Also the decrypt and unsigncrypt simulators are indistinguishable from genuine random oracles. The final simulator to consider is the extract simulator. Looking at the H_0 simulator we see that it chooses one H_0 query made by the adversary and responds to this with group elements from the BDHP instance that it is trying to solve. The simulator hopes that this will be the identity chosen by $\mathcal{A}$ for the recipient in the challenge. This will be the case with probability at least

$$\frac{1}{q_0}$$

If this is not the case we say that an error has occurred in the extract simulator because, if the adversary tried to extract the private key for this identity the simulator would abort. An error in the extract simulator is denoted by E-ER.

Let us now consider what errors that could be when $\mathcal{B}$ execute phase2 of $\mathcal{A}$. All the same errors are possible, in addition the simulator will fail if the adversary makes the H_2 query with $\omega = (P, P)^{abc}$. However if $\mathcal{A}$ has any advantage it must make this query, and once it has done so we have trapped it into leaving enough information in L_2 to solve the BDHP.

Let Ask be the event that a H_2 query related to $\omega = (P, P)^{abc}$ is issued at some point.

$$\Pr[b' = b] = \Pr[b' = b \mid Ask]\Pr[Ask] + \Pr[b' = b \mid \neg Ask]\Pr[\neg Ask]$$
$$\leq \Pr[Ask] + \Pr[b' = b \mid \neg Ask](1 - \Pr[Ask])$$
$$= \Pr[b' = b \mid \neg Ask] + (1 - \Pr[b' = b \mid \neg Ask]) \cdot \Pr[Ask]$$

Clearly $\Pr[b' = b \mid \neg Ask] = \frac{1}{2}$. Hence

$$\frac{\epsilon + 1}{2} \leq \frac{1}{2} + \frac{1}{2}\Pr[Ask]$$
$$\Pr[Ask] \geq \epsilon$$

Thus we have

$$\mathrm{Adv}[\mathcal{B}] \geq \Pr[Ask \wedge \neg S - ER \wedge \neg SC - ER \wedge \neg E - ER \wedge ch_1]$$
$$\geq \epsilon \cdot \left(1 - \frac{q_s(q_1 + q_s + q_{sc})}{q}\right) \cdot \left(1 - \frac{q_{sc}(q_1 + q_s + q_{sc})}{q}\right) \cdot \frac{1}{q_0 q_2}$$

as required.

Proof of Theorem 5.2:

We are going to use the "forking lemma" technique of Pointcheval and Stern [10] to prove our result. We will in fact reduce the standard Diffie-Hellman problem (DHP) to the problem of forging. Since a black box for the DHP is sufficient to solve the BDHP the result will follow. We will now show how a EUF-IDGSC-CMA adversary $\mathcal{A}$ of IDGSC may be used to construct a simulator $\mathcal{B}$ that solves the DHP. Let (P, aP, bP) be the instant of the DHP that we wish to solve. The simulator runs in three stages: $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$.

The simulator $\mathcal{B}_1$ runs $\mathcal{A}$ with PKG public key $Ppub = bP$. It also creates algorithms to respond to queries made by $\mathcal{A}$ during its attack. To maintain consistency between queries made by $\mathcal{A}$, the simulator keeps lists as in the proof of Theorem 1.

Simulator: $H_0(ID_U)$

At the beginning of simulator choose i_a uniformly at random from $\{1, \ldots, q_0\}$. We show how to response to the ith query made by $\mathcal{A}$ below. Note that we assume that $\mathcal{A}$ does not make repeat queries.

— If $i = i_a$ then respond with $H_0(ID_U) = aP$ and set $ID_A = ID_U$.

— Else chooses x uniformly at random from $\mathbb{Z}_q^*$, compute $Q_U = xP$; compute $S_U = xP_{Pub}$; store (ID_U, Q_U, S_U, x) in L_0 and respond with Q_U.

Simulator: $H_1(X \| m)$ and $H_2(\omega)$ as in the proof of Theorem 1.

Simulator: Extract (ID_U)

We will assume that $\mathcal{A}$ makes the query $H_0(ID_U)$ before it makes the extraction query for ID_U.

— If $ID_U = ID_A$ abort the simulation.
— Else search L_0 for the entry (ID_U, Q_U, S_U, x) corresponding to ID_U and return S_U.

Simulator: Sign (m, ID_1, ID_ϕ)

We will assume that $\mathcal{A}$ makes the queries $H_0(ID_1)$ before it makes a sign query using this identity. We have following cases to consider

Case1: $ID_1 \neq ID_A$

Use the simulation from case1 of sign in the proof of Theorem 1.

Case2: $ID_1 = ID_A$

Use the simulation from case2 of sign in the proof of Theorem 1 by replacing ID_β by ID_A and Q_β by Q_A.

Simulator: Signcrypt (m, ID_1, ID_2)

We will assume that $\mathcal{A}$ makes the queries $H_0(ID_1)$ and $H_0(ID_2)$ before it makes a signcryption query using these identities. We have following cases to consider

Case1: $ID_1 \neq ID_A$

Use the simulation from case1 of signcrypt in the proof of Theorem 1.

Case2: $ID_1 = ID_A$

Use the simulation from case2 of signcrypt in the proof of Theorem 1 by replacing ID_β by ID_A and Q_β by Q_A.

Simulator: Decrypt $(X, y) ID_2$

We will assume that $\mathcal{A}$ makes the queries $H_0(ID_2)$ before it makes a decryption query for this identity. We have following cases to consider

Case1: $ID_2 \neq ID_A$

Use the simulator from case1 of decrypt in the proof of Theorem 1.

Case2: $ID_2 = ID_A$

Use the simulation from case2 of decrypt in the proof of Theorem 1 by replacing ID_β by ID_A and aP by Q_A.

Simulator: Unsigncrypt $(X, y) ID_2$

We will assume that $\mathcal{A}$ makes the queries $H_0(ID_2)$ before it makes an unsigncryption query for this identity. We have following cases to consider

Case1: $ID_2 \neq ID_A$

Use the simulator from case1 of unsigncrypt in the proof of Theorem 1.

Case2: $ID_2 = ID_A$

Use the simulation from case2 of unsigncrypt in the proof of Theorem 1 by replacing ID_β by ID_A and aP by Q_A.

Let us now consider how our simulation could fail i.e. what events could cause $\mathcal{A}$'s view to differ when run by $\mathcal{B}_1$ from its view in a real attack. We call such an event error and denote it ER. It is clear that the simulations for H_0, H_1 and H_2 are indistinguishable from genuine random oracles. Since they are only defines at points where they are called by $\mathcal{A}$ or $\mathcal{B}_1$.

Let us now consider how the simulation for sign and signcrypt could fail. The analysis of these case are identical to the equivalent cases in Theorem 1. An error is therefore introduced in sign and signcrypt simulator respectively with probability at most

$$\frac{q_s(q_1 + q_s + q_{sc})}{q} \text{ and } \frac{q_{sc}(q_1 + q_s + q_{sc})}{q} \tag{1}$$

Also the decrypt and unsigncrypt simulators are indistinguishable from genuine random oracles. The final simulator to consider is the extract simulator. Note that the adversary will only succeed in its task with non-negligible probability if it queries H_0 with identity under which the message contained in the ciphertext it returns is signed. Looking at the H_0 simulator we see that it chooses one H_0 query made by the adversary and responds to this with group elements from the DHP instance that it is trying to solve. The simulator hopes that this will be the signer identity for the ciphertext it returns. This will be the case with probability at least

$$\frac{1}{q_0} \tag{2}$$

If this is not the case we say that an error has occurred in the extract simulator because, if the adversary tried to extract the private key for this identity the simulator would abort.

Now from (1) and (2), it is clear that with probability grater or equal to

$$\epsilon \cdot \left(1 - \frac{q_s(q_1 + q_s + q_{sc})}{q}\right) \cdot \left(1 - \frac{q_{sc}(q_1 + q_s + q_{sc})}{q}\right) \cdot \frac{1}{q_0} \tag{3}$$

the simulator obtains from the adversary a recipient identity ID_B and a ciphertext c such that, if (m, ID_A, σ) is the result of decrypting c under the secrete key corresponding to ID_B, **Verify** $(m, ID_A, \sigma) = \top$. Note that, since we are assuming that $\mathcal{A}$ has been successful, $ID_B \neq ID_A$ and so we can use the decryption process of the simulator. Also, if $\sigma = (X, Z)$ then, except with negligible probability, the H_1 query must have been made at some point during the simulation. We call this query the critical query.

Let ϕ, ψ be the random tapes for random oracle H_1 and simulator $\mathcal{B}_1$ respectively. That is to say ψ is the random tape for all functions of $\mathcal{B}_1$ except random oracle H_1. The random oracle H_1 is called $(q_1 + q_s + q_{sc})$ times. The second step in the process of solving DHP is to choose $j \leftarrow \{1, \ldots, (q_1 + q_s + q_{sc})\}$ at random. We now split ϕ into ϕ_1 and ϕ_2 where ϕ_1 contains the random responses for queries $1, \ldots, j-1$ and ϕ_2 contains responses for

queries $j, \ldots, (q_1 + q_s + q_{sc})$. The next step of the simulation, $\mathcal{B}_2$ is to run a simulator similar to $\mathcal{B}_1$ with the same ψ and ϕ_1 but a new ϕ_2, say ϕ_2'. With probability

$$1/(q_1 + q_s + q_{sc}) \tag{4}$$

the value of j that we chose corresponds to the critical query.

Our proof now uses the following lemma from [10].
Lemma: (The splitting lemma)
Let $E \subset \Theta \times \mathcal{L}$ be such that $\Pr[E] \geq \upsilon$. Define
$$F = \{(\theta, v) \in \Theta \times \mathcal{L} : \Pr_{v' \in \mathcal{L}}[(\theta, v') \in E] \geq \upsilon/2\}$$
we have the following
1. $\forall (\theta, v) \in F, \Pr_{v' \in \mathcal{L}}[(\theta, v') \in E] \geq \upsilon/2$
2. $\Pr[F \mid E] \geq 1/2$

Now, from (3), (4) and lemma1 applied with $\Theta = \psi \cup \phi_1$ and $\mathcal{L} = \phi_2$, with probability greater than equal to

$$\epsilon^2 \cdot \left(1 - \frac{q_s(q_1 + q_s + q_{sc})}{q}\right)^2 \cdot \left(1 - \frac{q_{sc}(q_1 + q_s + q_{sc})}{q}\right)^2 \cdot \frac{1}{4q_0^2(q_1 + q_s + q_{sc})^2} \tag{5}$$

the two simulated runs of $\mathcal{A}$ gives us two signatures (X, Z) and (X, Z') on m with the following properties. After the first run of the simulator there is an $h_1 \in L_1$ and after the second run there is an $h_1' \in L_1$ (the responses to the critical queries) such that
$$Z = rbP + (r + h_1)abP \text{ and}$$
$$Z' = rbP + (r + h_1')abP \tag{6}$$
where $X = rP + raP$. Assuming that $\mathcal{B}_1$ and $\mathcal{B}_2$ are successful, it is easy to see from (6) that the third stage of the simulation, $\mathcal{B}_3$ can compute
$$abP = (h_1 - h_1')^{-1}(Z - Z') \tag{7}$$
The result follows from (7) and (5).

Proof of Theorem 5.3:
We will show how an AUTH-IDGSC-CMA adversary $\mathcal{A}$ of IDGSC may be used to construct a simulator $\mathcal{B}$ that solves the BDH problem. Let (P, aP, bP, cP) be the instant of the BDH problem that we wish to solve.

The simulator $\mathcal{B}$ runs $\mathcal{A}$ with PKG public key $Ppub = cP$. It also creates algorithms to respond to queries made by $\mathcal{A}$ during its attack. To maintain consistency between queries made by $\mathcal{A}$, the simulator keeps lists as in the proof of Theorem 1.

Simulator: $H_0(ID_U)$
At the beginning of simulator choose i_a, i_b uniformly at random from $\{1, \ldots, q_0\}$ (subject to $i_a \neq i_b$). We show how to response to the ith query made by $\mathcal{A}$ below. Note that we assume that $\mathcal{A}$ does not make repeat queries.
— If $i = i_a$ then respond with $H_0(ID_U) = aP$ and set $ID_A = ID_U$.
— If $i = i_b$ then respond with $H_0(ID_U) = bP$ and set $ID_B = ID_U$.

— Else chooses x uniformly at random from $\mathbb{Z}_q^*$, compute $Q_U = xP$; compute $S_U = xP_{Pub}$; store (ID_U, Q_U, S_U, x) in L_0 and respond with Q_U.

Simulator: $H_1(X \,\|\, m)$ and $H_2(\omega)$ as in the proof of Theorem 1.

Simulator: Extract (ID_U)

      We will assume that $\mathcal{A}$ makes the query $H_0(ID_U)$ before it makes the extraction query for ID_U.

— If $ID_U = ID_A$ or $ID_U = ID_B$, abort the simulation.
— Else search L_0 for the entry (ID_U, Q_U, S_U, x) corresponding to ID_U and return S_U.

Simulator: Sign (m, ID_1, ID_ϕ)

      We will assume that $\mathcal{A}$ makes the queries $H_0(ID_1)$ before it makes a sign query using this identity. We have following cases to consider
Case1: $ID_1 \neq ID_A$ and $ID_1 \neq ID_B$
      Use the simulation from case1 of sign in the proof of Theorem 1.
Case2: $ID_1 = ID_A$
      Use the simulation from case2 of sign in the proof of Theorem 1 by replacing ID_β by ID_A and Q_β by Q_A.
Case3: $ID_1 = ID_B$
      Use the simulation from case2 of sign in the proof of Theorem 1 by replacing ID_β by ID_B and Q_β by Q_B.

Simulator: Signcrypt (m, ID_1, ID_2)

      We will assume that $\mathcal{A}$ makes the queries $H_0(ID_1)$ and $H_0(ID_2)$ before it makes a signcryption query using these identities. We have following cases to consider
Case1: $ID_1 \neq ID_A$ and $ID_1 \neq ID_B$
      Use the simulation from case1 of Signcrypt in the proof of Theorem 1.
Case2: $ID_1 = ID_A, ID_2 \neq ID_A$ and $ID_2 \neq ID_B$
      Use the simulation from case2 of Signcrypt in the proof of Theorem 1 by replacing ID_β with ID_A and Q_β with Q_A.
Case3: $ID_1 = ID_B, ID_2 \neq ID_A$ and $ID_2 \neq ID_B$
      Use the simulation from case2 of Signcrypt in the proof of Theorem 1 by replacing ID_β with ID_B and Q_β with Q_B.
Case4: $ID_1 = ID_A$ and $ID_2 = ID_B$
— If $(m, ID_A, X, Z, r, h_1) \in L_s$ for m then
- Choose $h_2 \in \{0,1\}^{k_0+k_1+n}$ uniformly at random
- Compute $y = h_2 \oplus Z \,\|\, ID_A \,\|\, m$
- Add $(ID_A, ID_B, X, y, Z, m, r, h_1, h_2)$ to L_{sc}

21

- Return (X, y)

— Else choose r, h_1 uniformly at random from $\mathbb{Z}_q^*$

- Compute $X = rP - h_1 Q_A$ and $Z = rP_{Pub}$
- Add $(X \| m, h_1)$ to L_1
- Choose $h_2 \in \{0,1\}^{k_0+k_1+n}$ uniformly at random
- Compute $y = h_2 \oplus Z \| ID_A \| m$
- Add $(ID_A, ID_B, X, y, Z, m, r, h_1, h_2)$ to L_{sc}
- Return (X, y)

Case5: $ID_1 = ID_B$ and $ID_2 = ID_A$

Use the simulator of case4 swapping (ID_A, Q_A, ID_B) with (ID_B, Q_B, ID_A).

Simulator: Decrypt $(X, y) ID_2$

We will assume that $\mathcal{A}$ makes the queries $H_0(ID_2)$ before it makes a decryption query for this identity. We have following cases to consider

Case1: $ID_2 \neq ID_A$ and $ID_2 \neq ID_B$

Use the simulator from case1 of decrypt in the proof of Theorem 1.

Case2: $ID_2 = ID_A$

Use the simulation from case2 of decrypt in the proof of Theorem1 by replacing ID_β by ID_A.

Case3: $ID_2 = ID_B$

Use the simulation from case2 of decrypt in the proof of Theorem 1 by replacing ID_β by ID_B and aP by bP.

Simulator: Unsigncrypt $(X, y), ID_2$

We will assume that $\mathcal{A}$ makes the queries $H_0(ID_2)$ before it makes an unsigncryption query for this identity. We have following cases to consider

Case1: $ID_2 \neq ID_A$ and $ID_2 \neq ID_B$

Use the simulator from case1 of Unsigncrypt in the proof of Theorem 1.

Case2: $ID_2 = ID_B$

— If $(ID_A, ID_B, X, y, Z, m, r, h_1, h_2) \in L_{sc}$ for some m. return m, (X, Z), ID_A.

— Else, add (X, y), ID_B to L_u and step through the list L_2 with entries (ω, h_2) as follows.

- Compute $Z \| ID_1 \| m = y \oplus h_2$
- If $ID_1 = ID_A$, $ID_1 = ID_\phi$ or $ID_1 = ID_A$, move to the next element in L_2 and begin again
- If $ID_1 \in L_0$ let $Q_1 = H_0(ID_1)$ and find S_1 in L_0 else move to the next element in L_2 and begin again
- If $X \| m \in L_1$ let $h_1 = H_1(X \| m)$, else move to the next element in L_2 and begin again
- Check that $\omega = e(Z - h_1 S_1, Q_B)$ and if not move to the next element in L_2 and begin again
- Check that $e(Z, P) = e(P_{Pub}, X + h_1 Q_1)$, if so return m, (X, Z) and ID_1 else move to the next element in L_2 and begin again

— If no message has been returned after stepping through the list L_2, step through the list L_{sc} as follows

- If the entry has the form $(ID_A, ID_B, X', y, Z, m', r, h'_1, h_2)$ the check that $X' = X$. If so continue, else move on to the next element of L_{sc} and begin again.
- Else if the current entry has the form $(ID_B, ID_A, X', y, Z, m', r, h'_1, h_2)$ then checks that $e(X', Q_A) = e(X, Q_B)$. If so continue, if not move to the next element of L_{sc} and begin again.
- Compute $Z \| ID_1 \| m = y \oplus h_2$
- If $ID_1 = ID_B$ or $ID_1 = ID_\phi$, move to the next element in L_{sc} and begin again.
- If $ID_1 \in L_0$ let $Q_1 = H_0(ID_1)$, else move to the next element in L_{sc} and begin again
- If $X \| m \in L_1$ let $h_1 = H_1(X \| m)$, else move to the next element in L_{sc} and begin again
- Check that $e(Z, P) = e(P_{Pub}, X + h_1 Q_1)$, if so return m, (X, Z) and ID_1 else move to the next element in L_{sc} and begin again

— If no message has been retuned, return $\perp$.

Case3: $ID_2 = ID_A$

Use the simulator of case2 replacing (ID_B, Q_B, ID_A) by (ID_A, Q_A, ID_B).

Once $\mathcal{A}$ has been run, $\mathcal{B}$ does one of the two things.

1. With probability $q_{sc}/(q_{sc} + q_u)$ choose a random element from L_{sc} and a random element (ω, h_2) from L_2. We call this event Ch_1 in the analysis below (Ch for choice). This is the worst case scenario.

— If the chosen element has form $(ID_A, ID_B, X, y, Z, m, r, h_1, h_2)$, compute

$$B = (\omega / e(rbP, cP))^{-1/h_1}$$

— If the chosen element has form $(ID_B, ID_A, X, y, Z, m, r, h_1, h_2)$, compute

$$B = (\omega / e(raP, cP))^{-1/h_1}$$

2. With probability $q_u/(q_{sc} + q_u)$ choose a random element from L_u and a random element (ω, h_2) from L_2. We call this event Ch_2 in the analysis below. This is the worst case scenario.

— If the chosen element has form (X, y), ID_B compute $y \oplus h_2$. If $y \oplus h_2$ has the form $Z \| ID_A \| m$ for some Z, m compute

$$B = (\omega / e(Z, bP))^{-1/h_1}$$

If $y \oplus h_2$ does not have this form $\mathcal{B}$ has failed.

— If the chosen element has form (X, y), ID_A compute $y \oplus h_2$. If $y \oplus h_2$ has the form $Z \| ID_B \| m$ for some Z, m compute

$$B = (\omega / e(Z, aP))^{-1/h_1}$$

If $y \oplus h_2$ does not have this form $\mathcal{B}$ has failed.

Let us now consider how our simulation could fail i.e. describe events that could cause $\mathcal{A}$'s view to differ when run by $\mathcal{B}$ from its view in a real attack. We call such an event error and denote it ER.

It is clear that the simulations for H_0, H_1 are indistinguishable from real random oracles. Let us now consider the H_2 simulator. The important point here is that H_2 is not only defined at points where the H_2 simulator is called by $\mathcal{A}$ or by the simulator itself. It is also defined at certain points implicitly by the Signcrypt simulator. For example, suppose that the Signcrypt simulator responds to a query m, ID_A, ID_B. In this case it adds an entry $(ID_A, ID_B, X, y, Z, m, r, h_1, h_2)$ to L_{sc}. This implicitly defines $H_2\big(e(X, S_B)\big) = h_2$ although it is not actually able to compute $e(X, S_B)$. If the H_2 simulator is subsequently called with $\omega = e(X, S_B)$ it will not recognize it so it will not return h_2. We denote such events H-ER. However, such an event occurs we have

$$\omega = e(X, S_B) = e(rP - h_1 Q_A, S_B)$$

for which it is possible to compute

$$e(P, P)^{abc} = e(Q_A, S_B) = \big(\omega/e(rQ_B, P_{Pub})\big)^{-1/h_1} = \big(\omega/e(rbP, cP)\big)^{-1/h_1} \tag{1}$$

Similarly if the H_2 simulator is called with ω that is implicitly defined by an entry $(ID_B, ID_A, X, y, Z, m, r, h_1, h_2) \in L_{sc}$ we can compute

$$e(P, P)^{abc} = e(Q_B, S_A) = \big(\omega/e(rQ_A, P_{Pub})\big)^{-1/h_1} = \big(\omega/e(raP, cP)\big)^{-1/h_1} \tag{2}$$

Let us now consider how the simulation for sign could fail. We denote such an event S-ER. The most likely failure will caused by the sign simulator responded to the query of the form case 2 or case 3 (see simulator). Since we do not know how often each case will occur. We will be conservative and assume that each query will be one of these, 2 say. The only possibility for introducing an error here defining $H_1(X \| m)$ when it is already defined. Since X takes its values uniformly at random in $\langle P \rangle$, the chance of these events occurring $(q_1 + q_s + q_{sc})/q$ for each query. The q_s and q_{sc} come from the fact that the sign and Signcrypt simulator adds elements to L_1. Therefore over the whole simulation, the chance of an error introduced in this way is at most

$$\frac{q_s(q_1 + q_s + q_{sc})}{q} \tag{3}$$

Now we consider how the simulator for Signcrypt could fail. We denote such an error by SC-ER. The most likely failure will cause by the Signcrypt simulator responding to the query of the form case 4 or case 5 (see simulator). Again we do not know how often each case will occur so assume that each query will be one of these, 4 say. The only possibility for introducing an error here defining $H_1(X \| m)$ when it is already defined or defining $H_2\big(e(X, S_B)\big)/H_2\big(e(X, S_A)\big)$ when it is already defined. The chance of one of these events occurring $(q_1 + q_2 + q_s + 2q_{sc})/q$ for each query. Therefore over the whole simulation, the chance of an error introduced in this way is at most

$$\frac{q_{sc}(q_1 + q_2 + q_s + 2q_{sc})}{q} \tag{4}$$

Also the decrypt simulator is indistinguishable from genuine random oracles. We now turn our attention to the Unsigncrypt simulator. An error in this simulator is denoted by $U - ER$. It is clear that this simulator never accepts an invalid encryption. What we worry about is the possibility that it rejects a valid one. This can only occur with non-negligible

probability in case 2 or case 3. Suppose that we are trying to decrypt (X, y), ID_B (i.e. case 2). An error will only occur if while stepping through L_2 there is an entry (ω, h_2) such that $Z \| ID_A \| m = y \oplus h_2$ and (X, y) is a valid encryption of m from ID_A to ID_B. In this case we must have

$$\omega = e(Z - h_1 S_A, Q_B) = e(Z, Q_B)e(-h_1 S_A, Q_B) = e(Z, bP)e(-h_1 acP, bP)$$

where $h_1 = H_1(X \| m)$. From the above we can compute

$$e(P, P)^{abc} = \left(\omega / e(Z, bP)\right)^{-1/h_1} \tag{5}$$

Suppose we are trying to decrypt (X, y), ID_A (i.e. case 3). An error will only occur if while stepping through L_2 there is an entry (ω, h_2) such that $Z \| ID_B \| m = y \oplus h_2$ and (X, y) is a valid encryption of m from ID_B to ID_A. In this case we must have

$$\omega = e(Z - h_1 S_B, Q_A) = e(Z, Q_A)e(-h_1 S_B, Q_A) = e(Z, aP)e(-h_1 bcP, aP)$$

where $h_1 = H_1(X \| m)$. From the above we can compute

$$e(P, P)^{abc} = \left(\omega / e(Z, aP)\right)^{-1/h_1} \tag{6}$$

The final simulator is the extract simulator. Note that the adversary will only succeed in its task with non-negligible probability if it queries H_0 with the two identities under which the encrypted and signed message it produces is supposed to be valid. Looking at the H_0 simulator we see that it chooses two H_0 queries made by the adversary and responds to these with group elements from the BDHP instance that it is trying to solve. The simulator hopes that these will be the signer identities for the adversary's encrypted and signed message. This will be the case with probability at least

$$\frac{1}{q_0(q_0 - 1)} \tag{7}$$

If this is not the case we say that an error has occurred in the extract simulator because, if the adversary tried to extract the private key for these identities the simulator would abort. An error in the extract simulator is denoted by E-ER.

Once $\mathcal{A}$ has been run by the simulator $\mathcal{B}$, there are two courses of action: Ch_1 and Ch_2 (as describe above). If Ch_1 has been chosen, we denote the event that $\mathcal{B}$ selects the correct elements to solve the BDHP from L_{sc} and L_2 by CG_1 (under the assumption that there are such correct elements in the lists at the end of the simulation). Likewise if Ch_2 has been chosen, we denote the event that $\mathcal{B}$ selects the correct elements from L_u and L_2 by CG_2.

With the events described above we have

$$Adv[\mathcal{B}] \geq \Pr[\neg E - ER \wedge H - ER \wedge \neg S - ER \wedge \neg SC - ER \wedge Ch_1 \wedge CG_1]$$
$$+ \Pr[U - ER \wedge \neg E - ER \wedge \neg H - ER \wedge \neg S - ER \wedge \neg SC - ER \wedge Ch_2 \wedge CG_2] \tag{8}$$

we have

$$\Pr[\neg E - ER \wedge H - ER \wedge \neg S - ER \wedge \neg SC - ER \wedge Ch_1 \wedge CG_1]$$
$$= \Pr[\neg E - ER \wedge \neg S - ER \wedge \neg SC - ER] \Pr[Ch_1 \wedge CG_1] \Pr[H - ER] \tag{9}$$

Also,

$$\Pr[U - ER \wedge \neg E - ER \wedge \neg H - ER \wedge \neg S - ER \wedge \neg SC - ER \wedge Ch_2 \wedge CG_2]$$
$$= \Pr[U - ER] \Pr[\neg E - ER \wedge \neg H - ER \wedge \neg S - ER \wedge \neg SC - ER] \Pr[Ch_2 \wedge CG_2] \tag{10}$$

Note that, in the event $\neg E - ER \wedge \neg H - ER \wedge \neg S - ER \wedge \neg SC - ER$ the adversary $\mathcal{A}$ is run by $\mathcal{B}$ in exactly the same way that it would we run in a real attack until the event U-ER occurs. Moreover, in the event $\neg E - ER \wedge \neg H - ER \wedge \neg S - ER \wedge \neg SC - ER$, $\mathcal{A}$ winning and U-ER are equivalent. This means (10) becomes

$$\Pr[U - ER \wedge \neg E - ER \wedge \neg H - ER \wedge \neg S - ER \wedge \neg SC - ER \wedge Ch_2 \wedge CG_2]$$

$$= \varepsilon \Pr[\neg E - ER \wedge \neg H - ER \wedge \neg SER \wedge \neg SC - ER] \Pr[Ch_2 \wedge CG_2] \tag{11}$$

From the definition of Ch_1, CG_1, Ch_2 and CG_2 above we have

$$\Pr[Ch_1 \wedge CG_1] = \frac{q_{sc}}{q_{sc+}q_u} \frac{1}{q_{sc}(q_2 + q_{sc})} = \frac{1}{(q_{sc} + q_u)(q_2 + q_{sc})} \tag{12}$$

$$\Pr[Ch_2 \wedge CG_2] = \frac{q_u}{q_{sc+}q_u} \frac{1}{q_u(q_2 + q_{sc})} = \frac{1}{(q_{sc} + q_u)(q_2 + q_{sc})} \tag{13}$$

From, the fact that $\Pr[H - ER] + \Pr[\neg H - ER] = 1$, (8), (9), (11), (12) and (13) we have

$$Adv[\mathcal{B}] \geq (\Pr[H - ER] + \varepsilon \Pr[\neg H - ER]) \Pr[\neg E - ER \wedge \neg S - ER \wedge \neg SC - ER]$$

$$\frac{1}{(q_{sc} + q_u)(q_2 + q_{sc})}$$

$$\geq \varepsilon (\Pr[H - ER] + \Pr[\neg H - ER]) \Pr[\neg E - ER \wedge \neg S - ER \wedge \neg SC - ER]$$

$$\frac{1}{(q_{sc} + q_u)(q_2 + q_{sc})}$$

$$= \varepsilon \Pr[\neg E - ER \wedge \neg S - ER \wedge \neg SC - ER] \frac{1}{(q_{sc} + q_u)(q_2 + q_{sc})} \tag{14}$$

Finally, by the independence of E-ER, S-ER and SC-ER, using (3), (4), (7) and (14)

$$Adv[\mathcal{B}] \geq \varepsilon \left(1 - \frac{q_s(q_1 + q_s + q_{sc})}{q}\right)\left(1 - \frac{q_{sc}(q_1 + q_2 + q_s + 2q_{sc})}{q}\right) \frac{1}{q_0(q_0 - 1)(q_{sc} + q_u)(q_2 + q_{sc})}$$

as required.