

# Unlinkable Randomizable Signature and Its Application in Group Signature <sup>\*</sup>

Sujing Zhou<sup>1,2</sup> and Dongdai Lin<sup>1</sup>

<sup>1</sup> SKLOIS Lab, Institute of Software, Chinese Academy of Sciences,  
Beijing, P. R. China, 100080

<sup>2</sup> Graduate School of the Chinese Academy of Sciences,  
Beijing, P. R. China, 100039  
{zhousujing, ddlin}@is.iscas.ac.cn

**Abstract.** We formalize a generic method of constructing efficient group signatures, specifically, we define new notions of unlinkable randomizable signature, indirectly signable signature and  $\Sigma$ -protocol friendly signature.

We conclude that designing efficient secure group signatures can be boiled down to designing ordinary signatures satisfying the above three properties, which is supported by observations that almost all currently known secure efficient group signatures have alternative constructions in this line without deteriorating the efficiency.

**Keywords:** Digital Signature, Group Signature, Randomizable Signature, Sigma-protocol.

## 1 Introduction

In brief, a group signature scheme is composed of the following steps: (1) GM, the group manager, along with some third trusted party, chooses the security parameters as well as a group secret key and a group public key. (2) Any group member candidate is required to choose his *member secret key*, and run an interactive protocol with GM to join in the group, during which GM generates a signature on the member secret key blindly, i.e., not knowing the secret key value, the signature is also called *member certificate*. (3) Any group member can generate group signatures using his *group signing key* which includes member secret key and member certificate.

A common paradigm of constructing group signatures [1–4] is as follows: GM adopts an ordinary signature scheme to generate membership certificate for group members, i.e., sign on some secret key known only to members. The group signature is in fact a non-interactive zero-knowledge proof of knowledge of member certificate and member secret key, transformed in Fiat-Shamir’s heuristic method [5] from interactive proofs.

---

<sup>\*</sup> Supported by 973 Project of China (No.2004CB318004), 863 Project of China (No. 2003AA144030) and NSFC90204016. This is the full version of the paper at Inscrypt 07, and appended with an important note Appendix D.

Recently, a kind of randomizable signatures (given a signature of a message, someone other than the signer can get a new signature with respect to the same message) have been adopted in some schemes [6–9] to generate membership certificates. The following construction of group signature has been widely recognized: to sign on a message, a member firstly randomizes his member certificate, then generates a proof of knowledge of member secret key and part of the randomized member certificate. This method might result in more efficient group signature because the relation between member secret key and other items is much simplified due to concealing only part of the randomized member certificate instead of concealing it all in previous constructions.

We formalize the characteristics of randomizable signatures that are required to build secure efficient group signatures. Specifically, we define new notions of unlinkable randomizable signature, indirectly signable signature,  $\Sigma$ -protocol friendly signature.

We conclude that designing efficient secure group signatures can be boiled down to designing ordinary signatures satisfying the above three properties, which is supported by observations that almost all currently known secure efficient group signatures (except [10]) have alternative constructions in this line without deteriorating the efficiency, i.e., the signature schemes used to generate member certificates in the group signature can be modified into randomizable signatures with unlinkability, indirectly signability and  $\Sigma$ -protocol friendliness. For example, the scheme in [7] can be seen as the randomizable version of the well known ACJT scheme [4], satisfying the above three characteristics.

Apart from pointing out the obvious alternative constructions of some current group signatures, we propose the more complicated alternative constructions of others. They include the alternative construction of the scheme [11] from randomizable signatures (denoted as NSN04\*). We propose two new randomizable signatures (denoted Wat05+, ZL06+) resulting in new efficient group signatures. We also slightly improve the scheme with concurrent join [12] by replacing the member certificate generation signature with an randomizable signature (denoted as BBS04+).

**Organization.** The new notions of unlinkable randomizable signature, indirectly signable signature,  $\Sigma$ -protocol friendly signature are presented in Section 3, where you can also find the new randomizable signatures satisfying the above three properties: NSN04\*, Wat05+, ZL06+. A generic construction of group signatures from the above randomizable signature is described in Section 4.1 as well as its security analysis (Section 4.2). We present the slight improvement to the group signature with concurrent join [12] in Section 4.3.

## 2 Preliminary

**Notations.** If  $(P, V)$  is a non-interactive proof for relation  $\rho$ ,  $P(x, w, R)$  denotes the operation of generating a proof for  $(x, w) \in \rho$  under the common reference string  $R$ ,  $V(x, \pi, R)$  denotes the operation of verifying a proof  $\pi$ .

**Definition 1 (wUF-ATK[13])** A signature scheme  $DS=(Gen, Sig, Ver)$  is wUF-ATK secure ( $ATK \in \{CMA, ACMA\}$ ), i.e., weakly unforgeable against ATK attack, if for every probabilistic polynomial-time algorithm  $\mathcal{A}$ , it holds that

$$\text{Adv}_{DS, \mathcal{A}}^{wUF-ATK} = \Pr\{(pk, sk) \leftarrow Gen(1^k), (m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}_{Sig(sk, \cdot)}}(pk, ATK) : \\ Ver(pk, m, \sigma) = 1, m \notin Q\} < \epsilon(k)$$

where  $\epsilon(k)$  is a negligible function, the probability is taken over the coin tosses of algorithms  $Gen, Sig$  and  $\mathcal{A}$ .  $Q$  denotes the set of queries to oracle  $\mathcal{O}_{Sig(sk, \cdot)}$  made by  $\mathcal{A}$ .

### 3 The New Notions

#### 3.1 Unlinkable Randomizable Signature (URS)

**Definition 2 (Randomizable Signature)** A randomizable signature scheme is a digital signature scheme that has an efficient signature randomization algorithm  $Rnd$  besides algorithms  $(Gen, Sig, Ver)$ :

- $Gen: N \rightarrow K$ : a probabilistic polynomial-time algorithm with input  $k$  (called security parameter), output  $(pk, sk) \in K$ , where  $K$  is a finite set of possible keys;  $pk$  is called public key,  $sk$  is secret key kept to the signer, i.e., the owner of the instance of the signature scheme.
- $Sig: K \times M \rightarrow S$ : a probabilistic polynomial-time algorithm with input  $(sk, m)$ , where  $sk$  is the same output from  $K$  above,  $m \in M$ ,  $M$  is a finite set of possible messages. Output is  $\sigma = (\Upsilon, \Xi) \in S$ , where  $\Upsilon$  is randomly chosen and independent from  $m$ ,  $\Xi$  is calculated from  $\Upsilon, m$  and  $sk$ .
- $Ver: K \times M \times S \rightarrow \{0, 1\}$ : a deterministic polynomial-time algorithm with input  $(pk, m, \sigma)$ , output 1 if  $\sigma$  is valid, i.e.,  $\sigma$  is really computed by the owner of the signature instance, output 0 otherwise.
- $Rnd: M \times S \rightarrow S$ : a probabilistic polynomial-time algorithm with a message  $m$  and a signature  $(\Upsilon, \Xi)$  on it, output a  $(\Upsilon', \Xi') \neq (\Upsilon, \Xi)$  that is also a signature on  $m$ .

$\text{Exp}_{\mathcal{A}}^{unlink-b}(k), b \in \{0, 1\}$ :  $(pk, sk) \xleftarrow{\$} Gen(1^k), (m_0, \Upsilon_0, \Xi_0, m_1, \Upsilon_1, \Xi_1) \xleftarrow{\$} \mathcal{A}(sk, pk)$ , If  $Ver(pk, m_0, (\Upsilon_0, \Xi_0)) = 0$  or  $Ver(pk, m_1, (\Upsilon_1, \Xi_1)) = 0$ , return 0.  $(\Upsilon', \Xi') \xleftarrow{\$} Rnd(m_b, \Upsilon_b, \Xi_b), b' \leftarrow \mathcal{A}(sk, pk, \Xi')$ . return  $b'$ .

**Definition 3 (Perfectly Unlinkable)** A randomizable signature  $rDS = (Gen, Sig, Ver, Rnd)$  is perfectly unlinkable if for any algorithm  $\mathcal{A}$ , the distribution of output of  $\text{Exp}_{\mathcal{A}}^{unlink-b}(k)$  (defined above) are the same for  $b \in \{0, 1\}$ , that is

$$\Pr\{\text{Exp}_{\mathcal{A}}^{unlink-1}(k) = 1\} = \Pr\{\text{Exp}_{\mathcal{A}}^{unlink-0}(k) = 1\}.$$

The above equation is identical to

$$\Pr\{\Xi' \xleftarrow{\$} Rnd(m_1, \Upsilon_1, \Xi_1) | (pk, sk) \xleftarrow{\$} Gen(1^k), ((m_0, \Upsilon_0, \Xi_0), (m_1, \Upsilon_1, \Xi_1)) \xleftarrow{\$} \mathcal{A}(sk)\} \\ = \Pr\{\Xi' \xleftarrow{\$} Rnd(m_0, \Upsilon_0, \Xi_0) | (pk, sk) \xleftarrow{\$} Gen(1^k), ((m_0, \Upsilon_0, \Xi_0), (m_1, \Upsilon_1, \Xi_1)) \xleftarrow{\$} \mathcal{A}(sk)\}.$$

**Definition 4 (Statistically Unlinkable)** A randomizable signature  $rDS = (Gen, Sig, Ver, Rnd)$  is statistically unlinkable if for any algorithm  $\mathcal{A}$ , the statistical distance between output of  $\text{Exp}_{\mathcal{A}}^{\text{unlink}^{-b}}(k)$  (defined above) for  $b \in \{0, 1\}$  is negligible, that is

$$\sum |\Pr\{\text{Exp}_{\mathcal{A}}^{\text{unlink}^{-1}}(k) = 1\} - \Pr\{\text{Exp}_{\mathcal{A}}^{\text{unlink}^{-0}}(k) = 1\}| < \epsilon(k),$$

where the sum is over all random choices of  $Gen$ ,  $\mathcal{A}$  and  $Rnd$ .

**Definition 5 (Computationally Unlinkable)** A randomizable signature  $rDS = (Gen, Sig, Ver, Rnd)$  is computationally unlinkable if for any probabilistic polynomial time algorithm  $\mathcal{A}$ , the probability between output of  $\text{Exp}_{\mathcal{A}}^{\text{unlink}^{-b}}(k)$  (defined above) for  $b \in \{0, 1\}$  is negligible, that is

$$\Pr\{\text{Exp}_{\mathcal{A}}^{\text{unlink}^{-1}}(k) = 1\} - \Pr\{\text{Exp}_{\mathcal{A}}^{\text{unlink}^{-0}}(k) = 1\} < \epsilon(k)$$

The above definitions of unlinkability can be further weakened by not allowing the adversary obtain the secret key, but granting access to signing oracle  $\mathcal{O}_{sig}(sk, \cdot)$  as in experiment  $\text{Exp}_{\mathcal{A}}^{w\text{-unlink}^{-b}}(k)$  defined below. Then we get weak perfectly unlinkability, weak statistically unlinkability, weak computationally unlinkability analogously.

$\text{Exp}_{\mathcal{A}}^{w\text{-unlink}^{-b}}(k), b \in \{0, 1\}$ :  $(pk, sk) \xleftarrow{\$} \text{Gen}(1^k), (m_0, \Upsilon_0, \Xi_0, m_1, \Upsilon_1, \Xi_1) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{sig}(sk, \cdot)}(pk)$ , If  $\text{Ver}(pk, m_0, \langle \Upsilon_0, \Xi_0 \rangle) = 0$  or  $\text{Ver}(pk, m_1, \langle \Upsilon_1, \Xi_1 \rangle) = 0$ , return 0.  $(\Upsilon', \Xi') \xleftarrow{\$} \text{Rnd}(m_b, \Upsilon_b, \Xi_b), b' \leftarrow \mathcal{A}^{\mathcal{O}_{sig}(sk, \cdot)}(pk, \Xi')$ . return  $b'$ .

**Definition 6 (Unlinkable Randomizable Signature)** A (perfectly, statistically, computationally) URS  $urDS = (Gen, Sig, Ver, Rnd)$  is a randomizable signature that is also (perfectly, statistically, computationally) unlinkable respectively.

### 3.2 $\Sigma$ -protocol Friendly Randomizable and Indirectly Signable Signature

**Definition 7 ( $\Sigma$ -protocol Friendly Randomizable Signature)** A randomizable signature  $rDS = (Gen, Sig, Ver, Rnd)$  is  $\Sigma$ -protocol friendly if there exists a  $\Sigma$ -protocol  $\mathcal{P}$  for relation  $\mathcal{R} = \{(\Xi, \langle \Upsilon, m \rangle) | \text{Ver}(pk, m, \langle \Upsilon, \Xi \rangle) = 1\}$ , that is [14]

- $\mathcal{P}$  is of 3-move form, and if Prover and Verifier follow the protocol, Verifier always accepts.
- From any  $\Xi$  and any pair of accepting conversations with different initial message from Prover on input the same  $\Xi$ , one can efficiently compute  $(\Upsilon, m)$  such that  $(\Xi, \langle \Upsilon, m \rangle) \in \mathcal{R}$ .
- There exists a polynomial time simulator  $M$ , which on input  $\Xi$ , and a random second message sent from Verifier, outputs an accepting conversation with the same probability distribution as between the honest Prover, Verifier on input  $\Xi$ .

The following concept of *indirectly signable* is actually a restatement of signatures on committed message [6].

**Definition 8 (Indirectly Signable)** *A signature is indirectly signable if there exists a one way function  $f$  (as defined in Chapter 9.2.4, [15] or more technically as in Chapter 2.2, [16]) and an efficient algorithm  $Sig_f$  that  $Sig(sk, m) = Sig_f(sk, f(m))$ . That is  $\Pr\{(pk, sk, f) \xleftarrow{\$} Gen(1^k), m \xleftarrow{\$} M, v \leftarrow f(m), \sigma \leftarrow Sig_f(sk, v) : Ver(pk, m, \sigma) = 1\} = 1$ , and for any probabilistic polynomial time algorithm  $\mathcal{A}$ ,  $\Pr\{(pk, sk, f) \xleftarrow{\$} Gen(1^k), m \xleftarrow{\$} M, v \leftarrow f(m), m' \leftarrow \mathcal{A}(sk, v) : m' = m\} < \epsilon(k)$ .*

Actually signatures with above characteristics have been proposed and adopted explicitly or implicitly [7, 6, 8, 9], see Table 1 (the scheme on the right is the corresponding URS signature with indirect signability and  $\Sigma$ -protocol friendliness with respect to the scheme on the left).

To illustrate the unlinkable randomness, take Scheme A in [6] as an example (shown in Table 1). If we set  $\mathcal{Y} = \text{NULL}$ ,  $\mathcal{X} = (a, b, c)$ , it is not even computationally unlinkable, because anyone can check if  $(m_1, a', b', c')$  or  $(m_0, a', b', c')$  is a valid signature. That is why group signatures adopting the above signature only result in selfless anonymity (a weaker anonymity where the adversary should not know the message  $m$ ) [9].

If we set  $\mathcal{Y} = (a)$ ,  $\mathcal{X} = (b, c)$ , then it is still not even computationally unlinkable, but is weak computationally unlinkable assuming DDH is hard over group  $\mathbb{G}_1$ .

If we further set  $\mathcal{Y} = (a, b)$ ,  $\mathcal{X} = (c)$ , then it is perfectly unlinkable. So it is rather easy to come up with an unlinkable randomizable signature, just reveal the randomized signature as less as possible. But revealing too little of the randomized signature may lose  $\Sigma$ -protocol friendliness.

ACJT [4]	CL02 [7]
Let $n = pq$ be an RSA modulus. $S_e = [2^{l_e} - 2^{\mu_e}, 2^{l_e} + 2^{\mu_e}]$ , $S_m = [2^{l_m} - 2^{\mu_m}, 2^{l_m} + 2^{\mu_m}]$ , $S_s = [2^{l_s} - 2^{\mu_s}, 2^{l_s} + 2^{\mu_s}]$ , $\mu_e > l_m$ .	
<p><i>Gen.</i> <math>a, c \xleftarrow{\\$} QR_n^*</math>, <math>sk = (p, q)</math>, <math>pk = (n, a, c, S_e, S_m)</math>.</p> <p><i>Sig.</i> If <math> m  = l_m</math>, <math>e \xleftarrow{\\$} S_e \cap \text{Prime}</math>, <math>A \leftarrow (a^m c)^{\frac{1}{e}} \bmod n</math>.</p> <p><i>Ver.</i> Given <math>m</math>, <math>(e, A)</math>, check if <math> m  = l_m</math>, <math>A^e = a^m c \bmod n</math>.</p> <p><i>Rnd.</i> -</p>	<p><i>Gen.</i> <math>a, b, c \xleftarrow{\\$} QR_n^*</math>, <math>sk = (p, q)</math>, <math>pk = (n, a, b, c, S_e, S_m, S_s)</math>.</p> <p><i>Sig.</i> If <math> m  = l_m</math>, <math>e \xleftarrow{\\$} S_e \cap \text{Prime}</math>, <math>s \xleftarrow{\\$} S_s</math>, <math>A \leftarrow (a^m b^s c)^{\frac{1}{e}} \bmod n</math>. <math>\mathcal{Y} = (e, s)</math>, <math>\mathcal{X} = (A)</math></p> <p><i>Ver.</i> Given <math>m</math>, <math>(\mathcal{Y}, \mathcal{X}) = (e, s, A)</math>, check if <math> m  = l_m</math>, <math> s  = l_s</math>, <math>A^e = a^m b^s c \bmod n</math>.</p> <p><i>Rnd.</i> Given <math>m</math>, <math>(\mathcal{Y}, \mathcal{X}) = (e, s, A)</math>, choose random <math>r</math> with length <math>l_r = l_s - l_e - 1</math>, <math>\mathcal{Y}' = (e, s + re)</math>, <math>\mathcal{X}' = (Ab^r)</math>.</p>
CL04 [6]	CL04+

Let $\mathbb{G}_1 = \langle g \rangle$ , $\mathbb{G}_2 = \langle \tilde{g} \rangle$ be $p$ order cyclic groups that there exists a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ . <sup>3</sup>	
<i>Gen.</i> $x, y \xleftarrow{\$} Z_p^*$ , $sk = (x, y)$ , $X = \tilde{g}^x$ , $Y = \tilde{g}^y$ , $pk = (p, g, \tilde{g}, \mathbb{G}_1, \mathbb{G}_2, e, X, Y)$ .	
<i>Sig.</i> $d \xleftarrow{\$} \mathbb{G}_1$ , $\Upsilon = \text{NULL}$ , $\Xi = (d, d^y, d^{x+mx})$ .	<i>Sig.</i> $d \xleftarrow{\$} \mathbb{G}_1$ , $s \xleftarrow{\$} Z_p^*$ , $\Upsilon = (s)$ , $\Xi = (d^s, d^{sy}, d^{x+mx})$ .
<i>Ver.</i> Given $m$ , $(\Upsilon, \Xi) = (a, b, c)$ , check if $e(a, Y) = e(b, \tilde{g})$ , $e(a, X)e(b, X)^m = e(c, \tilde{g})$ .	<i>Ver.</i> Given $m$ , $(\Upsilon, \Xi) = (s, a, b, c)$ , check if $e(a, Y) = e(b, \tilde{g})$ , $e(a, X)e(b, X)^m = e(c, \tilde{g})^s$ .
<i>Rnd.</i> Given $m$ , $(\Upsilon, \Xi) = (a, b, c)$ , $r \xleftarrow{\$} Z_p^*$ , $\Upsilon' = \text{NULL}$ , $\Xi' = (a', b', c') = (a^r, b^r, c^r)$ .	<i>Rnd.</i> Given $m$ , $(\Upsilon, \Xi) = (s, a, b, c)$ , $r_1, r_2 \xleftarrow{\$} Z_p^*$ , $\Upsilon' = (s') = (r_2s)$ , $\Xi' = (a', b', c') = (a^{r_1r_2}, b^{r_1r_2}, c^{r_1})$ .
BBS04 [8]	BBS04+
<i>Gen.</i> $x \xleftarrow{\$} Z_p^*$ , $w = \tilde{g}^x$ , $h_1 \xleftarrow{\$} \mathbb{G}_1$ . $sk = (x)$ , $pk = (p, \mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, h_1, e, w)$ .	
<i>Sig.</i> $s \xleftarrow{\$} Z_p^*$ , $A \leftarrow (h_1^m g)^{\frac{1}{x+s}}$ .	<i>Sig.</i> $s, t \xleftarrow{\$} Z_p^*$ , $A \leftarrow (h_1^m g)^{\frac{t}{x+s}}$ , $\Upsilon = (s, t)$ , $\Xi = (A)$ .
<i>Ver.</i> Given $m$ , $(s, A)$ , check if $e(A, w\tilde{g}^s) = e(h_1^m g, \tilde{g})$ .	<i>Ver.</i> Given $m$ , $(\Upsilon, \Xi) = (s, t, A)$ , check if $e(A, w\tilde{g}^s) = e(h_1^m g, \tilde{g}^t)$ .
<i>Rnd.</i> -	<i>Rnd.</i> Given $m$ , $(\Upsilon, \Xi) = (s, t, A)$ , $r \xleftarrow{\$} Z_p^*$ , $\Upsilon' = (s, rt)$ , $\Xi' = (A^r)$ .

Table 1: Comparison of signatures and URS.

### 3.3 Some New Unlinkable Randomizable Signatures

**NSN04\***. As we have mentioned, the ACJT scheme [4] has an alternative construction utilizing URS CL02. As for the scheme in [11], no similar alternative has been proposed. In this section, we propose a new URS NSN04\*, which can be adopted to build a new efficient group signature.

[11]	<b>NSN04*</b> .
Let $\mathbb{G}$ be a $p$ order additive cyclic group, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}'$ a bilinear map on $\mathbb{G} = \langle P \rangle$ .	
<i>Gen.</i> $\gamma \xleftarrow{\$} Z_p^*$ , $P_{pub} = \gamma P$ , $P_0 \xleftarrow{\$} \mathbb{G}$ , $sk = (\gamma)$ , $pk = (p, \mathbb{G}, \mathbb{G}', P, P_0, P_{pub}, e)$ .	
<i>Sig.</i> $a \xleftarrow{\$} Z_p^*$ , $A = \frac{1}{\gamma+a}[mP + P_0]$ .	<i>Sig.</i> $(a, b, c) \xleftarrow{\$} Z_p^{*3}$ , $A = \frac{1}{\gamma+a}[mP + (b + \gamma c)P_{pub} + P_0]$ , $\Upsilon = (a, b, c)$ , $\Xi = (A)$ .
<i>Ver.</i> Given $m$ , $(a, A)$ , check if $e(A, P_{pub} + aP) = e(mP + P_0, P)$ .	<i>Ver.</i> Given $m$ , $(\Upsilon, \Xi) = (a, b, c, A)$ , check if $e(A, P_{pub} + aP) = e(mP + bP_{pub} + P_0, P)e(cP_{pub}, P_{pub})$ .
<i>Rnd.</i> -	<i>Rnd.</i> Given $m$ , $(\Upsilon, \Xi) = (a, b, c, A)$ , $r \xleftarrow{\$} Z_p^*$ , $\Upsilon' = (a', b', c') = (a, b + ra, c + r)$ , $\Xi' = (A') = (A + rP_{pub})$ .

<sup>3</sup> Note that the parameters here are according to the setup in [9], i.e., a SXDDH hard curve. We also have some comments on the two schemes at Appendix D.

**Lemma 1.** *NSN04\* is wUF-ACMA if  $q$ -SDH problem in  $\mathbb{G}$  is hard, where  $q$  is polynomial in  $|p|$ . See Appendix A for the proof.*

NSN04\* is indirectly signable if we define  $f(m) = mP$  assuming Computational Diffie-Hellman problem on  $\mathbb{G}$  is hard. Obviously, NSN04\* is perfectly unlinkable because each randomized  $\Xi'$  only consists of one element that is generated independently and randomly each time.

NSN04\* is  $\Sigma$ -protocol friendly, because there exists an efficient  $\Sigma$ -protocol for the relation  $\{(m, a, b, c) | e(A, P_{pub})e(A, P)^a = e(P, P)^m e(P_{pub}, P)^b e(P_0, P)e(P_{pub}, P_{pub})^c\}$ .

**Wat05+.** The recently proposed signature in [17], which is provable secure under CBDH assumption (Computational Bilinear Diffie-Hellman assumption) without random oracle, is also an URS if only we change a bit on it, see the following restatement with an extra algorithm *Rnd*.

<b>Wat05+</b>
<p>Let <math>\mathbb{G}, \mathbb{G}'</math> be two <math>p</math> order cyclic groups, and there exists a bilinear map <math>e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}'</math>. <math>\mathbb{G} = \langle g \rangle</math>.</p> <p><i>Gen.</i> Set secret key <math>sk = (x)</math>, <math>pk = (e, g_1, g_2, u, u', u_i, i = 0, \dots, l)</math>, where <math>g_1, g_2, u, u', u_i</math> are all elements from <math>\mathbb{G}</math>, <math>g_1 = g^x</math>, <math>l</math> is the maximum binary length of a message to be signed.</p> <p><i>Sig.</i> Given a message <math>m</math> with length at most <math>l</math>, the signature <math>(\Upsilon, \Xi)</math> is <math>\Upsilon = (s)</math>, <math>\Xi = (a, b) = (g^r, g_2^x (u' \prod_{i=1}^l u_i^{m_i})^r) u^s</math>, where <math>s \xleftarrow{\\$} Z_p</math>. Note that <math>(a, bu^{-s})</math> is a signature of <math>m</math> according to the scheme in [17].</p> <p><i>Ver.</i> Given a message <math>m</math> and its signature <math>(\Upsilon, \Xi) = (s, a, b)</math>, it is a valid signature on <math>m</math> if <math>e(b, g) = e(u', a) e(g_2, g_1) \prod_{i=1}^l e(u_i, a)^{m_i} e(u, g)^s</math>.</p> <p><i>Rnd.</i> On input <math>pk</math>, message <math>m</math>, and a signature <math>(\Upsilon, \Xi)</math>, where <math>\Upsilon = (s)</math>, <math>\Xi = (a, b)</math>, choose <math>(r_1, r_2) \xleftarrow{\\$} Z_p \times Z_p</math>, set <math>\Upsilon' = (s') = (s + r_1)</math>, <math>\Xi' = (a', b') = (ag^{r_2}, b(u' \prod_{i=1}^l u_i^{m_i})^{r_2} u^{r_1})</math>. The new randomized signature on <math>m</math> is <math>(\Upsilon', \Xi')</math>.</p>

Wat05+ is wUF-ACMA. The proof is easy, omitted here.

Wat05+ is  $\Sigma$ -protocol friendly, because there exists efficient  $\Sigma$ -protocol for the relation  $\{(m_1, \dots, m_l, s) | e(b, g) = e(u', a) e(g_2, g_1) \prod_{i=1}^l e(u_i, a)^{m_i} e(u, g)^s\}$ .

Wat05+ is indirectly signable if we define  $f(m) = \prod_{i=1}^l u_i^{m_i}$ , it is one way if  $l = O(k)$ , where  $k$  is the security parameter. That is because the probability of  $f(m) = f(m')$  for  $m \neq m'$  is about  $1/p$ , i.e., the solution to  $f(m) = c$  for a random  $c \in \mathbb{G}$  is unique non-negligibly. To obtain the unique solution,  $2^l$  tests must be carried out.

Wat05+ signature is perfectly unlinkable, because  $a'$  and  $b'$  are obtained from independent random variables.

Note that the original scheme Wat05 [17] is already utilized in the compact group signature [18]. But Wat05+ has not been adopted anywhere.

**ZL06+.** ZL06+ is a new URS similar to the standard signature proposed in [19].

<b>ZL06+</b> <sup>4</sup>
<p>Let <math>\mathbb{G}_1</math> be a <math>p</math> order cyclic group that exists a bilinear map <math>e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3</math>.  <math>\mathbb{G}_1 = \langle g \rangle</math>, <math>\mathbb{G}_2 = \langle \tilde{g} \rangle</math>.</p> <p><i>Gen.</i> Select <math>(x, y) \xleftarrow{\\$} Z_p^* \times Z_p^*</math>, set <math>X = g^x</math>, <math>Y = g^y</math>, <math>\tilde{X} = \tilde{g}^x</math>, <math>\tilde{Y} = \tilde{g}^y</math>. The secret key is <math>sk = (x, y)</math>, public key is <math>pk = (X, Y, \tilde{X}, \tilde{Y}, g, \tilde{g}, e, p)</math>.</p> <p><i>Sig.</i> Given a message <math>m \in Z_p</math>, its signature is <math>(\mathcal{Y}, \Xi)</math>, where <math>\mathcal{Y} = (s)</math>, <math>\Xi = (a, b) = (g^r, g^{r(x+my)+sx+xy})</math>, <math>(r, s) \xleftarrow{\\$} Z_p^* \times Z_p^*</math>.</p> <p><i>Ver.</i> Given a signature <math>(\mathcal{Y}, \Xi) = (s, a, b)</math> of <math>m</math>, check if <math>e(b, \tilde{g}) = e(a, \tilde{X}\tilde{Y}^m)e(X, \tilde{Y})e(X, \tilde{g})^s</math>. If the equation holds, then accept <math>(\mathcal{Y}, \Xi)</math> as a valid signature of <math>m</math>, otherwise reject it as invalid.</p> <p><i>Rnd.</i> On input <math>pk</math>, message <math>m</math>, and a signature <math>(\mathcal{Y}, \Xi) = (s, a, b)</math>, choose random <math>r_1, r_2 \in Z_p \times Z_p</math>, output <math>(\mathcal{Y}', \Xi')</math> where <math>\mathcal{Y}' = (s') = (s+r_1)</math>, <math>\Xi' = (a', b') = (ag^{r_2}, b(XY^m)^{r_2}X^{r_1})</math>.</p>

ZL06+ is wUF-ACMA secure under the assumption proposed in [19]. The proof is easy, omitted here.

ZL06+ is  $\Sigma$ -protocol friendly, because there exists efficient  $\Sigma$ -protocol for the relation  $\{(m, s) \mid e(b, \tilde{g}) = e(a, \tilde{X})e(a, \tilde{Y})^m e(X, \tilde{Y})e(X, \tilde{g})^s\}$ .

ZL06+ is indirectly signable if define  $f(m) = g^m$  assuming Computational Diffie-Hellman problem on  $\mathbb{G}_1$  is hard.

ZL06+ signature is perfectly unlinkable, because  $a'$  and  $b'$  are obtained from independent random variables.

## 4 Group Signature from URS

**Definition 9 (Group Signature [20])** *A group signature is a signature scheme composed of the following algorithms between GM (including IA, issuing authority, and OA, opening authority), group members and verifiers.*

- **Setup:** an algorithm run by GM (IA and OA) to generate group public key  $gpk$  and group secret key  $gsk$ .
- **Join:** a probabilistic interactive protocol between GM (IA) and a group member candidate. If the protocol ends successfully, the candidate becomes a new group member with a group signing key  $gsk_i$  including member secret key  $msk_i$  and member certificate  $cert_i$ ; and GM (IA) adds an entry for  $i$  (denoted as  $reg_i$ ) in its registration table  $reg$  storing the protocol transcript, e.g.  $cert_i$ . Sometimes the procedure is also separated into **Join** and **Iss**, where **Join** emphasize the part run by group members as well as **Iss** denotes the part run by IA.
- **GSig:** a probabilistic algorithm run by a group member, on input a message  $m$  and a group signing key  $gsk_i = (msk_i, cert_i)$ , returns a group signature  $\sigma$ .
- **GVer:** a deterministic algorithm which, on input a message-signature pair  $(m, \sigma)$  and GM's public key  $gpk$ , returns 1 or 0 indicating the group signature is valid or invalid respectively.

<sup>4</sup> Note that we have some comments on this scheme at Appendix D.



- **Open**: a deterministic algorithm which, on input a message-signature pair  $(m, \sigma)$ , secret key  $gsk$  of GM (OA), and the registration table  $reg$ , returns identity of the group member who signed the signature, and a proof  $\pi$ .
- **Judge**: a deterministic algorithm with output of *Open* as input, returns 1 or 0, i.e., the output of *Open* is valid or invalid.

#### 4.1 Generic Construction of GS

Select an URS  $DS = (K_s, Sig, Ver, Rnd)$  which is indirectly signable with a one way function  $f$ , a probabilistic public encryption  $PE = (K_e, Enc, Dec)$ .

Define the following relations:

$\rho: (x, w) \in \rho$  iff  $x = f(w)$ .

$\rho_1: ((pk_e, pk_s, C, \Xi), (w, \Upsilon, r)) \in \rho_1$  iff  $Ver(pk_s, w, (\Upsilon, \Xi)) = 1$  and  $C = Enc(pk_e, f(w), r)$  and  $(pk_s, \cdot) \leftarrow K_s, (pk_e, \cdot) \leftarrow K_e$ .

$\rho_2: ((pk_e, C, m), (w)) \in \rho_2$  iff  $Dec(pk_e, w, C) = m$  and  $(pk_e, \cdot) \leftarrow K_e$ .

Assume  $(P, V)$ ,  $(P_1, V_1)$  and  $(P_2, V_2)$  are non-interactive proofs for relation  $\rho$ ,  $\rho_1$  and  $\rho_2$ , which have access to common reference string  $R$ ,  $R_1$  and  $R_2$  respectively. Let  $SIM$ ,  $SIM_1$ ,  $SIM_2$  be their corresponding simulation algorithm. The detailed definition of non-interactive proof is referred to [20].

$(P, V)$  is also defined to be with an online extractor (in the random oracle model), i.e., it has the following features (let  $k$  be the security parameter) [21]:

① **Completeness**: For any random oracle  $H$ , any  $(x, w) \in \rho$ , and any  $\pi \leftarrow P^H(x, w, R)$ , it satisfies  $\Pr \{V^H(x, \pi, R) = 1\} \geq 1 - \epsilon_1(k)$ , where  $\epsilon_1(k)$  is a negligible function.

② **Online Extractor**: There exists a probabilistic polynomial time algorithm  $K$ , the online extractor, such that the following holds for any algorithm  $A$ . Let  $H$  be a random oracle,  $Q_H(A)$  be the answer sequence of  $H$  to queries from  $A$ . Let  $w \leftarrow K(x, \pi, Q_H(A))$ , then as a function of  $k$ ,  $\Pr\{(x, w) \notin \rho, V^H(x, \pi, R) = 1\} < \epsilon_2(k)$ , where  $\epsilon_2(k)$  is a negligible function.

GS is constructed as follows, see Table 5 for the details.

**Setup**. Select an instance of  $DS$  and  $PE$ , let secret key of  $DS$  be the secret key of IA, secret key of  $PE$  be the secret key of OA.

**Join**. User  $i$  selects its member secret key  $sk_i$  in message space of  $DS$ , computes  $pk_i = f(sk_i)$ , generates  $\pi$ , a non-interactive zero-knowledge proof of knowledge of  $sk_i$  for relation  $\rho$ . IA checks the correctness of  $\pi$  and generates a  $DS$  signature on  $sk_i$ :  $cert_i = Sig_f(sk_s, pk_i) = Sig(sk_s, sk_i)$ , sets  $reg_i = pk_i$ . The group signing key of  $i$  is  $gsk_i = (cert_i, sk_i)$ .

**GSig**. On input  $(gpk, gsk_i, m)$ , parse  $cert_i$  into  $(\Upsilon, \Xi)$ , firstly derive a new certification  $(\Upsilon', \Xi') = Rnd(gpk, sk_i, \Upsilon, \Xi)$ ; then encrypt  $pk_i$  with  $PE$ :  $C = Enc(pk_e, pk_i, r_i)$  where  $r_i$  is random; then generate  $\pi_1$ , a non-interactive zero-knowledge of proof of knowledge of  $(sk_i, \Upsilon', r_i)$  for relation  $\rho_1$ ; in the end, transform  $\pi_1$  into a signature on  $m$  using any method of transforming a non-interactive zero-knowledge proof into a signature [5, 22–24], we also use  $\pi_1$  to note the transformed signature for simplicity. The group signature on  $m$  is  $\sigma = (C, \Xi', \pi_1)$ .

**GVer.** On input  $(gpk, m, \sigma)$ , parse  $\sigma$  as  $(C, \Xi', \pi_1)$ , check the correctness of  $\pi_1$ , return 1 if it is correct, return 0 otherwise.

**Open.** On input  $(gpk, ok, reg, m, \sigma)$ , parse  $\sigma$  as  $(C, \Xi', \pi_1)$ . OA firstly checks the validity of the group signature  $\sigma$  on  $m$ , if it is not valid, stops; otherwise decrypts  $C$  to get  $M$ , and generates  $\pi_2$ , a proof of knowledge of decryption key  $ok$  for relation  $\rho_2$ . If  $M = pk_i$  for some  $pk_i$  in  $reg$ , return the corresponding index or identity and  $\pi_2$ , else returns zero and  $\pi_2$ .

**Judge.** Check the correctness of  $\pi_2$ , return 1 if it is correct, return 0 otherwise.

<p>Algorithm <b>Setup</b><math>(1^k)</math>:  <math>R \xleftarrow{\\$} \{0, 1\}^{P(k)}</math>, <math>R_1 \xleftarrow{\\$} \{0, 1\}^{P_1(k)}</math>,  <math>R_2 \xleftarrow{\\$} \{0, 1\}^{P_2(k)}</math>, <math>(pk_s, sk_s) \leftarrow K_s(1^k)</math>,  <math>(pk_e, sk_e) \leftarrow K_e(1^k)</math>,  <math>gpk = (R, R_1, R_2, pk_e, pk_s)</math>,  <math>ok = (sk_e)</math>, <math>ik = (sk_s)</math>.  return <math>(gpk, ok, ik)</math>.</p>	<p>Algorithm <b>Join</b>:  User <math>i \xrightarrow{pk_i, \pi}</math> IA: User selects <math>sk_i, pk_i = f(sk_i)</math>, <math>\pi = P(pk_i, sk_i, R)</math>  User <math>i \xleftarrow{cert_i}</math> IA: IA checks if <math>V(pk_i, \pi, R) = 1</math>, calculates <math>cert_i = Sig_f(sk_s, pk_i)</math>, sets <math>reg_i = pk_i</math>.  User <math>i</math>: sets <math>gsk_i = (pk_i, sk_i, cert_i)</math>.</p>
<p>Algorithm <b>GSig</b><math>(gpk, gsk_i, m)</math>:  Parse <math>cert_i</math> as <math>(\Upsilon, \Xi)</math>,  Parse <math>gpk</math> as <math>(R, R_1, R_2, pk_e, pk_s)</math>,  <math>(\Upsilon', \Xi') = Rnd(gpk, sk_i, \Upsilon, \Xi)</math>;  <math>C \leftarrow Enc(pk_e, pk_i, r_i)</math>, <math>r_i</math> random;  <math>\pi_1 = P_1(\langle pk_e, pk_s, m, C, \Xi' \rangle, \langle sk_i, \Upsilon', r_i \rangle, R_1)</math>.  <math>\sigma = (C, \Xi', \pi_1)</math>.  return <math>\sigma</math>.</p>	<p>Algorithm <b>GVer</b><math>(gpk, m, \sigma)</math>:  Parse <math>\sigma</math> as <math>(C, \Xi', \pi_1)</math>,  Parse <math>gpk</math> as <math>gpk = (R, R_1, R_2, pk_e, pk_s)</math>,  Return <math>V_1(\langle pk_e, pk_s, C, \Xi' \rangle, \pi_1, R_1)</math>.    (Note that <math>\pi_1</math> here denotes the signature on <math>m</math> transformed from the non-interactive proof.)</p>
<p>Algorithm <b>Open</b><math>(gpk, ok, reg, m, \sigma)</math>:  Parse <math>gpk</math> as <math>gpk = (R, R_1, R_2, pk_e, pk_s)</math>,  Parse <math>\sigma</math> as <math>(C, \Xi', \pi_1)</math>,  If <math>GVer(gpk, m, \sigma) = 0</math>, return <math>\perp</math>.  <math>M \leftarrow Dec(sk_e, C)</math>,  If <math>M = reg_i</math>, <math>\exists i</math>, <math>id \leftarrow i</math>, else <math>id \leftarrow 0</math>.  <math>\pi_2 = P_2(\langle pk_e, C, M \rangle, \langle sk_e \rangle, R_2)</math>,  return <math>(id, \tau)</math>, where <math>\tau = (M, \pi_2)</math>.</p>	<p>Algorithm <b>Judge</b><math>(gpk, reg, m, \sigma, i, M, \pi_2)</math>:  Parse <math>gpk</math> as <math>gpk = (R, R_1, R_2, pk_e, pk_s)</math>,  Parse <math>\sigma</math> as <math>(C, \Xi', \pi_1)</math>,  If <math>GVer(gpk, m, \sigma) = 0</math>, return <math>\perp</math>.  return <math>V_2(\langle pk_e, C, M \rangle, \pi_2, R_2)</math>.</p>

Table 5: Algorithms Setup, GSig, GVer, Open, Judge of GS.

**Comparison.** The above generic construction can be seen as a particular case of the construction in [20]:

In [20], the group signature is  $\sigma = (C, \pi_1) = (Enc(pk_e, \langle i, pk_i, \Upsilon, \Xi, s \rangle, r_i), \pi_1)$ , where  $s = S(sk_i, m)$  and  $\pi_1$  is a proof of knowledge of  $(pk_i, \Upsilon, \Gamma, s, r_i)$  satisfying  $Ver(pk_s, \langle i, pk_i \rangle, (\Upsilon, \Xi)) = 1$ ,  $C = Enc(pk_e, \langle i, pk_i, \Upsilon, \Xi, s \rangle, r_i)$ , and  $V(pk_i, m, s) = 1$ .  $(S, V)$  is the signature generation and verification algorithms of an independent signature scheme.

However in this construction, the group signature is  $\sigma = (C, \Xi', \pi_1) = (Enc(pk_e, pk_i, r_i), \Xi', \pi_1)$ , where  $\pi_1$  is a transformed signature of the proof of knowledge of  $(sk_i, \Upsilon', r_i)$  satisfying  $Ver(pk_s, sk_i, (\Upsilon', \Xi')) = 1$  and  $C = Enc(pk_e, f(sk_i), r_i)$ .

The construction is more efficient in that less items are encrypted in  $C$  and the relation between member secret key, member certificate and other items is much simplified, thus efficient proof of knowledge of encrypted context is obtained.

## 4.2 Security Proofs

The above generic group signature utilizing unlinkable randomizable signature can be proved secure according to the proof methods for the security results in [20] under a variant model (see Appendix B).

**Lemma 2.** *The above GS is anonymous if DS is computationally unlinkable, PE is IND-CCA2,  $(P_1, V_1)$  is a simulation sound, computational zero-knowledge proof,  $(P_2, V_2)$  is a computational zero-knowledge proof.*

**Lemma 3.** *The above GS is traceable if DS is wUF-ACMA,  $(P_1, V_1)$ ,  $(P_2, V_2)$  are sound proofs of knowledge and  $(P, V)$  is a proof of knowledge with online extractor (in random oracle model).*

**Lemma 4.** *The above GS is non-frameable if  $f(\cdot)$  is one way function,  $(P, V)$  is a computational zero-knowledge proof,  $(P_1, V_1)$  and  $(P_2, V_2)$  are sound proofs of knowledge.*

Note that there is a gap between the generic construction GS and the realization of it by adopting the  $\Sigma$ -protocol friendly URS' we have described earlier (the reason we require  $\Sigma$ -protocol friendliness is from efficiency consideration), because  $\Sigma$ -protocols (after they are transformed into non-interactive forms [5]) are not guaranteed simulation sound. It can be fixed in proof by utilizing rewinding techniques [25, 26] so that an adversary, even after it has been given simulated group signatures, can not generate a valid group signature unless the ciphertext therein is correctly constructed.

## 4.3 Improvement to a Group Signature

### Review of KY05's Scheme.

**Setup.** At first, select the following public parameters:

- two groups  $\mathbb{G}_1 = \langle g_1 \rangle$ ,  $\mathbb{G}_2 = \langle g_2 \rangle$  of order  $p$  (length is  $l_p$  bits), and there exists a bilinear map  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ .
- an RSA modulus  $n$  of  $l_n$  bits.
- three integer ranges  $S, S', S''$  where  $S' \subset S \subset Z_{\phi(n)}$ , the upper bound of  $S''$  is smaller than the lower bound of  $S$ .
- an RSA modulus  $N$  of  $l_N$  bits, choose  $G \in QR_{N^2}$  so that  $\langle G \rangle$  is also  $N$ -th residues,  $\#\langle G \rangle = \phi(N)/4$ .

Then IA selects ①  $\gamma, \delta \xleftarrow{\$} Z_p$ , set  $w = g_2^\gamma, v = g_2^\delta$ ; ②  $\alpha, \beta \xleftarrow{\$} Z_p, u \xleftarrow{\$} \mathbb{G}_1$ , set  $u' = u^{\alpha/\beta}, h = u^\alpha(u')^\beta = u^{2\alpha}$ ; ③  $g, f_1, f_2, f_3 \xleftarrow{\$} QR_n$ ; ④ a collision resistant hash function HASH.

OA selects ①  $a_1, a_2, a_3 \xleftarrow{\$} Z_{\lfloor N/4 \rfloor}$ , set  $H_1 = G^{a_1}, H_2 = G^{a_2}, H_3 = G^{a_3}$ ; ② a universal one-way hash function family UOHF, and a hash key  $hk$ .

Group public key  $gpk = \{g_1, g_2, u, u', h, w, v, g, f_1, f_2, f_3, n, N, G, H_1, H_2, H_3, hk, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \text{UOHF}\}$ . Group secret key  $gsk = \{\gamma, \delta, a_1, a_2, a_3\}$ .

**Join.** A user selects  $x = x_1x_2, x_1 \xleftarrow{\$} S''$ , sends  $x$  to IA; IA checks whether  $x \in S'$ , if that is the case, selects  $r \xleftarrow{\$} Z_p^*, s \xleftarrow{\$} Z_p^*$ , calculates  $\sigma \leftarrow g_1^{\overline{\gamma+x+\delta r}}$ , sends  $(r, s, \sigma)$  to the user; the user checks if  $e(\sigma, wg_2^s v^r) = e(g_1, g_2)^s$ , if so, sets  $cert = (x, r, s, \sigma), msk = (x_1, x_2)$ .

**GSig.** If a user with member certificate  $(x, \sigma, r)$  and member secret key  $(x_1, x_2)$  wants to generate a group signature on  $m$ , he firstly computes  $T_1, T_2, T_3, T_4, T_5, C_0, C_1, C_2$  as follows.

$T_1 = u^z, z \xleftarrow{\$} Z_p$  in  $\mathbb{G}_1$ ;  $T_2 = (u')^{z'}, z' \xleftarrow{\$} Z_p$  in  $\mathbb{G}_1$ ;  $T_3 = h^{z+z'}\sigma$  in  $\mathbb{G}_1$ ;  $T_4 = g^y f_1^{x_1}, y \xleftarrow{\$} S(1, 2^{l_n-2})$  in  $QR_n$ ;  $T_5 = g^{y'} f_2^{x_2} f_3^t, y' \xleftarrow{\$} S(1, 2^{l_n-2})$  in  $QR_n$ ;  $C_0 = G^t, t \xleftarrow{\$} S(1, 2^{l_N-2})$  in  $Z_{N^2}^*$ ;  $C_1 = H_1^t(1+N)^x$  in  $Z_{N^2}^*$ ;  $C_2 = \|(H_2 H_3^{\mathcal{H}(hk, C_0, C_1)})^t\|$  in  $Z_{N^2}^*$ .

Then he generates a signature of knowledge by applying the Fiat-Shamir heuristic [5] on a proof of knowledge of the fourteen witnesses  $\theta_z, \theta_{z'}, \theta_x, \theta_{xz}, \theta_{xz'}, \theta_r, \theta_{rz'}, \theta_{x_1}, \theta_{x_2}, \theta_y, \theta_{y'}, \theta_{yx_2}, \theta_t$  that satisfy the following relations:

$T_1 = u^{\theta_z},$	$T_2 = (u')^{\theta_{z'}},$	$1 = T_1^{-\theta_x} u^{\theta_{xz}}, 1 = T_2^{-\theta_x} (u')^{\theta_{xz'}},$
$1 = T_1^{-\theta_r} u^{\theta_{rz}},$	$1 = T_2^{-\theta_r} (u')^{\theta_{rz'}},$	$T_4 = g^{\theta_y} f_1^{\theta_{x_1}}, 1 = T_4^{-\theta_{x_2}} g^{\theta_{yx_2}} f_1^{\theta_x},$
$T_5 = g^{\theta_{y'}} f_2^{\theta_{x_2}} f_3^{\theta_t},$	$\theta_x \in S',$	$\theta_{x'} \in S', \quad C_0 = G^{\theta_t},$
$C_1 = H_1^{\theta_t} (1+N)^{\theta_x}, C_2 = (H_2 H_3^{\mathcal{H}(hk, C_0, C_1)})^{2\theta_t},$		
$e(g_1, g_2)/e(T_3, w) = e(T_3, v)^{\theta_r} e(T_3, g_2)^{\theta_x} e(h, g_2)^{-\theta_{xz} - \theta_{xz'}} e(h, v)^{-\theta_{rz} - \theta_{rz'}} e(h, w)^{-\theta_z - \theta_{z'}}$		

The realization of the above signature of knowledge is quite standard, so we omit it here. The output is  $(T_1, T_2, T_3, T_4, T_5, C_0, C_1, C_2, c, s_z, s_{z'}, s_{xz}, s_{xz'}, s_r, s_{rz}, s_{rz'}, s_x, s_{x_1}, s_{x_2}, s_y, s_{y'}, s_{yx_2}, s_t)$ .

**GVer.** The verification is achieved by checking the above proof of knowledge, omitted here.

**Open.** Firstly the group signature is verified as well as the relation  $C_2^2 = C_0^{2(a_2+a_3\mathcal{H}(hk, C_0, C^{-1}))}$  is checked. If all the tests pass, OA computes  $x = (C_1 C_0^{-a_1} - 1)/N$ , then checks if there exists a matching member certificate in the database maintained by IA.

#### Group Signature KY05+.

Replacing the member certificate signature with the following BB04+ signature, the scheme in [12] can be improved.

<b>BB04+</b>
Let $\mathbb{G}_1, \mathbb{G}_2$ be two $p$ order cyclic groups, and there exists a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3. \mathbb{G}_1 = \langle g \rangle, \mathbb{G}_2 = \langle \tilde{g} \rangle.$

*Gen.* It chooses  $x \xleftarrow{\$} Z_p^*$ ,  $y \xleftarrow{\$} Z_p^*$ , and sets  $sk = (x, y)$ ,  $pk = (p, \mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, X, Y, e)$ , where  $X = \tilde{g}^x$ ,  $Y = \tilde{g}^y$ .

*Sig.* On input message  $m$ , secret key  $sk$ , and public key  $pk$ , choose  $(s, t) \xleftarrow{\$} Z_p^{*2}$ , compute  $A = g^{\frac{t}{x+m+ys}}$ , output the signature  $(\mathcal{Y}, \Xi)$  where  $\mathcal{Y} = (s, t)$ ,  $\Xi = (A)$ . Note that  $(s, A^{\frac{1}{t}})$  is a valid [27] signature on  $m$ .

*Ver.* On input  $pk$ , message  $m$ , and purported signature  $(\mathcal{Y}, \Xi) = (s, t, A)$ , check that  $e(A, XY^s \tilde{g}^m) = e(g^t, \tilde{g})$ .

*Rnd.* On input  $pk$ , message  $m$ , and a signature  $(\mathcal{Y}, \Xi) = (s, t, A)$ , choose  $r \xleftarrow{\$} Z_p^*$ , output  $(\mathcal{Y}', \Xi')$  where  $\mathcal{Y}' = (s', t') = (s, rt)$ ,  $\Xi' = (A') = (A^r)$ .

It is easy to prove the wUF-ACMA of BB04+, similar to the original scheme [27]. Obviously, BB04+ is perfectly unlinkable because each randomized  $\Xi'$  only consists of one element that is generated independently and randomly each time, but it is not indirectly signable because  $m$  must be known to calculate a signature on it. BB04+ is  $\Sigma$ -protocol friendly, because there exists an efficient  $\Sigma$ -protocol for the relation  $\{(m, s, t) | e(A, X)e(A, Y)^s e(A, \tilde{g})^m = e(g, \tilde{g})^t\}$ .

Now we turn back to the group signature of KY05+. Public parameters and algorithms Setup, Join, Open are exactly as [12], except that key-setup for linear ElGamal encryption is eliminated.

**GSig.** If a user with member certificate  $(x, \sigma, r)$  and member secret key  $(x_1, x_2)$  wants to generate a group signature on  $m$ , he firstly computes  $(\sigma', s', T_4, T_5, C_0, C_1, C_2)$  as described in the following table.

$\sigma' = \sigma^{r'}$ , $s' = r's$	$r' \xleftarrow{\$} Z_p^*$ in $\mathbb{G}_1$
$T_4 = g^y f_1^{x_1}$	$y \xleftarrow{\$} S(1, 2^{l_n-2})$ in $QR_n$
$T_5 = g^{y'} f_2^{x_2} f_3^t$	$y' \xleftarrow{\$} S(1, 2^{l_n-2})$ in $QR_n$
$C_0 = G^t$	$t \xleftarrow{\$} S(1, 2^{l_N-2})$ in $Z_{N^2}^*$
$C_1 = H_1^t(1 + N)^x$	in $Z_{N^2}^*$
$C_2 = \ (H_2 H_3^{\mathcal{H}(hk, C_0, C_1)})^t\ $	in $Z_{N^2}^*$

Then he generates a signature of knowledge by applying the Fiat-Shamir heuristic [5] on a proof of knowledge of the nine witnesses  $(\theta_x, \theta_{x_1}, \theta_{x_2}, \theta_y, \theta_{y'}, \theta_{yx_2}, \theta_t, \theta_r, \theta_{s'})$  that satisfy the specified relations in the following table.

$g^{\theta_y} f_1^{\theta_{x_1}} = T_4$ ,	$g^{\theta_{y'}} f_2^{\theta_{x_2}} f_3^{\theta_t} = T_5$ ,	$T_4^{-\theta_{x_2}} g^{\theta_{yx_2}} f_1^{\theta_x} = 1$ ,
$e(\sigma', wg_2^{\theta_x} v^{\theta_r}) = e(g_1, g_2)^{\theta_{s'}}$ ,	$G^{\theta_t} = C_0$ ,	$H_1^{\theta_t}(1 + N)^{\theta_x} = C_1$ ,
$(H_2 H_3^{\mathcal{H}(hk, C_0, C_1)})^{2\theta_t} = C_2$ ,	$\theta_x \in S'$ ,	$\theta_{x'} \in S''$ .

Note that the number of witnesses that need proving is fewer than that of [12]. Thus a group signature of KY05+ is  $(\sigma', T_4, T_5, C_0, C_1, C_2, c, s_r, s_x, s_{x_1}, s_{x_2}, s_y, s_{y'}, s_{yx_2}, s_t, s_{s'})$ , about  $7|p| = 1190$  bits shorter than [12].

If we view  $x = x_1 x_2$  as a one way function since factoring of  $x$  is hard, KY05+ is an application of the proposed generic construction on BB04+ except that a non-interactive zero-knowledge proof of knowledge with online extractor

is not adopted in Join. The security of it follows from that of proposed generic construction and [12].

## References

1. J. Camenisch and M. Stadler, "Efficient group signatures schemes for large groups," in *Advances in Cryptology - CRYPTO'97*, LNCS 1296, pp. 410–424, Springer, 1997.
2. J. Camenisch and M. Michels, "A group signature scheme with improved efficiency," in *Advances in Cryptology - ASIACRYPT'98*, LNCS 1514, pp. 160–174, Springer, 1998.
3. J. Camenisch and M. Michels, "A group signature scheme based on an RSA-variant," in *Technical Report RS-98-27*, BRICS, University of Aarhus, 1998.
4. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *Advances in Cryptology - CRYPTO'00*, LNCS 1880, pp. 255–270, Springer, 2000.
5. A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," in *Advances in Cryptology - CRYPTO'86*, LNCS 263, pp. 186–194, Springer, 1987.
6. J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *Advances in Cryptology - CRYPTO'04*, LNCS 3152, pp. 56–72, Springer, 2004.
7. J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *Security in Communication Networks - SCN'02*, LNCS 2576, pp. 268–289, Springer, 2002.
8. D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology - CRYPTO'04*, LNCS 3152, pp. 45–55, Springer, 2004.
9. G. Ateniese, J. Camenisch, B. de Medeiros, and S. Hohenberger, "Practical group signatures without random oracles," *Cryptology ePrint Archive*, Report 2005/385.
10. X. Boyen and B. Waters, "Full-domain subgroup hiding and constant-size group signatures," in *Public Key Cryptography - PKC'07*, LNCS 4450, pp. 1–15, Springer, 2007.
11. L. Nguyen and R. Safavi-Naini, "Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings," in *Advances in Cryptology - ASIACRYPT'04*, LNCS 3329, pp. 372–386, Springer, 2004.
12. A. Kiayias and M. Yung, "Group signatures with efficient concurrent join," in *Advances in Cryptology - EUROCRYPT'05*, LNCS 3494, pp. 198–214, Springer, 2005.
13. H. Delfs and H. Knebl, *Introduction to Cryptography : Principles and Applications*. Springer, December 2001.
14. I. Damgård, *On Sigma-protocols*. <http://www.daimi.au.dk/~ivan/CPT.html>, 2005.
15. A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.
16. O. Goldreich, *Foundations of Cryptography*, vol. Basic Tools. Cambridge University Press, 2001.
17. B. Waters, "Efficient identity-based encryption without random oracles," in *Advances in Cryptology - EUROCRYPT'05*, LNCS 3494, pp. 114–127, Springer, 2005.
18. X. Boyen and B. Waters, "Compact group signatures without random oracles," in *Advances in Cryptology - EUROCRYPT'06*, pp. 427–444, Springer, 2006. Full paper at <http://eprint.iacr.org/2005/381>.

19. S. Zhou and D. Lin, “Shorter verifier-local revocation group signatures from bilinear maps,” in *Cryptology and Network Security - CANS’06*, LNCS 4301, pp. 126–143, Springer, 2006. Full text at <http://eprint.iacr.org/2006/286>.
20. M. Bellare, H. Shi, and C. Zhang, “Foundations of group signatures: The case of dynamic groups,” in *Topics in Cryptology - CT-RSA’05*, LNCS 3376, pp. 136–153, Springer, 2005. Full Paper at <http://www-cse.ucsd.edu/~mihir/papers/dgs.html>.
21. M. Fischlin, “Communication-efficient non-interactive proofs of knowledge with online extractor,” in *Advances in Cryptology - CRYPTO’05*, LNCS 3621, pp. 152–168, Springer, 2005.
22. M. Bellare and S. Goldwasser, “New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs,” in *Advances in Cryptology - CRYPTO’89*, LNCS 435, pp. 194–211, Springer, 1990.
23. R. Cramer and I. Damgård, “Secure signature schemes based on interactive protocols,” in *Advances in Cryptology - CRYPTO’95*, LNCS 963, pp. 297–310, Springer, 1995.
24. M. Chase and A. Lysyanskaya, “On signatures of knowledge,” in *Advances in Cryptology - CRYPTO’06*, LNCS 4117, pp. 78–96, Springer, 2006.
25. D. Pointcheval and J. Stern, “Security arguments for digital signatures and blind signatures,” *Journal of Cryptology*, vol. 13, no. 3, pp. 361–396, 2000.
26. A. Kiayias and M. Yung, “Secure scalable group signature with dynamic joins and separable authorities,” *International Journal of Security and Networks*, vol. 1, no. 1/2, pp. 24–45, 2006. Also titled with “Group Signatures: Provable Security, Efficient Constructions and Anonymity from Trapdoor-Holders” at <http://eprint.iacr.org/2004/076>.
27. D. Boneh and X. Boyen, “Short signatures without random oracles,” in *Advances in Cryptology - EUROCRYPT’04*, LNCS 3027, pp. 56–73, Springer, 2004.
28. D. Boneh and H. Shacham, “Group signatures with verifier-local revocation,” in *ACM conference on Computer and Communications Security - CCS’04*, pp. 168–177, ACM Press, 2004.
29. S. Zhou and D. Lin, “Unlinkable randomizable signature and its application in group signature,” in *Inscrypt’07*, 2007. Full paper at <http://eprint.iacr.org/2007/213>.
30. J. Groth, “Fully anonymous group signatures without random oracles,” in *Advances in Cryptology - ASIACRYPT’07*, LNCS ????, pp. ??–??, Springer, 2007.

## A Proof of Lemma 1

*Proof.* Suppose there exists an adversary  $\mathcal{B}$  to the signature, we now construct an adversary  $\mathcal{A}$  to resolve  $q$ -SDH problem ([27]) in  $\mathbb{G}$ : to calculate  $(c, \frac{1}{z+c}Q), c \in Z_p^*$  given a random tuple  $(Q, zQ, \dots, z^qQ)$ .

$\mathcal{B}$  should be given public key of the signature and access to oracle Sig answered by  $\mathcal{A}$ , obtaining  $q_{sig} (\leq q-1)$  message-signature pairs  $(m_i, a_i, b_i, c_i, A_i), i = 1, \dots, q_{sig}$ ,  $\mathcal{B}$  wins by outputting a forgery, i.e., a new message-signature  $(m^*, a^*, b^*, c^*, A^*)$  that  $m^* \notin \{m_1, \dots, m_{q_{sig}}\}$ . There may be two different types of forgeries. The first type,  $a^* \neq a_i, \forall i$ ; the second type,  $a^* = a_l, \exists l \in [1, q_{sig}]$ .  $\mathcal{A}$  will choose a random bit from  $\{1, 2\}$  to indicate its guess for the forgery type, and simulate accordingly. (Note that  $A = \frac{1}{\gamma+a}[mP + (b + \gamma c)P_{pub} + P_0] = \frac{1}{\gamma+a}[mP + (b - ac)P_{pub} + P_0] + cP_{pub}$ ).

**Type 1.**  $a^* \neq a_i, \forall i$ .

$\mathcal{A}$  selects  $a_i \xleftarrow{\$} Z_p^*, i \in [1, q_{sig}]$  that are not equal to each other, and  $s \xleftarrow{\$} Z_p^*$ , let  $f(y) = \prod_{i=1}^{q_{sig}} (y + a_i)$ ,  $\gamma = z$ , sets public key as  $P = f(z)Q$ ,  $P_{pub} = zf(z)Q$ ,  $P_0 = sf(z)Q$ , which are computable from  $(Q, zQ, \dots, z^qQ)$ .

When  $\mathcal{B}$  queries about a signature on  $m_i$ ,  $\mathcal{A}$  firstly selects  $b_i, c_i \xleftarrow{\$} Z_p^*$ , calculates  $A_i = \frac{1}{z+a_i} [m_iP + (b_i - a_i c_i)P_{pub} + P_0] + c_i P_{pub}$ , which is computable from  $(Q, zQ, \dots, z^qQ)$  since  $(z + a_i) | f(z)$ .

The forgery  $(m^*, a^*, b^*, c^*, A^*)$  satisfies  $A^* = \frac{1}{z+a^*} [m^*P + (b^* - a^* c^*)P_{pub} + P_0] + c^* P_{pub}$ , i.e.,  $A^* - c^* P_{pub} = \frac{1}{z+a^*} [(m^* + s + (b^* - a^* c^*)z) \prod_{i=1}^{q_{sig}} (z + a_i)Q]$ , the probability of  $m^* + s + (b^* - a^* c^*)z = 0 \pmod p$  is negligible otherwise  $\mathcal{B}$  can be invoked to solve discrete logarithm problem in  $\mathbb{G}$  if  $z$  is chosen by  $\mathcal{A}$  and  $sQ$  is given as a discrete logarithm challenge. Then there exist  $g(z), r \neq 0 \pmod p$  that  $(m^* + s + (b^* - a^* c^*)z) \prod_{i=1}^{q_{sig}} (z + a_i) = g(z)(z + a^*) + r$ , so  $(a^*, \frac{1}{z+a^*}Q)$ , computable from  $A^*$  and  $(Q, zQ, \dots, z^qQ)$ , is a resolution to the  $q$ -SDH challenge.

**Type 2.**  $a^* = a_l, \exists l \in [1, q_{sig}]$ .

$\mathcal{A}$  selects  $a_i \xleftarrow{\$} Z_p^*, i \in [1, q_{sig}]$  that are not equal to each other,  $t \xleftarrow{\$} Z_p^*$ , and  $d \xleftarrow{\$} Z_p^*$ , let  $f(y) = \prod_{i=1}^{q_{sig}} (y + a_i)$ ,  $\gamma = z - a_l$ , sets public key as  $P = \frac{f(z-a_l)}{z}Q = \prod_{i=1, i \neq l}^{q_{sig}} (z - a_l + a_i)Q$ ,  $P_{pub} = (z - a_l)P$ ,  $P_0 = tzP + dP = t \prod_{i=1}^{q_{sig}} (z - a_l + a_i)Q + dP$ , which are computable from  $(Q, zQ, \dots, z^qQ)$ .

When  $\mathcal{B}$  queries about a signature on  $m_i, i \neq l$ ,  $\mathcal{A}$  firstly selects  $b_i, c_i \xleftarrow{\$} Z_p^*$ , calculates  $A_i = \frac{1}{z-a_l+a_i} [m_iP + (b_i - a_i c_i)P_{pub} + P_0] + c_i P_{pub}$ , which is computable from  $(Q, zQ, \dots, z^qQ)$  since  $(z - a_l + a_i) | f(z - a_l)$ .

When  $\mathcal{B}$  queries about a signature on  $m_l$ ,  $\mathcal{A}$  firstly selects  $b_l, c_l, s \in Z_p^*$  so that  $b_l - a_l c_l = (d + m_l)a_l^{-1}$ , and  $s = t + (d + m_l)a_l^{-1}$ , then it can be verified that  $m_lP + (b_l - a_l c_l)P_{pub} + P_0 = szP$ , so  $A_l = \frac{1}{\gamma+a_l} [m_lP + (b_l - a_l c_l)P_{pub} + P_0] + c_l P_{pub} = sP + c_l P_{pub}$  is computable.

The forgery  $(m^*, a^*, b^*, c^*, A^*)$  satisfies  $A^* = \frac{1}{\gamma+a^*} [m^*P + (b^* - a^* c^*)P_{pub} + P_0] + c^* P_{pub}$ , i.e.,  $A^* - c^* P_{pub} = \frac{1}{z} [m^* - a_l(b^* - a^* c^*) + d + (b^* - a^* c^* + t)z] \prod_{i=1, i \neq l}^{q_{sig}} (z - a_l + a_i)Q$ , the probability of  $m^* - a_l(b^* - a^* c^*) + d = 0 \pmod p$  is negligible otherwise  $\mathcal{B}$  can be invoked to solve discrete logarithm problem in  $\mathbb{G}$  if  $z$  is chosen by  $\mathcal{A}$  and  $dQ$  is given as a discrete logarithm challenge. Then there exist  $g(z), r \neq 0 \pmod p$  that  $[m^* - a_l(b^* - a^* c^*) + d + (b^* - a^* c^* + t)z] \prod_{i=1, i \neq l}^{q_{sig}} (z - a_l + a_i) = g(z)z + r$ , so  $(0, \frac{1}{z}Q)$ , computable from  $A^*$  and  $(Q, zQ, \dots, z^qQ)$ , is a resolution to the  $q$ -SDH challenge. Note that any algorithm for  $\frac{1}{z}Q$  can be used to calculate a  $(c \neq 0, \frac{1}{z+c}Q)$ .

## B A Formal Model of Group Signature - A Variant of [20]

[20]'s model assumes that IA can not delete contents of the registration table Reg; OA is assumed only partially corrupted in considering traceability, i.e., OA will abide by specified algorithm Open. The existence of a secure (private and authentic) channel between any prospective group member and IA is also assumed.



For simplicity, we additionally assume that IA will not generate a new group signing key for an existing member, nor will IA modify existing records in Reg; OA will not report an existing member to be non-existent or another existing member after it has opened a group signature according to specified algorithms.

The additional assumption about IA can be guaranteed by introducing an additional trusted third authority CA independent from IA as explicitly defined in the model of [20]: every member is given a user public key from CA and a user secret key kept to himself; in Join, a member signed on whatever he has generated and sent to IA; IA stores the signed transcript in registration table; execution of Open should reveal the signer identity and stored transcript carrying a signature by the signer.

The additional assumption about OA can be guaranteed by granting accesses of reading/seaching Reg to judges (the executors of algorithm Judge).

We define the oracles similar to [20]. It is assumed that several global variables are maintained by the oracles:  $HU$ , a set of honest users;  $CU$ , a set of corrupted users;  $GSet$ , a set of message signature pairs; and  $Chlist$ , a set of challenged message signature pairs. Note that not all the oracles will be available to adversaries in defining a certain security feature.

*AddU* ( $i$ ): If  $i \in HU \cup CU$ , the oracle returns  $\perp$ , else adds  $i$  to  $HU$ , executes algorithm Join.

*CrptU* ( $i$ ): If  $i \in HU \cup CU$ , the oracle returns  $\perp$ , else  $CU \leftarrow CU \cup \{i\}$ , and awaits an oracle query to *SndToI*.

*SndToI* ( $i, M_{in}$ ): If  $i \notin CU$ , the oracle returns  $\perp$ ; else it plays the role of IA in algorithm Join replying to  $M_{in}$ .

*SndToU* ( $i, M_{in}$ ): If  $i \in HU \cup CU$ , the oracle returns  $\perp$ , else it plays the role of user  $i$  in algorithm Join,  $HU \leftarrow HU \cup \{i\}$ .

*USK* ( $i$ ): If  $i \in HU$ , the oracle returns  $sk_i$  and  $gsk_i$ ,  $CU \leftarrow CU \cup \{i\}$ ,  $HU \leftarrow HU \setminus \{i\}$ ; else returns  $\perp$ .

*RReg* ( $i$ ): The oracle returns  $reg_i$ .

*WReg* ( $i, s$ ): The oracle sets  $reg_i = s$  if  $i$  has not been added in reg.

*GSig* ( $i, m$ ): If  $i \notin HU$ , the oracle returns  $\perp$ , else returns a group signature  $\sigma$  on  $m$  by user  $i$ .  $GSet \leftarrow GSet \cup \{(i, m, \sigma)\}$ .

*Ch* ( $b, i_0, i_1, m$ ): If  $i_0 \notin HU \cup CU$  or  $i_1 \notin HU \cup CU$ , the oracle returns  $\perp$ , else generates a valid group signature  $\sigma$  with  $i_b$  being the signer.  $Chlist \leftarrow Chlist \cup \{(m, \sigma)\}$ .

*Open* ( $m, \sigma$ ): If  $(m, \sigma) \in Chlist$ , the oracle returns  $\perp$ , else if  $(m, \sigma)$  is valid, the oracle returns  $Open(m, \sigma)$ .

*CrptIA*: The oracle returns the secret key  $ik$  of IA.

*CrptOA*: The oracle returns the secret key  $ok$  of OA.

We say an oracle is over another oracle if availability of the oracle implies functions of another oracle. For example, *WReg* is over *RReg* since the adversary can try to remember everything it has written to Reg; *CrptIA* is over *CrptU*, *SndToI* since knowledge of  $ik$  enables the adversary answer the two oracles itself; *CrptOA* is over *Open*. Note that we do not let *CrptIA* (*CrptOA*) over *WReg*

(*RReg*) to provide flexibility when accesses to the database *Reg* are granted by an independent DBA (database administrator).

**Correctness.** For any adversary that is not computationally restricted, a group signature generated by an honest group member is always valid; algorithm *Open* will always correctly identify the signer given the above group signature; the output of *Open* will always be accepted by algorithm *Judge*.

<p>Experiment <math>\text{Exp}_{GS,A}^{corr}(k)</math>  <math>(gpk, ik, ok) \xleftarrow{\\$} \text{Setup}(1^k); HU \leftarrow \emptyset;</math>  <math>(i, m) \xleftarrow{\\$} A(gpk : \text{AddU}, RReg),</math>          If <math>i \notin HU</math>, return 0;  <math>\sigma \leftarrow \text{GSig}(gpk, gsk_i, m); (j, \tau) \leftarrow \text{Open}(gpk, ok, reg, m, \sigma),</math>          If <math>\text{GVer}(gpk, m, \sigma) = 0</math>, or <math>j \neq i</math>, or <math>\text{Judge}(gpk, i, reg, m, \sigma, \tau) = 0</math>,          then return 1 else return 0.</p>
--

**Table 10.** Correctness.

**Anonymity.** Imagine a polynomial time adversary  $\mathcal{A}$ , whose goal is to distinguish the signer of a group signature  $\sigma \leftarrow \text{Ch}(b, i_0, i_1, m)$  between  $i_0, i_1$ , where  $i_0, i_1, m$  are chosen by  $\mathcal{A}$  itself.

Naturally the adversary  $\mathcal{A}$  might want to get the group signing keys of  $i_0, i_1$  or some other honest group members (through oracle *USK*); it might want to obtain some group signatures signed by  $i_0, i_1$  (through oracle *GSig*); it might want to see some outputs of OA (through oracle *Open* except  $(m, \sigma)$ ); it might also try to corrupt some group members by running *Join* with IA (through oracles *CrptU* and *SndToI*); it might observe the communication of some honest members joining in (through *SndToU* if IA is corrupted, not available otherwise); it might want to write to, read from *Reg* (through oracles *WReg*, *RReg*); or  $\mathcal{A}$  might corrupt IA (through oracle *CrptIA*). Obviously  $\mathcal{A}$  should not be allowed to corrupt OA.

A group signature  $\text{GS}=(\text{Setup}, \text{Join}, \text{GSig}, \text{GVer}, \text{Open}, \text{Judge})$  is anonymous if the probability for any polynomial time adversary to win is negligible, i.e., the value of  $\text{Adv}_{GS,A}^{anon}$  defined below is negligible.

$$\text{Adv}_{GS,A}^{anon}(k) = \Pr\{\text{Exp}_{GS,A}^{anon-1}(k) = 1\} - \Pr\{\text{Exp}_{GS,A}^{anon-0}(k) = 1\},$$

where experiments  $\text{Exp}_{GS,A}^{anon-b}(k)$  are defined as in the above description.

If  $\{i_0, i_1\} \subseteq HU$ , and *CrptIA* is not queried, the group signature is *selfless anonymous* [28].

If  $\{i_0, i_1\} \subseteq CU$ , and *CrptIA* is not queried, the group signature is *anonymous* in the sense of [26].

If  $\{i_0, i_1\} \subseteq HU$ , and *CrptIA* is queried, the group signature is *anonymous* in the sense of [20].

We define a group signature GS is *anonymous* if  $\{i_0, i_1\} \subseteq CU$  and  $CrptIA$  is queried in the above game, (in this case  $GSig$  is implied if  $CrptIA$  is queried), i.e., the corresponding experiments are defined as in Table 11.

Experiment  $\text{Exp}_{GS,A}^{anon-b}(k)$ ,  $b \in \{0, 1\}$   
 $(gpk, ik, ok) \xleftarrow{\$} \text{Setup}(1^k)$ ;  $CU \leftarrow \emptyset$ ,  $HU \leftarrow \emptyset$ ,  $Chlist \leftarrow \emptyset$ ;  
 $d \xleftarrow{\$} A(gpk: CrptIA, Open, SndToU, USK, Ch(b, ., ., .), WReg)$ ,  
 Return  $d$ .

**Table 11.** Anonymity.

**Traceability.** Imagine a polynomial time adversary  $\mathcal{A}$ , whose goal is to produce a valid group signature  $(m, \sigma)$ , the output of  $Open$  on which points to a non-existent member or an existing corrupted member but can not pass Judge.

Naturally the adversary  $\mathcal{A}$  might corrupt some group members by running  $Join$  with  $IA$  (through oracles  $CrptU$  and  $SndToI$ ); it might want to see some outputs of  $OA$  (through oracle  $Open$ ); it might want to read from (through oracles  $RReg$ ); or  $\mathcal{A}$  might corrupt  $OA$  directly (through oracle  $CrptOA$ ). Obviously  $\mathcal{A}$  should not be allowed to corrupt  $IA$  and query  $WReg$ . Note that  $\mathcal{A}$  might not bother to query about honest group members for they are of little help for it.

A group signature GS is traceable if the probability for any polynomial time adversary to win is negligible, i.e., the value of  $\text{Adv}_{GS,A}^{trace}$  defined below is negligible.

$$\text{Adv}_{GS,A}^{trace}(k) = \Pr\{\text{Exp}_{GS,A}^{trace}(k) = 1\},$$

where experiment  $\text{Exp}_{GS,A}^{trace}(k)$  is defined as in the above description.

If  $CrptOA$  is not queried, the group signature is secure against *misidentification attack* [26].

If  $CrptOA$  is queried, the group signature is *traceable* in the sense of [20].

We define a group signature GS is *traceable* if  $CrptOA$  is queried in the above game, i.e., the corresponding experiment is defined as in Table 12.

Experiment  $\text{Exp}_{GS,A}^{trace}(k)$   
 $(gpk, ik, ok) \xleftarrow{\$} \text{Setup}(1^k)$ ;  $CU \leftarrow \emptyset$ ,  $HU \leftarrow \emptyset$ ;  
 $(m, \sigma) \xleftarrow{\$} A(gpk: CrptOA, CrptU, SndToI, RReg)$ .  
 If  $GVer(gpk, m, \sigma) = 0$ , return 0, else  $(i, \tau) \leftarrow \text{Open}(gpk, ok, Reg, m, \sigma)$ .  
 If  $i = 0$  or  $(\text{Judge}(gpk, reg, m, \sigma, \tau) = 0$  and  $i \in CU$ ) then return 1, else return 0.

**Table 12.** Traceability.

**Non-frameability.** Imagine a polynomial time adversary  $\mathcal{A}$ , whose goal is to produce a valid group signature  $(m, \sigma)$ , the output of `Open` on which points to an existing honest member  $i_h$  and the result passes `Judge`.

Naturally the adversary  $\mathcal{A}$  might want to get the group signing keys of some group members (through oracle  $USK$ ); it might want to obtain some group signatures signed by some honest group members (through oracle  $GSig$ ); it might want to see some outputs of `OA` (through oracle  $Open$ ); it might also try to corrupt some group members by running `Join` with `IA` (through oracles  $CrptU$  and  $SndToI$ ); it might observe the communication of some honest members joining in (through  $SndToU$  if  $CrptIA$  is queried, not available otherwise); it might wait until more group members has joined in (through  $AddU$ ); it might want to write to, read from, `Reg` (through oracles  $WReg, RReg$ ); or  $\mathcal{A}$  might corrupt `OA` or `IA` directly (through oracle  $CrptOA$  and  $CrptIA$ ). Obviously  $\mathcal{A}$  should not be allowed to query  $CrptU(i_h)$ ,  $SndToI(i_h, \cdot)$ ,  $USK(i_h)$ .

A group signature `GS` is non-frameable if the probability for any polynomial time adversary to win is negligible, i.e., the value of  $\text{Adv}_{GS, \mathcal{A}}^{nf}$  defined below is negligible.

$$\text{Adv}_{GS, \mathcal{A}}^{nf}(k) = \Pr\{\text{Exp}_{GS, \mathcal{A}}^{nf}(k) = 1\},$$

where experiment  $\text{Exp}_{GS, \mathcal{A}}^{nf}(k)$  is defined as in the above description.

If  $CrptIA$  and  $CrptOA$  are queried, the group signature is secure against *framing attack* [26] or non-frameable [20].

We define a group signature `GS` is *non-frameable* if  $CrptIA$ ,  $CrptOA$  are queried in the above game, and the corresponding experiment is defined as in Table 13.

<p>Experiment <math>\text{Exp}_{GS, \mathcal{A}}^{nf}(k)</math>  <math>(gpk, ik, ok) \xleftarrow{\\$} \text{Setup}(1^k); CU \leftarrow \emptyset, HU \leftarrow \emptyset, GSet \leftarrow \emptyset;</math>  <math>(m, \sigma, i, \tau) \xleftarrow{\\$} A(gpk : CrptIA, CrptOA, SndToU, GSig, USK, WReg).</math>          If <math>\text{GVer}(gpk, m, \sigma) = 0</math>, return 0.          Else if <math>i \in HU</math> and <math>\text{Judge}(gpk, reg, m, \sigma, \tau) = 1</math>          and <math>(i, m, \cdot) \notin GSet</math>, return 1, else return 0.</p>
--

**Table 13.** Non-frameability.

**Definition 10** *A group signature scheme is secure if it is anonymous, traceable and non-frameable.*

## C Security Proofs of the Generic Construction

### C.1 Proof of Lemma 2

Note that the difference between our construction 4 and the generic construction in [20] is that, our ultimate group signature is  $\sigma = (C, \Xi', \pi_1) = (\text{Enc}(pk_e, pk_i, r_i),$

$\Xi', \pi_1$ ), where  $\pi_1$  is a proof of knowledge of  $(sk_i, \mathcal{Y}', r_i)$  satisfying  $Ver(pk_s, sk_i, (\mathcal{Y}', \Xi')) = 1$  and  $C = Enc(pk_e, f(sk_i), r_i)$ ; while the ultimate group signature of [20] is  $\sigma = (C, \pi_1) = (Enc(pk_e, \langle i, pk_i, \mathcal{Y}, \Xi, s \rangle, r_i), \pi_1)$ , where  $s = S(sk_i, m)$  and  $\pi_1$  is a proof of knowledge of  $(pk_i, \mathcal{Y}, \Gamma, s, r_i)$  satisfying  $Ver(pk_s, \langle i, pk_i \rangle, (\mathcal{Y}, \Xi)) = 1$ ,  $C = Enc(pk_e, \langle i, pk_i, \mathcal{Y}, \Xi, s \rangle, r_i)$ , and  $V(pk_i, m, s) = 1$ .  $(S, V)$  is the signature generation and verification algorithms of an independent signature scheme.

So we have more information to expose than [20], i.e.,  $\Xi'$ , because the signature we adopted is *perfectly unlinkable*, it does not affect the anonymity of the generated group signature at all. Then we can follow the proof of [20].

The proof follows [20]. Suppose  $\mathcal{B}$  is an adversary to anonymity of GS, it can be invoked to construct an adversary  $\mathcal{A}_c, c \in \{0, 1\}$  to the public encryption scheme  $PE$ , an adversary  $\mathcal{A}_s$  to simulation soundness of  $(P_1, V_1)$ , adversaries  $\mathcal{D}_1$  and  $\mathcal{D}_2$  to zero-knowledge of  $P_1$  and  $P_2$  respectively, these adversaries will answer the oracle queries from  $\mathcal{B}$ .

**Description of  $\mathcal{A}_c$ .**  $\mathcal{A}_c$  is given the public key  $pk_e$  and accesses to oracles  $Ch_{PE}(b, \cdot, \cdot)$  and  $Dec(sk_e, \cdot)$ .

$\mathcal{A}_c$  selects keys  $(pk_s, sk_s)$  for  $DS$ , chooses common reference strings  $(R, R_1, R_2)$  for proofs  $P, SIM_1, SIM_2$ .  $\mathcal{A}_c$  gives  $gpk = (pk_e, pk_s, R, R_1, R_2)$  to  $\mathcal{B}$ .  $\mathcal{A}_c$  answers oracle queries from  $\mathcal{B}$  as follows:

*CrptIA*: returns  $sk_s$ .

*Open*  $(m, \sigma)$ : If  $(m, \sigma) = (m, C, \Xi', \pi_1)$  is valid and  $C$  is not returned by  $Ch(c, \cdot, \cdot)$ , queries oracle  $Dec(sk_e, \cdot)$ , and generates a simulation proof for  $\rho_2$ .

*SndToU*  $(i, \cdot)$ : Runs algorithm Join, adds  $i$  to the honest member set  $HU$ .

*USK*  $(i)$ : Returns  $(pk_i, sk_i, \mathcal{Y}, \Xi)$ , deletes  $i$  from  $HU$  and adds  $i$  to the corrupted member set  $CU$ .

*Ch* $(c, i_0, i_1)$ : If  $i_0, i_1$  are existing members, runs algorithm GSig on input  $(gpk, gsk_{i_c}, m)$  except that the encryption is replaced by the response from a query to  $Ch_{PE}(b, M_0, M_1)$  ( $M_c = (pk_{i_c}), M_{\bar{c}} = (0^{|M_c|})$ ), and the proof for  $\rho_1$  is replaced by  $SIM_1$ .

*WReg*  $(i, s)$ : If  $i$  is a new member, sets  $reg_i = s$ .

$\mathcal{A}_c$  outputs what  $\mathcal{B}$  outputs unless  $\mathcal{B}$  has generated a new group signature  $(m, \hat{\sigma}) = (m, C, \hat{\Xi}, \hat{\pi})$  from the challenge  $(m, \sigma) = (m, C, \Xi', \pi_1)$ , in which case  $\mathcal{A}_c$  outputs  $c$ .

**Description of  $\mathcal{A}_s$ .**  $\mathcal{A}_s$  is given the common reference string  $R_1$  of  $SIM_1$  and access to oracle  $SIM_1$ .

$\mathcal{A}_s$  setups GS as in algorithm Setup except that  $P_2$  is replaced by its simulation  $SIM_2$ .

$\mathcal{A}_s$  gives  $gpk = (pk_e, pk_s, R, R_1, R_2)$  to  $\mathcal{B}$ .  $\mathcal{A}_s$  answers oracle queries from  $\mathcal{B}$  as follows:

*CrptIA*: returns  $sk_s$ .

*Open*  $(m, \sigma)$ : If  $(m, \sigma) = (m, C, \Xi', \pi_1)$ , is valid and  $C$  is not returned by  $Ch(b, \cdot, \cdot)$ , runs algorithm Open since  $\mathcal{A}_s$  knows  $ok(= sk_e)$ , and generates a simulation proof for  $\rho_2$ .

*SndToU*  $(i, \cdot)$ : Runs as algorithm Join, adds  $i$  to the honest member set  $HU$ .

*USK* ( $i$ ): Returns  $(pk_i, sk_i, \mathcal{T}, \Xi)$ , deletes  $i$  from  $HU$  and adds  $i$  to the corrupted member set  $CU$ .

*Ch*( $b, i_0, i_1$ ): If  $i_0, i_1$  are existing members, runs algorithm *GSig* on input  $(gpk, gsk_{i_1}, m)$  except that always encrypts  $M_0 = (0^{|pk_1|})$  no matter the value of  $b$ , and the proof for  $\rho_1$  is replaced by the response from a query to  $SIM_1$ , returns  $(C, \Xi', \pi_1)$ .

*WReg* ( $i, s$ ): If  $i$  is a new member, sets  $reg_i = s$ .

$\mathcal{A}_s$  fails unless  $\mathcal{B}$  has generated a new group signature  $(m, \hat{\sigma}) = (m, C, \hat{\Xi}, \hat{\pi})$  from the challenge  $(m, \sigma) = (m, C, \Xi', \pi_1)$ , in which case  $\mathcal{A}_s$  outputs  $(pk_e, pk_s, m, C, \hat{\Xi})$  and  $\hat{\pi}$ .

**Description of  $\mathcal{D}_1$ .**  $\mathcal{D}_1$  is given the common reference string  $R_1$ , and access to oracle  $Prove_1(\cdot)$  which may be  $P_1$  or  $SIM_1$ .

$\mathcal{D}_1$  setups GS as in algorithm Setup except that  $P_2$  is replaced by a simulation  $SIM_2$ .

$\mathcal{D}_1$  gives  $gpk = (pk_e, pk_s, R, R_1, R_2)$  to  $\mathcal{B}$  and answers oracle queries from  $\mathcal{B}$  as follows:

*CrptIA*: returns  $sk_s$ .

*Open* ( $m, \sigma$ ): If  $(m, \sigma)$  is valid, runs algorithm *Open* since  $\mathcal{D}_1$  knows  $ok(=sk_e)$ , and generates a simulation proof for  $\rho_2$ .

*SndToU* ( $i, \cdot$ ): Runs as algorithm *Join*, adds  $i$  to the honest member set  $HU$ .

*USK* ( $i$ ): Returns  $(pk_i, sk_i, \mathcal{T}, \Xi)$ , deletes  $i$  from  $HU$  and adds  $i$  to the corrupted member set  $CU$ .

*Ch*( $b, i_0, i_1$ ): If  $i_0, i_1$  are existing members, runs algorithm *GSig* on input  $(gpk, gsk_{i_0}, m)$  except that generates  $\pi_1$  by querying oracle  $Prove_1$ .

*WReg* ( $i, s$ ): If  $i$  is a new member, sets  $reg_i = s$ .

$\mathcal{D}_1$  returns 1 if output of  $\mathcal{B}$  equals  $b$ , returns 0 otherwise.

**Description of  $\mathcal{D}_2$ .**  $\mathcal{D}_2$  is given the common reference string  $R_2$ , and access to oracle  $Prove_2(\cdot)$  which may be  $P_2$  or  $SIM_2$ .

$\mathcal{D}_2$  setups GS as in algorithm Setup.

$\mathcal{D}_2$  gives  $gpk = (pk_e, pk_s, R, R_1, R_2)$  to  $\mathcal{B}$  and answers oracle queries from  $\mathcal{B}$  as follows:

*CrptIA*: returns  $sk_s$ .

*Open* ( $m, \sigma$ ): If  $(m, \sigma)$  is valid, runs algorithm *Open* since  $\mathcal{D}_2$  knows  $ok(=sk_e)$ , and generates the proof for  $\rho_2$  by querying oracle  $Prove_2$ .

*SndToU* ( $i, \cdot$ ): Runs as algorithm *Join*, adds  $i$  to the honest member set  $HU$ .

*USK* ( $i$ ): Returns  $(pk_i, sk_i, \mathcal{T}, \Xi)$ , deletes  $i$  from  $HU$  and adds  $i$  to the corrupted member set  $CU$ .

*Ch*( $b, i_0, i_1$ ): If  $i_0, i_1$  are existing members, runs algorithm *GSig* on input  $(gpk, gsk_{i_0}, m)$ .

*WReg* ( $i, s$ ): If  $i$  is a new member, sets  $reg_i = s$ .

$\mathcal{D}_2$  returns 1 if output of  $\mathcal{B}$  equals  $b$ , returns 0 otherwise.

It follows from the same analysis in [20] that

$$\begin{aligned} \text{Adv}_{GS, \mathcal{B}}^{\text{anon}}(k) &\leq \text{Adv}_{PE, \mathcal{A}_0}^{\text{ind-cca}}(k) + \text{Adv}_{PE, \mathcal{A}_1}^{\text{ind-cca}}(k) + \text{Adv}_{SIM_1, \mathcal{A}_s}^{ss}(k) \\ &\quad + 2(\text{Adv}_{P_1, SIM_1, \mathcal{D}_1}^{zk}(k) + \text{Adv}_{P_2, SIM_2, \mathcal{D}_2}^{zk}(k)). \end{aligned}$$

### C.2 Proof of Lemma 3

The proof follows [20]. Suppose  $\mathcal{B}$  is an adversary to traceability of GS, it can be invoked to construct an adversary  $\mathcal{A}_{ds}$  to the digital signature scheme  $DS$ , the adversary will answer the oracle queries from  $\mathcal{B}$ .

**Description of  $\mathcal{A}_{ds}$ .**  $\mathcal{A}_{ds}$  is given the public key  $pk_s$  and access to oracle  $Sig(sk_s, \cdot)$ .

$\mathcal{A}_{ds}$  selects keys  $(pk_e, sk_e)$  for  $PE$ , chooses common reference strings  $R, R_1, R_2$  for relation  $\rho, \rho_1$  and  $\rho_2$  respectively.  $\mathcal{A}_{ds}$  gives  $gpk = (pk_e, pk_s, R, R_1, R_2)$  to  $\mathcal{B}$ .  $\mathcal{A}_{ds}$  answers oracle queries from  $\mathcal{B}$  as follows:

*CrptOA*: returns  $sk_e$ .

*CrptU* ( $i$ ): If  $i$  is not a group member yet, adds  $i$  to the corrupted members set  $CU$ .

*SndToI* ( $i, \cdot$ ): Parses the input into  $(pk_i, \pi)$  from which extracts  $sk_i$  using the online extractor algorithm  $K$  of  $(P, V)$  by manipulating the random oracle, queries oracle  $Sig(sk_s, sk_i)$ .

*RReg* ( $i$ ): If  $i$  exists in  $Reg$ , returns  $reg_i$ .

If  $\mathcal{B}$  wins with non-negligible probability, i.e., outputs a valid group signature  $(m, \sigma) = (m, C, \Xi', \pi_1)$  and  $i = 0$ , where  $(i, \tau) \leftarrow Open(sk_e, m, \sigma)$ . Another case that  $i > 0$  will not occur because of the correctness of GS and the assumptions for GS in our model (Appendix B).

From generalized forking lemma [26], (GVer be the predicate), in random oracle model, there exist  $(m, C, \Xi', c, s)$ ,  $(m, C, \Xi', c', s')$  from which  $(w, \Upsilon', r)$  can be extracted,  $(\Upsilon', \Xi')$  is a valid  $DS$  signature on  $w$ , and  $w$  is not queried to  $Sig(sk_s, \cdot)$ .

It follows from the same analysis in [20] that

$$\text{Adv}_{GS, \mathcal{B}}^{\text{trace}}(k) \leq 2^{-k} + \text{Adv}_{DS, \mathcal{A}_{ds}}^{wUF\text{-acma}}(k).$$

### C.3 Proof of Lemma 4

The proof follows [20]. Suppose  $\mathcal{B}$  is an adversary to non-frameability of GS, it can be invoked to construct an adversary  $\mathcal{A}_f$  to the one way function  $f$ , the adversary will answer the oracle queries from  $\mathcal{B}$ .

**Description of  $\mathcal{A}_f$ .**  $\mathcal{A}_f$  is given  $y$  in the range of the one way function  $f$ .

$\mathcal{A}_f$  sets up GS as in algorithm Setup, selects a random variable  $\iota \in [1, n(k)]$ ,  $n(k)$  is the maximum number of queries from  $\mathcal{B}$ .

$\mathcal{A}_f$  gives  $gpk = (pk_e, pk_s, R, R_1, R_2)$  to  $\mathcal{B}$  and answers oracle queries from  $\mathcal{B}$  as follows:

*CrptIA*: returns  $sk_s$ .

*CrptOA*: returns  $sk_e$ .

*SndToU* ( $i, \cdot$ ): If  $i = \iota$ , sets  $pk_i = y$ , and runs Join by simulating a proof for relation  $\rho$ ; otherwise runs exactly as algorithm Join. Then adds  $i$  to the honest member set  $HU$ .

*USK* ( $i$ ): If  $i = \iota$ ,  $\mathcal{A}_f$  stops and restarts again; otherwise if  $i \in HU$ , returns  $(pk_i, sk_i, \Upsilon, \Xi)$ , deletes  $i$  from  $HU$  and adds  $i$  to the corrupted member set  $CU$ .

*GSig* ( $i, m$ ): If  $i \in HU$  and  $i = \iota$ , runs algorithm *GSig* except that replacing proof  $P_1$  by the simulation  $SIM_1$ ; otherwise if  $i \in HU$ , runs *GSig* exactly.  $GSet \leftarrow GSet \cup \{(i, m, \sigma)\}$ .

*WReg* ( $i, s$ ): If  $i$  is a new member, sets  $reg_i = s$ .

$\mathcal{A}_f$  returns 1 if  $\mathcal{B}$  outputs a valid group signature that  $(\iota, m, \sigma) \notin GSet$  and  $Judge(gpk, reg, m, \sigma, \tau) = 1$  where  $(\iota, \tau) = Open(m, \sigma)$ .

Parse  $(\iota, m, \sigma)$  into  $(\iota, m, C, \Xi', c, s)$ , then there exist  $(\iota, m, C, \Xi', c, s)$ ,  $(m, C, \Xi', c', s')$  in random oracle model according to generalized forking lemma [26], (*GVer* be the predicate), so  $(w, \mathcal{Y}', r)$  can be extracted, where  $(\mathcal{Y}', \Xi')$  is a valid *DS* signature on  $w$ , and  $f(w) = y$ .

It follows from a similar analysis in [20] that  $\text{Adv}_{GS, \mathcal{B}}^{nf}(k) \leq \epsilon(k) + n(k) \text{Adv}_{f, \mathcal{A}_f}^{ow}(k)$ , where  $\epsilon(k)$  is negligible.

## D A Note on Some Group Signatures and Anonymous Credentials from Bilinear Maps

We show that some group signature schemes and anonymous credentials from bilinear maps are flawed, in particular, any one other than valid group members can also generate valid group signatures. We also provide a fix to some of the flawed schemes.

### D.1 Review of Camenisch-Lysyabskaya's Scheme

Camenisch-Lysyabskaya's group signature scheme is based on the following basic signature scheme:

Scheme A in [6].

Let  $\mathbb{G}_1 = \langle g \rangle$ ,  $\mathbb{G}_2 = \langle \tilde{g} \rangle$  be  $p$  order cyclic groups that there exists a bilinear map  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ .

- *Gen.*  $x, y \xleftarrow{\$} Z_p^*$ ,  $sk = (x, y)$ ,  $X = \tilde{g}^x$ ,  $Y = \tilde{g}^y$ ,  $pk = (p, g, \tilde{g}, \mathbb{G}_1, \mathbb{G}_2, e, X, Y)$ .
- *Sig.*  $d \xleftarrow{\$} \mathbb{G}_1$ ,  $\sigma = (d, d^y, d^{x+my})$ .
- *Ver.* Given  $m$ ,  $\sigma = (a, b, c)$ , check if  $e(a, Y) = e(b, \tilde{g})$  and  $e(a, X)e(b, X)^m = e(c, \tilde{g})$ .

To obtain a group signature, the above signature is firstly modified into the following scheme with randomization (as described in [29]).

- *Sig.*  $d \xleftarrow{\$} \mathbb{G}_1$ ,  $s \xleftarrow{\$} Z_p^*$ ,  $\sigma = (s, d^s, d^{sy}, d^{x+my})$ .
- *Ver.* Given  $m$ ,  $\sigma = (s, a, b, c)$ , check if  $e(a, Y) = e(b, \tilde{g})$  and  $e(a, X)e(b, X)^m = e(c, \tilde{g})^s$ .



- *Rnd.* Given  $m, \sigma = (s, a, b, c), r_1, r_2 \xleftarrow{\$} Z_p^*, \sigma' = (s', a', b', c') = (r_2 s, a^{r_1 r_2}, b^{r_1 r_2}, c^{r_1})$ .

The group signature is as follows: a group member chooses a member secret key  $k$ , sends  $g^k$  to the group manager; the group manager signs on  $k$  as above to get  $(s, a, b, c)$ , i.e., the member certificate; to generate a group signature, the member firstly randomizes his member certificate to get  $(s', a', b', c')$ , then encrypts  $e(g^k, g)$  using Cramer-Shoup's encryption scheme into  $(c_1, c_2, c_3, c_4)$ , and computes the following proof of signature:

$$\Sigma = SK\{(k, s, u) : e(a', \tilde{X})e(b', \tilde{X})^k = e(c', \tilde{g})^s, c_1 = \mathbf{g}^u, c_2 = \mathbf{h}^u, c_3 = y_1^u \mathbf{g}^k, c_4 = (y_2 y_3^{H(c_1, c_2, c_3)})^u\}(m),$$

where  $u$  is a random value chosen by the encryption scheme, and  $\mathbf{g}, \mathbf{h}, y_1, y_2, y_3$  are public key of the encryption scheme.

A group signature consists of  $(a', b', c', c_1, c_2, c_3, c_4, \Sigma)$  and is valid if  $\Sigma$  is valid and if  $e(a', Y) = e(b, \tilde{g})$  holds.

**Forgery of Camenisch-Lysyabskaya's Scheme** The flaw of Camenisch-Lysyabskaya's scheme is that any one can generate a valid signature on 0 given the public key of Scheme A:  $a = g^r, b = Y^r, c = X^r$ , where  $r$  is randomly chosen from  $Z_p^*$ . So is the case of Scheme B, C, D in [6].

To obtain a forgery of the group signature, the adversary computes  $a = g^{rs}, b = Y^{rs}, c = X^r$ , where  $r, s$  are randomly chosen from  $Z_p^*$ , sets  $(s, a, b, c)$  as his member certificate; to generate a group signature, the adversary does the subsequent procedure as exactly as a valid group member. It is easy to see that the forgery is successful because verifiers can not distinguish a proof of knowledge of a zero value from any non-zero values. A similar forgery exists for the anonymous credential scheme in [6].

## D.2 Review of Zhou-Lin's Schemes

**The VLR Group Signature Scheme** The Verifier-Local Revocation group signature in [19] is based on the following basic signature scheme.

**Scheme 1** Let  $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$  and bilinear map  $e$  be as described above.

- *Gen.* Select  $(x, y) \xleftarrow{\$} Z_p^* \times Z_p^*, x \neq y$ , set  $X = g^x, Y = g^y, \tilde{X} = \tilde{g}^x, \tilde{Y} = \tilde{g}^y$ . The secret key is  $(x, y)$ , public key is  $(X, Y, \tilde{X}, \tilde{Y}, g, \tilde{g}, e, p)$ .
- *Sig.* Given a message  $m \in Z_p^*$ , its signature is  $(U, V)$ , where  $U = g^r, V = g^{r(x+my)+xy}, r \xleftarrow{\$} Z_p^* \setminus \{1\}$ .
- *Ver.* Given a signature  $(U, V)$  of  $m$ , check if  $e(V, \tilde{g}) = e(U, \tilde{X}\tilde{Y}^m)e(X, \tilde{Y})$ . If the equation holds, then accept  $(U, V)$  as a valid signature of  $m$ , otherwise reject it as invalid.

- *Rnd.* Given a signature  $(U, V)$  of  $m$ ,  $U' = Ug^{r'}$ ,  $V' = V(XY^m)^{r'}$ , where  $r' \xleftarrow{\$} Z_p^*$ .

The VLR group signature is : a group member chooses his secret key  $k$ , commits it (without information theoretic hiding) to  $C = Y^k$ . The group manager, as the signer of Scheme 1, signs blindly on  $k$ , i.e., outputs  $(U, V)$  as the member certificate. When the group member is asked to produce a group signature of a message, he firstly randomizes his member certificate to get  $(U', V')$ , then generates a proof of signature of  $k$ :

$$\Sigma = SK\{k : e(V', \tilde{g}) = e(U', \tilde{X}\tilde{Y}^k)e(X, \tilde{Y})\}(m).$$

**The Full Group Signature Scheme** One of the full group signature schemes in [29] is based on the following signature that is a generalization of Scheme 1.

**Scheme 2** Let  $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$  and bilinear map  $e$  be as described above.

- *Gen.* Select  $(x, y) \xleftarrow{\$} Z_p^* \times Z_p^*$ , set  $X = g^x$ ,  $Y = g^y$ ,  $\tilde{X} = \tilde{g}^x$ ,  $\tilde{Y} = \tilde{g}^y$ . The secret key is  $sk = (x, y)$ , public key is  $pk = (X, Y, \tilde{X}, \tilde{Y}, g, \tilde{g}, e, p)$ .
- *Sig.* Given a message  $m \in Z_p^*$ , its signature is  $\sigma = (a, b) = (g^r, g^{r(x+my)+sx+xy})$ , where  $(r, s) \xleftarrow{\$} Z_p^* \times Z_p^*$ .
- *Ver.* Given a signature  $\sigma = (s, a, b)$  of  $m$ , check if  $e(b, \tilde{g}) = e(a, \tilde{X}\tilde{Y}^m)e(X, \tilde{Y})e(X, \tilde{g})^s$ . If the equation holds, then accept  $\sigma$  as a valid signature of  $m$ , otherwise reject it as invalid.
- *Rnd.* On input  $pk$ , message  $m$ , and a signature  $\sigma = (s, a, b)$ , choose random  $r_1, r_2 \in Z_p \times Z_p$ , output  $(s', a', b') = (s + r_1, ag^{r_2}, b(XY^m)^{r_2}X^{r_1})$ .

The group signature is as follows: a group member chooses a member secret key  $k$ , sends  $g^k$  to the group manager; the group manager signs on  $k$  as above to get  $(s, a, b)$ , i.e., the member certificate; to generate a group signature, the member firstly randomizes his member certificate to get  $(s', a', b')$ , then encrypts  $g^k$  using an appropriate encryption scheme into  $C$ , and computes the following proof of signature:

$$\Sigma = SK\{(k, s, \dots) : e(b', \tilde{g}) = e(a', \tilde{X})e(a', \tilde{Y})^k e(X, \tilde{Y})^s, \dots\}(m),$$

where “...” is to be determined by the specific encryption scheme. A group signature consists of  $(a', b', C, \Sigma)$  and is valid if  $\Sigma$  is valid.

**Forgeries** As the flaw in Camenisch-Lysyabskaya’s scheme, any one can generate a valid signature on 0 given the public key of Scheme 1:  $a = g^r Y^{-1}$ ,  $b = X^r$ , where  $r$  is randomly chosen from  $Z_p^*$ .

Similarly any one can generate a valid signature on 0 given the public key of Scheme 2:  $a = g^r Y^{-1}$ ,  $b = X^{r+s}$ , where  $r, s$  are randomly chosen from  $Z_p^*$ .

### D.3 Fixes

The reason of the above forgeries is that for a standard signature as Camenisch-Lysyabskaya's Scheme and Scheme 1, 2, the message space is limited to  $Z_p^*$ , so the resulting group signature should include a proof of knowledge of  $k \neq 0 \pmod p$  but were neglected in the original proposals.

So a natural method to fix the above flaws is modifying the concerned proofs of signature so that  $k \neq 0 \pmod p$  is included. But that will make the implementation of the proof of signature less efficient for no such efficient proof techniques have been provided.

The method adopted in [9] is modifying the Scheme A in [6] into a new signature scheme  $CL^+$  so that a forgery of certificate on 0 could be easily detected: a signature on  $m$  has the form of  $(a, a^y, a^{x+xy^m}, a^m, a^{my})$ , and the underlying groups are limited to SXDDH groups, i.e., DDH should be hard on both groups  $\mathbb{G}_1$  and  $\mathbb{G}_2$ . Thus the forgery of a group signature from valid underlying signature on 0 can be found by checking if the fourth element is 1 (in  $Z_p$ ). But this method is not sufficient for a full group signature scheme with strong anonymity.

A better method would be enhancing the message space of the underlying signature scheme into the whole space  $Z_p$ . Inspired by the scheme in [30], the flawed group signatures can be fixed by the following method: add a public key  $T = e(z, \tilde{g})$ , a secret key  $z$ ; the member certificate becomes  $(\hat{U}, \hat{V}) = (U, Vz)$  in the VLR scheme (Section D.2) and  $(\hat{s}, \hat{a}, \hat{b}) = (s, a, bz)$  in the full scheme (Section D.2); the corresponding proof of signature becomes

$$SK\{k : e(\hat{V}, \tilde{g}) = Te(\hat{U}, \tilde{X}\tilde{Y}^k)e(X, \tilde{Y})\}(m)$$

and

$$SK\{(k, s, \dots) : e(\hat{b}, \tilde{g}) = Te(\hat{a}, \tilde{X})e(\hat{a}, \tilde{Y})^k e(X, \tilde{Y})^s, \dots\}(m).$$

As for the schemes in [6], the randomization algorithm Rnd should also be modified:

- *Sig.*  $d \xleftarrow{\$} \mathbb{G}_1, s \xleftarrow{\$} Z_p^*, \sigma = (s, d^s, d^{sy}, d^{x+mxysz})$ .
- *Ver.* Given  $m, \sigma = (s, a, b, c)$ , check if  $e(a, Y) = e(b, \tilde{g})$  and  $e(a, X)e(b, X)^m T = e(c, \tilde{g})^s$ .
- *Rnd.* Given  $m, \sigma = (s, a, b, c)$ ,  $r_1, r_2 \xleftarrow{\$} Z_p, \sigma' = (s', a', b', c') = (r_2s, a^{r_2}g^{sr_1r_2}, b^{r_2}Y^{sr_1r_2}, cX^{r_1})$ .

Alternatively the VLR scheme (Section D.2) and the full scheme (Section D.2) can be fixed by directly substituting  $z$  for  $xy$  in the underlying signatures, as described in the following taking the full scheme as an example:

- *Gen.* Select  $(x, y, z) \xleftarrow{\$} Z_p^{*3}$ , set  $X = g^x, Y = g^y, \tilde{X} = \tilde{g}^x, \tilde{Y} = \tilde{g}^y$ . The secret key is  $sk = (x, y, z)$ , public key is  $pk = (X, Y, \tilde{X}, \tilde{Y}, g, \tilde{g}, e, p, T)$ , where  $T = e(g^z, \tilde{g})$ .

- *Sig.* Given a message  $m \in Z_p^*$ , its signature is  $\sigma = (a, b) = (g^r, g^{r(x+my)+sx+z})$ , where  $(r, s) \xleftarrow{\$} Z_p^* \times Z_p$ .
- *Ver.* Given a signature  $\sigma = (s, a, b)$  of  $m$ , check if  $e(b, \tilde{g}) = e(a, \tilde{X}\tilde{Y}^m)e(X, \tilde{g})^sT$ . If the equation holds, then accept  $\sigma$  as a valid signature of  $m$ , otherwise reject it as invalid.
- *Rnd.* On input  $pk$ , message  $m$ , and a signature  $\sigma = (s, a, b)$ , choose random  $r_1, r_2 \in Z_p \times Z_p$ , output  $(s', a', b') = (s + r_1, ag^{r_2}, b(XY^m)^{r_2}X^{r_1})$ .