# Weakly only Unforgeable Signature and Its Application in Group Signature

Sujing Zhou, Dongdai Lin
SKLOIS Lab, Institute of Software,
Chinese Academy of Sciences, 100080, Beijing , P.R. China.
Email: {zhousujing,ddlin}@is.iscas.ac.cn

**Abstract**

If a signature scheme is secure in the sense that no forgery on any new message (i.e., a message that has never been signed) is available for any computation restricted adversary, it is said weakly unforgeable (wUF), in contrast to strongly unforgeable (sUF) meaning no new signature on any old message (i.e., a valid signature on the message is already known) is available to such adversaries. sUF signatures are generally considered advantageous over wUF ones because of preference for high level security. But the case may be different when they are employed to construct group signatures.

wUF but not sUF signatures, called WoUF signatures in this paper, are investigated in this paper. It is found that by applying a generic construction to WoUF signatures with indirectly-signability and perfectly-unlinkability (also defined in this paper), we can regenerate many efficient group signatures in literature.

We also propose improvements to the group signature schemes of CL04, NSN04, KY05, in line with our generic construction.

**Keywords:** Digital Signature, Group Signature, Weakly Unforgeable Signature, Strongly Unforgeable Signature.

## 1 Introduction

Suppose you are a digital signature user and have signed on many documents, you have sufficient reason to worry that someone else might be able to forge a signature of yours on a new document you have never seen although you have kept the secret signing key absolutely safe. What you worry is actually whether the signature is *existentially unforgeable against chosen message attacks* (eUF) [GMR88].

You may be less bothered by someone else deriving a new signature on an old document you have already signed, i.e., the signature is not *strongly unforgeable against chosen message attacks* (sUF) [ADR02]. But in some cases, strong unforgeability of a signature is important, e.g., when the signature is used to construct a chosen cipher-text secure cryptosystem where each cipher-text is appended with a signature on it [BSW06].

If a signature is sUF, then it is also eUF, but not vice versa. So existentially unforgeable signatures are also called weakly unforgeable (wUF) ones in contrast to higher security of sUF. A sUF signature has been considered advantageous over a wUF signature. If a signature is proved wUF, it is a satisfactory result; and if it is also sUF, then it is a bonus. No one has ever thought that wUF might be advantageous over sUF in some cases.

We will investigate a kind of signatures, called weakly only unforgeable (WoUF) signatures, that are wUF but not sUF, and there exist efficient algorithms to derive new signatures from a given signature. We show that WoUF signatures may be especially helpful in constructing group signatures, a primitive that has been found useful in various applications, e.g., anonymous authentication, internet voting, electronic bidding, trusted computing.

**Group signature.** The proposal of group signatures [CvH91] are motivated by enabling members of a group to sign on behalf of the group without leaking their true identities; but the signer's identity is able to be opened, i.e., discovered by the group manager (GM) on disputes. The counterpart of a group signature in the real world is official seal, at the sight of which, anyone can be assured that it is made by some person from the claimed authority, but have no idea of who that person is.

In brief, a group signature scheme is composed of the following steps: (1) GM, the group manager, 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., without knowing the secret key value, the signature is also called *member certificate*. (3) Any group member can generate group signatures using his member secret key and member certificate, called *group signing key* all together.

**Generic Constructions of Group Signature.** Any wUF signature can be used to construct secure group signature [Pet97][BSZ05], but their generic constructions are not much efficient.

We will investigate the generic construction based on WoUF signatures following the line of [BSZ05], aiming to get efficient schemes. What behind our idea is a linkage between randomization of WoUF signatures and anonymity of group signatures. Essentially a group signature can be viewed as a proof of knowledge of a signature signed by GM. If the signature is WoUF, then each member can derive a new signature, i.e., a new member certificate, on his member secret. Because the new member certificate seems random, it is safe to be published and be part of the generated group signature instead of being concealed and proved the correctness of concealment as in [BSZ05].

In fact, ad hoc examples of group signatures consisting with the above idea have been proposed [CL04, BBS04], but the generic method behind their observations has not been investigated and paid enough attention.

**Our Contribution.** We have given explicit definition of WoUF signatures, generalized and formalized the method of constructing a group signature from a randomizable signature, and pointed out what kind of such signatures can be adopted in this application: WoUF signature with indirectly-signability and perfectly-unlinkability, the latter feature is firstly defined and identified in this paper.

Our generic construction resulted in group signatures with shorter signature length or simpler signature generation. We propose improvements to the group signature schemes of [CL04], [NSN04], [KY05].

**Organization.** The definition of WoUF is given in Section 3, and some examples of WoUF are also demonstrated in this section, those marked with "+" mean they are variants of the original schemes to fit into our definition.

The generic construction of group signature based on a kind of WoUF signatures is shown in Section 4.1, with detailed guideline of proofs in Appendix C. The security model of group signature which our generic construction is based on is clarified in Appendix B. The examples of our generic construction are available in Section 4.3.

## 2 Preliminary

### 2.1 Notation

$x \overset{\$}{\leftarrow} X$ denotes $x$ is chosen uniformly at random from the set $X$. $x \overset{\$}{\leftarrow} A(.,.,.)$ denotes $x$ is generated from executing algorithm $A$ where random variables are chosen uniformly at random. $G^k$, $(Z_p^*)^k$ denote a $k$ tuple from $G$ and $Z_p^*$ respectively. $0^k$ $(1^k)$ denotes the string of $k$ zeros (ones). $|M|$ denotes the binary length of $M$. 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 $\pi$.

$\varepsilon(k)$ is a *negligible function* if $\varepsilon(k) \leq 1/P(k)$ for any polynomial $P(k)$ and all sufficiently large $k$.

### 2.2 Definitions

**Definition 1** (wUF-ATK[DK01]). *A signature scheme DS is wUF-ATK secure (ATK $\in$ {CMA, ACMA}), i.e., weakly unforgeable against ATK attack, if for every probabilistic polynomial-time oracle machine $\mathcal{A}$, it* *holds that*

$$\mathrm{Adv}_{DS,\mathcal{A}}^{wUF-ATK} = \Pr[Ver(pk,m,\sigma)=1, m \neq m_i, (m,\sigma)\leftarrow\mathcal{A}(m_i,\sigma_i,pk),$$
$$(pk,sk)\leftarrow Gen(k), (m_i,\sigma_i)\leftarrow Q_{\mathcal{A}}^{Sig(sk,.)}, i=1,...,q_{sig}] < \varepsilon(k)$$

*where $\varepsilon(k)$ is a negligible function, the probability is taken over the coin tosses of algorithms Gen, Sig and $\mathcal{A}$. $Q_{\mathcal{A}}^{Sig(sk,.)}$ denotes the finite set of queries to oracle $Sig(sk,.)$ made by $\mathcal{A}$.*

**Definition 2** (wUF). *A signature scheme is wUF, if it is wUF-CMA secure or wUF-ACMA secure.*

## 3 Weakly only Unforgeable (WoUF) Signature

The formal definitions of wUF and sUF are available in many textbooks, e.g., [DK01].

**Definition 3** (WoUF Signature). *A WoUF signature scheme is a wUF digital signature scheme that has a 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 $\sigma=(\Upsilon,\Sigma)\in S$, where $\Upsilon$ is random chosen independent from $m$, $\Sigma$ is calculated from $\Upsilon$ and $m$.*
- *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 random input and a signature $(\Upsilon,\Sigma)$ on $m$, output a $(\Upsilon',\Sigma')\neq(\Upsilon,\Sigma)$ that is also a signature on $m$.*

The following concept of *indirectly signable* is a restatement of signatures on committed message [CL04].

**Definition 4** (Indirectly Signable). *A signature is indirectly signable if there exists a one way function $f$ (as defined in Chapter 9.2.4, [MvOV96] or more technically as in Chapter 2.2, [Gol01]) and an efficient algorithm $Sig_f$ that $Sig(sk,m)=Sig_f(sk,f(m))$.*

**Definition 5** (Perfectly Unlinkable). *A WoUF signature wDS=(Gen,Sig,Ver,Rnd) is perfectly unlinkable if for any two message signature pairs $(m_i,\Upsilon_i,\Sigma_i)_{i=0,1}$, and a given $\Sigma'$ that $(\Upsilon',\Sigma')\overset{\$}{\leftarrow}Rnd(m_\phi,\Upsilon_\phi,\Sigma_\phi)$, $\phi\overset{\$}{\leftarrow}\{0,1\}$, there exists a random value for each $i$ that $\Sigma'$ is part of a randomization of $(m_i,\Upsilon_i,\Sigma_i)$, i.e., it is not possible to distinguish $\phi$ if only $\Sigma'$ is revealed.*

Note that this feature is important to generate a secure group signature as shown in proof of Lemma 4.1.

Not all WoUF signatures admits such a characteristic. For example, Scheme A in [CL04] has a natural randomization algorithm: $(a,b,c)\rightarrow(a^r,b^r,c^r)$ for any random $r$. But this randomization does not satisfy perfectly unlinkability, and adopting such randomization will result in insecure group signature. That may explain the reason why $(a,b,c)\rightarrow(a^r,b^r,c^{r'})$ is adopted in the generated group signature in [CL04]. But the feature of perfectly unlinkability is not identified there.

In the sequel, we demonstrate some examples of WoUF signatures with *indirectly-signability* and *perfectly-unlinkability*, where CL02 and CL04+ are restatements of [CL02] and [CL04] in line with our definitions. The examples are summarized in Table 1.

### 3.1 CL02

A CL02 signature on $m$ with length $l_m$ is $(e,s,v)$ that $v^e=a^m b^s c \bmod n$, where the lengths of $e,s$ are $l_e,l_s$ respectively and $e$ is a prime number, $v\in Z_n$. (See [CL02] for the notations.) It admits the following randomization algorithm:

| Scheme | Signature | Randomization | OWF | Security |
|---|---|---|---|---|
| CL02 | $\Upsilon = (e,s)$,$\Sigma = (v)$ | $\Upsilon' = (e, s + re)$,$\Sigma' = (vb^r)$ | $f(m) = a^m$ | ACMA, PU |
| CL04+ | $\Upsilon = (s)$,$\Sigma = (a,b,c)$ | $\Upsilon' = (sr_2)$,$\Sigma' = (a^{r_1 r_2}, b^{r_1 r_2}, c^{r_1})$ | $f(m) = g_1^m$ | ACMA, PU |
| Wat05+ | $\Upsilon = (s)$,$\Sigma = (a,b)$ | $\Upsilon' = (sr_1)$,$\Sigma' = (a^{r_1} g^{r_2}, b^{r_1}(u' \prod_{m_i=1} u_i)^{r_2})$ | — | ACMA, PU |
| BB04+ | $\Upsilon = (s,t)$,$\Sigma = (A)$ | $\Upsilon' = (s, rt)$,$\Sigma' = (A^r)$ | — | ACMA, PU |
| BBS04+ | $\Upsilon = (s,t)$,$\Sigma = (A)$ | $\Upsilon' = (s, rt)$,$\Sigma' = (A^r)$ | $f(m) = h_1^m$ | CMA, PU |
| NSN04* | $\Upsilon = (a,b,c)$,$\Sigma = (A)$ | $\Upsilon' = (a, b + ra, c + r)$,$\Sigma' = (A + rP_{pub})$ | $f(m) = mP$ | ACMA, PU |
| CL04* | $\Upsilon = (s)$,$\Sigma = (a,b)$ | $\Upsilon' = (sr_1)$,$\Sigma' = (a^{r_1} g^{r_2}, b^{r_1}(XY^m)^{r_2})$ | $f(m) = g^m$ | ACMA, PU |

Table 1: A Summary Table of Examples of WoUF Signatures (Note: PU denotes perfectly unlinkability).

- *Rnd.* On input $pk = (n, a, b, c)$, message $m$, and a signature $(\Upsilon, \Sigma)$, where $\Upsilon = (e, s)$, $\Sigma = (v)$, choose random $r$ with length $l_r = l_s - l_e - 1$, set $\Upsilon' = (e, s + re)$, $\Sigma' = (vb^r)$.

CL02 is ACMA secure (proved in standard model [CL02]); it is indirectly signable if we define $f(m) = a^m$; obviously, it is also perfectly unlinkable because each randomized $\Sigma'$ only consists of one element that is generated independently and randomly each time.

## 3.2 CL04+

The concept of WoUF has been adopted when signatures are applied in anonymous credential system, group signature, e.g., Scheme A in [CL04] combining its randomization in forming a anonymous credential or group signature scheme is actually WoUF, the following is a restatement of it according to definition 3 with an extra algorithm *Rnd*, we call it CL04+.

| | |
|---|---|
| **CL04+.** | Let $g$ be a $p$ order cyclic group that exists a bilinear map $e : G_1 \times G_2 \to G_3$. $G_1 = \langle g \rangle$, $G_2 = \langle \tilde{g} \rangle$. |
| *Gen.* | It chooses $x \overset{\$}{\leftarrow} Z_p^*$ and $y \overset{\$}{\leftarrow} Z_p^*$, and sets $sk = (x, y)$, $pk = (p, g, \tilde{g}, G_1, G_2, e, X, Y)$, where $X = \tilde{g}^x$ and $Y = \tilde{g}^y$. |
| *Sig.* | On input message $m$, secret key $sk = (x, y)$, and public key $pk$, choose a random $d \in G_1$, and a random $s \in Z_p^*$, and output the signature $(\Upsilon, \Sigma)$ where $\Upsilon = (s)$, $\Sigma = (d^s, d^{sy}, d^{x+mxy})$. Note that $(a^{1/s}, b^{1/s}, c)$ is a signature of $m$ according to Scheme A in [CL04]. |
| *Ver.* | On input $pk$, message $m$, and purported signature $(\Upsilon, \Sigma) = (s, a, b, c)$, check that the following verification equations hold: $e(a, Y) = e(b, \tilde{g})$ and $e(a, X)e(b, X)^m = e(c, \tilde{g})^s$. |
| *Rnd.* | On input $pk$, message $m$, and a signature $(\Upsilon, \Sigma) = (s, a, b, c)$, choose random $r_1 \in Z_p^*$ and $r_2 \in Z_p^*$, output $(\Upsilon', \Sigma')$ where $\Upsilon' = (s') = (r_2 s)$, $\Sigma' = (a', b', c') = (a^{r_1 r_2}, b^{r_1 r_2}, c^{r_1})$. |

CL04+ is wUF-ACMA which can be proved similarly as Scheme A in [CL04]. Briefly, suppose $\mathcal{A}$ is an adversary of CL04+, then an adversary $\mathcal{B}$ of LRSW assumption (defined in [CL04]) is available: when $\mathcal{A}$ queries signature on $m$, $\mathcal{B}$ transfers the query to LRSW oracle $O_{x,y}$ (defined in [CL04]); $\mathcal{B}$ will get a response from LRSW oracle $O_{x,y}$, i.e., $(a, b, c)$=$(a, a^y, a^{x+mxy})$, then $\mathcal{B}$ sends $(s, a^s, b^s, c)$ $(s \overset{\$}{\leftarrow} Z_p^*)$ to $\mathcal{A}$. If $\mathcal{A}$ outputs a signature $(s^*, a^*, b^*, c^*)$ on a message $m^*$ that it has never queried, then $(m^*, {a^*}^{\frac{1}{s^*}}, {b^*}^{\frac{1}{s^*}}, c^*)$ is a resolution to LRSW assumption, $\mathcal{B}$ wins.

CL04+ is indirectly signable if define $f(m) = g^m$. It is perfectly unlinkable because of the following reason. If $(m_i, s_i, a_i, b_i, c_i)_{i=0,1}$ are two valid message signature pairs, suppose $\Sigma' = (a', b', c')$ is from a randomization of $(m_\phi, s_\phi, a_\phi, b_\phi, c_\phi)_{\phi \in \{0,1\}}$, i,e., $a' = a_\phi^{r_1 r_2}$, $b' = b_\phi^{r_1 r_2}$, $c' = c_\phi^{r_1}$, then for $i = \phi$, the random value is $(r_1, r_2)$ obviously, for $i \neq \phi$, there exists random value $(\tilde{r}_1, \tilde{r}_2)$ resolvable from $r_1 r_2 = \alpha \tilde{r}_1 \tilde{r}_2$ and $r_1 s_\phi^{-1}(x + m_\phi xy) = \alpha \tilde{r}_1 s_i^{-1}(x + m_i xy)$, where $\alpha = \log_{a_i} a_\phi$.

## 3.3 Wat05+

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

**Wat05+.** Let $G$, $G'$ be two $p$ order cyclic groups, and there exists a bilinear map $e : G \times G \to G'$. $G = \langle g \rangle$.

*Gen.* Set secret key $sk = (x)$, $pk = (e, g_1, g_2, u', u_i, i = 0, .., l)$, where $g_1, g_2, u', u_i$ are all elements from $G$, $g_1 = g^x$, $l$ is the maximum binary length of a message to be signed.

*Sig.* Given a message $m$ with length at most $l$, the signature $(\Upsilon, \Sigma)$ is $\Upsilon = (s)$, $\Sigma = (a, b) = (g^r, g_2^{sx}(u' \prod_{m_i=1} u_i)^r)$, where $s \overset{\$}{\leftarrow} Z_p^*$. Note that $(a^{1/s}, b^{1/s})$ is a signature of $m$ according to Scheme [Wat05].

*Ver.* Given a message $m$ and its signature $(\Upsilon, \Sigma) = (s, a, b)$, it is a valid signature on $m$ if $e(b, g) = e(u' \prod_{m_i=1} u_i, a) e(g_2, g_1)^s$.

*Rnd.* On input $pk$, message $m$, and a signature $(\Upsilon, \Sigma)$, where $\Upsilon = (s)$, $\Sigma = (a, b)$, choose $(r_1, r_2) \overset{\$}{\leftarrow} Z_p^* \times Z_p^*$, set $\Upsilon' = (s') = (sr_1)$, $\Sigma' = (a', b') = (a^{r_1} g^{r_2}, b^{r_1}(u' \prod_{m_i=1} u_i)^{r_2})$. The new randomized signature on $m$ is $(\Upsilon', \Sigma')$.

Wat05+ is wUF-ACMA. Briefly, Suppose $\mathcal{A}$ is an adversary of Wat05+, then an adversary $\mathcal{B}$ of Wat05 is available: when $\mathcal{A}$ queries signature on $m$, $\mathcal{B}$ transfers the query to signature oracle of Wat05 obtaining $(a, b) = (g^r, g_2^x(u' \prod_{m_i=1} u_i)^r)$, which is then modified by $\mathcal{B}$ into $(s, a^s, b^s)$ where $s \overset{\$}{\leftarrow} Z_p^*$; the modification, now a valid Wat05+ signature on $m$, is sent to $\mathcal{A}$. If $\mathcal{A}$ outputs a signature $(s^*, a^*, b^*)$ on a message $m^*$ that it has never queried, then $(a^{*\frac{1}{s^*}}, b^{*\frac{1}{s^*}})$ is a valid Wat05 signature on $m^*$ that $\mathcal{B}$ has not queried.

Wat05+ is not indirectly signable because $m$ must be known to calculate the signature.

Wat05+ is perfectly unlinkable: if $(m_i, s_i, a_i, b_i)_{i=0,1}$ are two valid message signature pairs, suppose $\Sigma' = (a', b')$ is from a randomization of $(m_\phi, s_\phi, a_\phi, b_\phi)_{\phi \in \{0,1\}}$, i,e., $a' = a_\phi^{r_1} g^{r_2}$, $b' = b_\phi^{r_1}(u' \prod_{m_{\phi_j}=1} u_j)^{r_2}$, then for $i = \phi$, the random value is $(r_1, r_2)$ obviously, for $i \neq \phi$, there exists random value $(\tilde{r}_1, \tilde{r}_2)$ resolvable from $\alpha \tilde{r}_1 + \tilde{r}_2 = \beta r_1 + r_2$ and $\tilde{r}_1 s_i x - r_1 s_\phi x = \gamma(\beta r_1 + r_2)$, where $\alpha = \log_g a_i$, $\beta = \log_g a_\phi$, $\gamma = \log_g (\prod_{m_{\phi_j} \neq m_{i_j}} u_j^{m_{\phi_j} - m_{i_j}})$.

## 3.4 BB04+

Some proposed sUF signatures can be converted into WoUF signature, e.g., the following BB04+, which contributes a lot in improving a group signature with concurrent joining (Section 4.8.1), is such a transformation from the full scheme of [BB04].

**BB04+.** Let $G_1$, $G_2$ be two $p$ order cyclic groups, and there exists a bilinear map $e : G_1 \times G_2 \to G_3$. $G_1 = \langle g \rangle$, $G_2 = \langle \tilde{g} \rangle$.

*Gen.* It chooses $x \overset{\$}{\leftarrow} Z_p^*$, $y \overset{\$}{\leftarrow} Z_p^*$, and sets $sk = (x, y)$, $pk = (p, G_1, G_2, g, \tilde{g}, X, Y, e)$, where $X = \tilde{g}^x$, $Y = \tilde{g}^y$.

*Sig.* On input message $m$, secret key $sk$, and public key $pk$, choose $(s, t) \overset{\$}{\leftarrow} Z_p^{*2}$, compute $A = g^{\frac{t}{x+m+ys}}$, output the signature $(\Upsilon, \Sigma)$ where $\Upsilon = (s, t)$, $\Sigma = (A)$. Note that $(s, A^{\frac{1}{t}})$ is a valid [BB04] signature on $m$.

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

*Rnd.* On input $pk$, message $m$, and a signature $(\Upsilon, \Sigma) = (s, t, A)$, choose $r \overset{\$}{\leftarrow} Z_p^*$, output $(\Upsilon', \Sigma')$ where $\Upsilon' = (s', t') = (s, rt)$, $\Sigma' = (A') = (A^r)$.

BB04+ can be proved wUF-ACMA similarly to the scheme [BB04]. Briefly, suppose $\mathcal{A}$ is an adversary of BB04+, then an adversary $\mathcal{B}$ of [BB04] is available: when $\mathcal{A}$ queries signature on $m$, $\mathcal{B}$ transfers the query to signature oracle of [BB04]; $\mathcal{B}$ will get a response from the signature oracle, i.e., $(s, A)$, where

$e(A, XY^s\tilde{g}^m) = e(g, \tilde{g})$, then $\mathcal{B}$ chooses $t \overset{\$}{\leftarrow} Z_p^*$, sends $(s, t, A^t)$ to $\mathcal{A}$. If $\mathcal{A}$ outputs a signature $(s^*, t^*, A^*)$ on a message $m^*$ that it has never queried, then $(s^*, A^{*\frac{1}{t^*}})$ is a valid BB04+ signature on $m^*$, which $\mathcal{B}$ has never queried.

Obviously, BB04+ is perfectly unlinkable because each randomized $\Sigma'$ 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.

## 3.5 BBS04+

To compensate the drawback of non-directly signability of BB04+, the following BBS04+ is an indirectly signable WoUF signature based on [BBS04], at the cost of weaker security.

**BBS04+.** Let $G_1$, $G_2$ be two $p$ order cyclic groups, and there exists a bilinear map $e : G_1 \times G_2 \to G_3$. $G_1 = \langle g \rangle$, $G_2 = \langle \tilde{g} \rangle$.

*Gen.* It chooses $x \overset{\$}{\leftarrow} Z_p^*$, and sets $sk = (x)$, $pk = (p, G_1, G_2, g, \tilde{g}, h_1, e, w)$, where $w = \tilde{g}^x$, $h_1 \overset{\$}{\leftarrow} G_1$.

*Sig.* On input message $m$, secret key $sk = (x)$, and public key $pk$, choose $s \overset{\$}{\leftarrow} Z_p^*$ and $t \overset{\$}{\leftarrow} Z_p^*$, compute $A = (h_1^m g)^{\frac{t}{x+s}}$, output the signature $(\Upsilon, \Sigma)$ where $\Upsilon = (s, t)$, $\Sigma = (A)$. Note that $(s, A^{\frac{1}{t}})$ is a valid [BBS04] signature on $m$.

*Ver.* On input $pk$, message $m$, and purported signature $(\Upsilon, \Sigma) = (s, t, A)$, check that $e(A, w\tilde{g}^s) = e(h_1^m g, \tilde{g}^t)$.

*Rnd.* On input $pk$, message $m$, and a signature $(\Upsilon, \Sigma) = (s, t, A)$, choose $r \overset{\$}{\leftarrow} Z_p^*$, output $(\Upsilon', \Sigma')$ where $\Upsilon' = (s, rt)$, $\Sigma' = (A^r)$.

BBS04+ can be proved wUF-CMA similarly to the scheme [BBS04]. Briefly, suppose $\mathcal{A}$ is an adversary of BBS04+, then an adversary $\mathcal{B}$ of [BBS04] is available: when $\mathcal{A}$ queries signature on $m$, $\mathcal{B}$ transfers the query to signature oracle of [BBS04]; $\mathcal{B}$ will get a response from the signature oracle, i.e., $(s, A)$, where $e(A, w\tilde{g}^s) = e(h_1^m g, \tilde{g})$, then $\mathcal{B}$ chooses $t \overset{\$}{\leftarrow} Z_p^*$, sends $(s, t, A^t)$ to $\mathcal{A}$. If $\mathcal{A}$ outputs a signature $(s^*, t^*, A^*)$ on a message $m^*$ that it has never queried, then $(s^*, A^{*\frac{1}{t^*}})$ is a valid BBS04+ signature on $m^*$, which $\mathcal{B}$ has never queried.

BBS04+ is indirectly signable if we define $f(m) = h_1^m$. Obviously, BBS04+ is perfectly unlinkable because each randomized $\Sigma'$ only consists of one element that is generated independently and randomly each time.

## 3.6 NSN04*

NSN04* is a new WoUF signatures based on [NSN04]. As we mentioned before, BB04+ is ACMA secure but not indirectly signable, while BBS04+ is indirectly signable but only CMA secure, among the signatures that their security can be reduced to $q$-SDH assumption. NSN04* has all the features.

**NSN04*.** Let $G$ be a $p$ order additive cyclic group (to consistent with [NSN04]), and there exists a bilinear map $e : G \times G \to G'$. $G = \langle P \rangle$.

*Gen.* It chooses $\gamma \overset{\$}{\leftarrow} Z_p^*$, and sets $sk = (\gamma)$, $pk = (p, G, G', P, P_0, P_{pub}, e)$, where $P_{pub} = \gamma P$, $P_0 \overset{\$}{\leftarrow} G$.

*Sig.* On input message $m$, secret key $sk = (\gamma)$, and public key $pk$, choose $(a, b, c) \overset{\$}{\leftarrow} Z_p^{*3}$, compute $A = \frac{1}{\gamma+a}[mP + (b + \gamma c)P_{pub} + P_0]$, output the signature $(\Upsilon, \Sigma)$ where $\Upsilon = (a, b, c)$, $\Sigma = (A)$. Note that $(a, A)$ is a valid [NSN04] signature on $m$ if $b + \gamma c = 0 \mod p$.

*Ver.* On input $pk$, message $m$, and purported signature $(\Upsilon, \Sigma) = (a, b, c, A)$, check that $e(A, P_{pub} + aP) = e(mP + bP_{pub} + P_0, P)e(cP_{pub}, P_{pub})$.

*Rnd.* On input $pk$, message $m$, and a signature $(\Upsilon, \Sigma) = (a, b, c, A)$, choose $r \overset{\$}{\leftarrow} Z_p^*$, output $(\Upsilon', \Sigma')$ where $\Upsilon' = (a, b + ra, c + r)$, $\Sigma' = (A + rP_{pub})$. It can be checked that $e(A', P_{pub} + a'P) = e(mP + b'P_{pub} + P_0, P)e(c'P_{pub}, P_{pub})$, where $A' = A + rP_{pub}$, $a' = a$, $b' = b + ra$, $c' = c + r$.

The security if NSN04* is guaranteed by the following Lemma.

**Lemma 3.1.** *NSN04* is* wUF-ACMA *if q-SDH problem in G is hard, where q is polynomial in* $|p|$. *See Appendix A for the proof.*

NSN04* is indirectly signable if we define $f(m) = mP$. Obviously, NSN04* is perfectly unlinkable because each randomized $\Sigma'$ only consists of one element that is generated independently and randomly each time.

## 3.7 CL04*

CL04* is a new WoUF signature based on [CL04]. CL04* improves CL04+ by reducing the total signature size and keeping ACMA secure, perfectly unlinkable, indirectly signable.

**CL04*.** Let $G_1$ be a $p$ order cyclic group that exists a bilinear map $e : G_1 \times G_2 \to G_3$. $G_1 = \langle g \rangle$, $G_2 = \langle \tilde{g} \rangle$.

*Gen.* Select $(x, y) \xleftarrow{R} Z_p^* \times Z_p^*$, set $X = g^x$, $Y = g^y$, $\widetilde{X} = \tilde{g}^x$, $\widetilde{Y} = \tilde{g}^y$. The secret key is $sk = (x, y)$, public key is $pk = (X, Y, \widetilde{X}, \widetilde{Y}, g, \tilde{g}, e, p)$.

*Sig.* Given a message $m \in Z_p^*$, its signature is $(\Upsilon, \Sigma)$, where $\Upsilon = (s)$, $\Sigma = (a, b) = (g^r, g^{r(x+my)+sxy})$, $(r, s) \xleftarrow{\$} Z_p^* \times Z_p^*$.

*Ver.* Given a signature $(\Upsilon, \Sigma)$=$(s, a, b)$ of $m$, check if $e(b, \tilde{g}) = e(a, \widetilde{X}\widetilde{Y}^m)e(X, \widetilde{Y})^s$. If the equation holds, then accept $(\Upsilon, \Sigma)$ as a valid signature of $m$, otherwise reject it as invalid.

*Rnd.* On input $pk$, message $m$, and a signature $(\Upsilon, \Sigma) = (s, a, b)$, choose random $r_1, r_2 \in Z_p^* \times Z_p^*$, output $(\Upsilon', \Sigma')$ where $\Upsilon' = (s') = (r_1 s)$, $\Sigma' = (a', b') = (a^{r_1} g^{r_2}, b^{r_1}(XY^m)^{r_2})$.

The security of CL04* is based on the following assumption proposed in [ZL06b].

**Assumption 1.** *Suppose $G_1$, $G_2$ be two $p$ ordered cyclic group that exists a bilinear map $e : G_1 \times G_2 \to G_3$, $G_1 = \langle g \rangle$, $G_2 = \langle \tilde{g} \rangle$. Let $X = g^x$, $Y = g^y$, $\widetilde{X} = \tilde{g}^x$, $\widetilde{Y} = \tilde{g}^y$, $O_{x,y}(.)$ be an oracle that, on input a value $m \in Z_p^*$, outputs a pair $(g^r, g^{r(x+my)+xy})$ for a randomly chosen $r \in Z_p^* \setminus \{1\}$. Then for any probabilistic polynomial time bounded adversary $\mathcal{A}$, the following probability is negligible:*

$$\begin{aligned}\Pr[(p, G_1, G_2, G_3, e, g, \tilde{g}) \leftarrow Setup(1^k); x \xleftarrow{R} Z_p^*; y \xleftarrow{R} Z_p^*; X = g^x; Y = g^y; \widetilde{X} = \tilde{g}^x; \widetilde{Y} = \tilde{g}^y;\\ (m, a, b) \leftarrow \mathcal{A}^{O_{x,y}}(p, g, \tilde{g}, e, X, Y, \widetilde{X}, \widetilde{Y}) : m \in Z_p^* \backslash Q \;\wedge\\ a = g^r \;\wedge a \notin \{1_{G_1}, g\} \;\wedge\; b = g^{r(x+my)+xy}] < \epsilon,\end{aligned}$$

*where $Q$ is the set of queries that $\mathcal{A}$ has made to $O_{x,y}(.)$, $1_{G_1}$ is the unit element of $G_1$.*

CL04* is wUF-ACMA under Assumption 1. Briefly, Suppose $\mathcal{A}$ is an adversary of CL04*, then an adversary $\mathcal{B}$ of Assumption 1 is available: when $\mathcal{A}$ queries signature on $m$, $\mathcal{B}$ transfers the query to oracle $O_{x,y}$ obtaining $(a, b) = (g^r, g^{r(x+my)+xy})$, which is then modified by $\mathcal{B}$ into $(s, a^s, b^s)$ where $s \xleftarrow{\$} Z_p^*$; the modification, now a valid CL04* signature on $m$, is sent to $\mathcal{A}$. If $\mathcal{A}$ outputs a signature $(s^*, a^*, b^*)$ on a message $m^*$ that it has never queried, then $(a^{*\frac{1}{s^*}}, b^{*\frac{1}{s^*}})$ is a resolution to Assumption 1.

CL04* is indirectly signable if define $f(m) = g^m$. It is also perfectly unlinkable: if $(m_i, s_i, a_i, b_i)_{i=0,1}$ are two valid message signature pairs, suppose $\Sigma' = (a', b')$ is from a randomization of $(m_\phi, s_\phi, a_\phi, b_\phi)_{\phi \in \{0,1\}}$, i.e., $a' = a_\phi^{r_1} g^{r_2}$, $b' = b_\phi^{r_1}(XY^{m_\phi})^{r_2}$, then for $i = \phi$, the random value is $(r_1, r_2)$ obviously, for $i \neq \phi$, there exists random value $(\tilde{r}_1, \tilde{r}_2)$ resolvable from $\alpha\tilde{r}_1 + \tilde{r}_2 = \beta r_1 + r_2$ and $\tilde{r}_1 s_i x - r_1 s_\phi x = (\beta r_1 + r_2)(m_\phi - m_i)$, where $\alpha = \log_g a_i$, $\beta = \log_g a_\phi$.

# 4 Group Signature from WoUF Signature

**Definition 6** ([BSZ05]). 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 finishes 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;

– **Open:** a deterministic algorithm which, on input a message-signature pair $(m, \sigma)$, secret key $gsk$ of GM (OA), 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 a WoUF signature $DS = (K_s, Sig, Ver, Rnd)$ ($K_s = Gen$) which is indirectly signable with a one way function $f$, and perfectly unlinkable, a probabilistic public encryption $PE = (K_e, Enc, Dec)$.

Define the following relations:

$\rho$: $(x, w) \in \rho$ iff $x = f(w)$.
$\rho_1$: $((pk_e, pk_s, m, C, \Sigma), (w, \Upsilon, r)) \in \rho_1$ iff $Ver(pk_s, w, (\Upsilon, \Sigma)) = 1$ 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 [BSZ05].

$(P, V)$ 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) [Fis05]:

Completeness: For any random oracle $H$, any $(x, w) \in \rho$, and any $\pi \leftarrow P^H(x, w, R)$, it satisfies $\Pr[V^H(x, \pi, R) = 1] \geq 1 - \varepsilon_1(k)$, where $\varepsilon_1(k)$ is a negligible function.

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 $H$ to queries from $A$. Let $w \leftarrow K(x, \pi, Q_H(A))$, then as a function of $k$, $\Pr[(x, w) \notin \rho, V^H(x, \pi, R) = 1] < \varepsilon_2(k)$, where $\varepsilon_2(k)$ is a negligible function.

GS is constructed as follows, and the details are described in Table 2 and Table 3.

**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 $pk_i \leftarrow f(sk_i)$, 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 $(\Upsilon, \Sigma)$, firstly derive a new certification $(\Upsilon', \Sigma') = Rnd(gpk, sk_i, \Upsilon, \Sigma)$; encrypt $pk_i$ with $PE$: $C \leftarrow Enc(pk_e, pk_i, r_i)$, 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, transfer $\pi_1$

into a signature on $m$ using any method of transferring a non-interactive zero-knowledge proof into a signature [FS87, BG90, CD95, CL06], we also use $\pi_1$ to note the transferred signature for simplicity. The group signature on $m$ is $\sigma = (C, \Sigma', \pi_1)$.

**GVer.** On input $(gpk, m, \sigma)$, parse $\sigma$ as $(C, \Sigma', \pi_1)$, 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 $(C, \Sigma', \pi_1)$. 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 validity of the group signature and the output of Open.

| User $i$ | | Issue Authority |
|---|---|---|
| Select $sk_i$, $pk_i \leftarrow f(sk_i)$, | | |
| $\pi \leftarrow P(pk_i, sk_i, R)$ | $\xrightarrow{pk_i,\pi}$ | If $V(pk_i, \pi, R) = 1$, |
| | | $cert_i \leftarrow Sig_f(sk_s, pk_i)$, |
| $gsk_i \leftarrow (pk_i, sk_i, cert_i)$ | $\xleftarrow{cert_i}$ | $reg_i = pk_i$. |

Table 2: Algorithm Join of GS.

```
Algorithm Setup$(1^k)$:
$R \xleftarrow{\$} \{0,1\}^{P(k)}$, $R_1 \xleftarrow{\$} \{0,1\}^{P_1(k)}$,
$R_2 \xleftarrow{\$} \{0,1\}^{P_2(k)}$, $(pk_s, sk_s) \leftarrow K_s(1^k)$,
$(pk_e, sk_e) \leftarrow K_e(1^k)$,
$gpk = (R, R_1, R_2, pk_e, pk_s)$,
$ok = (sk_e)$, $ik = (sk_s)$.
return $(gpk, ok, ik)$.

Algorithm GVer$(gpk, m, \sigma)$:
Parse $\sigma$ as $(C, \Sigma', \pi_1)$,
Parse $gpk$ as $gpk = (R, R_1, R_2, pk_e, pk_s)$,
Return $V_1((pk_e, pk_s, m, C, \Sigma'), \pi_1, R_1)$.

Algorithm Judge$(gpk, reg, m, \sigma, i, M, \pi_2)$:
Parse $gpk$ as $gpk = (R, R_1, R_2, pk_e, pk_s)$,
Parse $\sigma$ as $(C, \Sigma', \pi_1)$,
If $GVer(gpk, m, \sigma) = 0$, return $\perp$.
If $i = 0$, and $M \neq Reg_j$ for all $j$,
return $V_2((pk_e, C, M), \pi_2, R_2)$,
else if $M = reg_i$,
return $V_2((pk_e, C, M), \pi_2, R_2)$.

Algorithm GSig$(gpk, gsk_i, m)$:
Parse $cert_i$ as $(\Upsilon, \Sigma)$,
Parse $gpk$ as $(R, R_1, R_2, pk_e, pk_s)$,
$(\Upsilon', \Sigma') = Rnd(gpk, sk_i, \Upsilon, \Sigma)$;
$C \leftarrow Enc(pk_e, pk_i, r_i)$, $r_i$ random;
$\pi_1 = P_1((pk_e, pk_s, m, C, \Sigma'), (sk_i, \Upsilon', r_i), R_1)$.
$\sigma = (C, \Sigma', \pi_1)$.
return $\sigma$.

Algorithm Open$(gpk, ok, reg, m, \sigma)$:
Parse $gpk$ as $gpk = (R, R_1, R_2, pk_e, pk_s)$,
Parse $\sigma$ as $(C, \Sigma', \pi_1)$,
If $GVer(gpk, m, \sigma) = 0$, return $\perp$.
$M \leftarrow Dec(sk_e, C)$,
If $M = reg_i$, $\exists i$,
$\pi_2 = P_2((pk_e, C, M), (sk_e), R_2)$,
return $(i, \tau)$, where $\tau = (M, \pi_2)$;
else return $(0, \tau)$, where $\tau = (M, \pi_2)$.
```

Table 3: Algorithms Setup, GSig, GVer, Open, Judge of GS.

**Comparison.** Notice the difference between ours construction and the generic construction in [BSZ05]:

In [BSZ05], the group signature is $\sigma = (C, \pi_1) = (Enc(pk_e, < i, pk_i, \Upsilon, \Sigma, s >, r_i), \pi_1)$, where $s = S(sk_i, m)$ and $\pi_1$ is a proof of knowledge of $(pk_i, \Upsilon, \Gamma, s, r_i)$ satisfying $Ver(pk_s, < i, pk_i >, (\Upsilon, \Sigma)) = 1$, $C = Enc(pk_e, < i, pk_i, \Upsilon, \Sigma, s >, r_i)$, and $V(pk_i, m, s) = 1$. $(S, V)$ is the signature generation and verification algorithms of an independent signature scheme.

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

Our construction is more efficient in that less items are to be encrypted in $C$, thus enabling efficient proof of knowledge of encrypted context, which is more clear from the specific examples in Section 4.3.

### 4.2 Security Proofs

The correctness of GS is obvious. The following proofs follow [BSZ05], and the detailed guideline of proofs are provided in Appendix C.1, C.2, C.3 respectively.

**Lemma 4.1.** *GS is anonymous if PE is IND-CCA, $(P_1, V_1)$ is a simulation sound, computational zero-knowledge proof, $(P_2, V_2)$ is a computational zero-knowledge proof.*

**Lemma 4.2.** *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 4.3.** *GS is non-frameable if $f$ 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.*

### 4.3 Examples

Many efficient group signatures have been proposed, e.g., [ACJT00], [CL04], [BBS04], [NSN04], [KY05]. We will replace their underlying signature scheme with the WoUF signatures listed above, substitute the encrypted content to $f(sk_i)$, and make some necessary revision in their proof of knowledge (i.e., signature of knowledge, denoted as $SK$ [CS97]) to match the change. The outputs of $SK\{x_1, ..., x_n : g_{i_1}^{x_{i_1}} \cdots g_{i_k}^{x_{i_k}} = A_i, \{i_1, ..., i_k\} \subseteq [1, n]\}\{m\}$ in the sequel (except 4.4) are standard, i.e., $\pi = (c, s_1, ..., s_n)$, where $c = H(...\|g_{1_1}^{s_{1_1}} \cdots g_{1_k}^{s_{1_k}} A_1^c\|...\|g_{i_1}^{s_{i_1}} \cdot \cdot \cdot g_{i_k}^{s_{i_k}} A_i^c\|...\|m)$, so the length of the output is associated with the number of the items to be proved, thus we do not have to go into the details of each $SK$ to compare their group signature lengths and computations, instead we can get a rough and enough comparison by counting the number of the items to be proved, since in our cases each item has a similar size.

### 4.4 Group Signature from CL02

Applying the proposed construction to the WoUF signature CL02 is a new group signature GS(CL02) with similar performance to [ACJT00], because they both have 7 items belonging to $Z_n^*$. The signature generation is more easier to realize, which is just a straightforward proof of knowledge of discrete logarithm, while more artifice are needed in [ACJT00] because $v$ is encrypted.

**GS(CL02).**
Member secret key: $x_i$. Member certificate $(\Upsilon, \Sigma) = (e, s, v)$
Relation: $v^e = a^{x_i} b^s c \bmod n$, $2^{l_e - 1} < e < 2^{l_e}$, $2^{l_s - 1} < s < 2^{l_s}$.
GSig.
1. Randomize his member certificate: $\Upsilon' = (e', s')$, $\Sigma' = (v')$
2. Encrypt $a^{x_i}$ as in [ACJT00]: $T_1 = a^{x_i} \mathsf{y}^r \bmod n$, $T_2 = \mathsf{g}^r \bmod n$; compute $\pi = SK\{s', e, x_i, r : T_1 = a^{x_i} \mathsf{y}^r \bmod n, T_2 = \mathsf{g}^r \bmod n, v'^{e'} = a^{x_i} b^{s'} c \bmod n, 2^{l_e - 1} < e < 2^{l_e}, 2^{l_s - 1} < s < 2^{l_s}\}\{m\}$.
3. The group signature is $(T_1, T_2, v', \pi)$.

### 4.5 Group Signature from BBS04+

In the end of introduction part of [BBS04], a new group signature using the method of [CL04] was briefly mentioned ("Their methodology can also be applied to the SDH assumption, yielding a different SDH-based group signature.").

We realize this new group signature GS(BBS04+) by applying the proposed construction to BBS04+ signature. The efficiency of GS(BBS04+) is almost the same as that of the one with strong exculpability [BBS04], but the signature generation of GS(BBS04+) is more easier to realize, which is just a straightforward proof of knowledge of discrete logarithm.

**GS(BBS04+).**
Member secret key: $x_i$. Member certificate $(\Upsilon, \Sigma) = (s, t, A)$
Relation: $e(A, w\tilde{g}^s) = e(h_1^{x_i} g, \tilde{g}^t)$.
GSig.

1. Randomize his member certificate: $\Upsilon' = (s, rt)$, $\Sigma' = (A^r)$, where $r \overset{\$}{\leftarrow} Z_p^*$
2. Encrypt $h_1^{x_i}$ as in [BBS04]: $T_1 = u^\alpha$, $T_2 = v^\beta$, $T_3 = \tilde{g}^s h^{\alpha+\beta}$
3. Let $S = A^r$, compute $\pi = SK\{s, t, x_i, \alpha, \beta : T_1 = u^\alpha, T_2 = v^\beta, T_3 = \tilde{g}^s h^{\alpha+\beta}, e(S, w\tilde{g}^s) = e(h_1^{x_i} g, \tilde{g}^t)\}\{m\}$.
4. The group signature is $(S, T_1, T_2, T_3, \pi)$.

### 4.6 Group Signature from CL04+, CL04*

The group signature of [CL04] is in fact an application of the proposed construction to CL04+ signature, see GS(CL04+) in the following table.

It can be improved in signature length by applying the construction to CL04*, the new group signature GS(CL04*) has less items than GS(CL04+), i.e., 170 bits shorter under the same security parameters of [NSN04, Ngu05].

**GS(CL04+)**: Please refer to [CL04] for the notations.
Member secret key: $k$. Member certificate: $a, b, c$
Relation: $e(a, Y) = e(g, b)$, $e(X, a)e(X, b)^k = e(g, c)$
GSig:
1. Encrypt $\Delta = e(g^k, g)$: $c_1 = \mathsf{g}^u, c_2 = \mathsf{h}^u, c_3 = \mathsf{y_1}^u \Delta, c_4 = \mathsf{y}_2^u \mathsf{y}_3^{u\mathcal{H}(c_1\|c_2\|c_3)}$
2. Randomize $(a, b, c) \to (\tilde{a}, \tilde{b}, \hat{c})$, compute $\pi = SK\{k, \rho, u : e(X, \tilde{a})e(X, \tilde{b})^k = e(g, \hat{c})^\rho, c_1 = \mathsf{g}^u, c_2 = \mathsf{h}^u,$
$c_3 = \mathsf{y_1}^u \Delta, c_4 = \mathsf{y}_2^u \mathsf{y}_3^{u\mathcal{H}(c_1\|c_2\|c_3)}\}\{m\}$.
3. The group signature is $(\tilde{a}, \tilde{b}, \hat{c}, c_1, c_2, c_3, c_4, \pi)$.

**GS(CL04*)**:
Member secret key: $k$. Member certificate: $s, a, b$
Relation: $e(b, \tilde{g}) = e(a, \tilde{X}\tilde{Y}^k)e(X, \tilde{Y})^s$
GSig:
1. Encrypt $\Delta = e(g^k, g)$: $c_1 = \mathsf{g}^u, c_2 = \mathsf{h}^u, c_3 = \mathsf{y_1}^u \Delta, c_4 = \mathsf{y}_2^u \mathsf{y}_3^{u\mathcal{H}(c_1\|c_2\|c_3)}$
2. Randomize $(s, a, b) \to (s', a', b')$, compute $\pi = SK\{k, s', u : e(b', \tilde{g}) = e(a', \tilde{X}\tilde{Y}^k)e(X, \tilde{Y})^{s'}, c_1 = \mathsf{g}^u, c_2 = \mathsf{h}^u,$
$c_3 = \mathsf{y_1}^u \Delta, c_4 = \mathsf{y}_2^u \mathsf{y}_3^{u\mathcal{H}(c_1\|c_2\|c_3)}\}\{m\}$.
3. The group signature is $(a', b', c_1, c_2, c_3, c_4, \pi)$.

### 4.7 Group Signature from NSN04*

The efficient group signature of [NSN04] can be improved in signature length by applying the generic construction to NSN04*, the new group signature GS(NSN04*) has less items than [NSN04], i.e., 170 bits shorter under the same security parameters of [NSN04][1].

[1]This group signature can be further improved as described in Appendix D, which will be shorter than GS(NSN04*), but GS(NSN04*) has an advantage of constant context to be encrypted if reversed dynamic accumulator, an efficient membership revocation solution, is integrated with the plain group signature [ZL06a].

**[NSN04]**: Please refer to [NSN04] for the notations.
Member secret key: $x_i$.     Member certificate: $a_i, S_i$
Relation: $e(a_iP + P_{pub}, S_i) = e(P, x_iP + P_0)$
GSig:
1. Encrypt $\Delta_i$=$e(S_i, P)$: $E_a = tG, \Lambda_a = \Delta_i\Theta_a^t, E_b, \Lambda_b, \varsigma$
2. Set $U_1 = S_i + r_1H$, $R = r_1G$, compute $\pi = (c, s_1, s_2, s_3, s_4, s_5) = SK\{t, r_1, a_i, r_1a_i, x_i : E_a = tG,$ $R = r_1G$, $0 = r_1a_iG - a_iR$, $e(P, U_1)\Lambda_a^{-1} = e(P, H)^{r_1}\Theta_a^{-t}$, $e(P, U_1)^{a_i}e(P_{pub}, U_1) = e(P, P)^{x_i}e(P, P_0)$ $e(P, H)^{r_1a_i}e(P_{pub}, H)^{r_1}\}\{m\}$.
3. The group signature is $(E_a, \Lambda_a, E_b, \Lambda_b, \varsigma, U_1, R, c, s_1, s_2, s_3, s_4, s_5)$.
Note: SK of [NSN04, Ngu05] was found flawed [ZC05]. The SK here follows the modified version of [Ngu05] excluding revocation.

**GS(NSN04*)**:
Member secret key: $x_i$.     Member certificate: $a_i, b_i, c_i, A_i$
Relation: $e(a_iP + P_{pub}, A_i) = e(P, x_iP + b_iP_{pub} + P_0)\ e(c_iP_{pub}, P_{pub})$
GSig:
1. Encrypt $\Delta_i$=$e(x_iP, P)$: $E_a = tG, \Lambda_a = \Delta_i\Theta_a^t, E_b, \Lambda_b, \varsigma$
2. Randomize $(a_i, b_i, c_i, A_i) \to (a_i', b_i', c_i', A_i')$, compute $\pi = (c, s_1, s_2, s_3, s_4, s_5) = SK\{t, a_i', b_i', c_i', x_i :$ $E_a = tG$, $\Lambda_a = e(P, P)^{x_i}\Theta_a^t$, $e(P, A_i')^{a_i}e(P_{pub}, A_i') = e(P, P)^{x_i}e(P, P_{pub})^{b_i'}e(P, P_0)e(P_{pub}, P_{pub})^{c_i'}\}\{m\}$.
3. The group signature is $(E_a, \Lambda_a, E_b, \Lambda_b, \varsigma, A_i', c, s_1, s_2, s_3, s_4, s_5)$.

## 4.8 Other Group Signatures

The proposed generic construction can not be immediately applied to BB04+ signature because it is not indirectly signable, but by replacing the [BB04] signature by BB04+ in the group signature with concurrent join [KY05], we will get a more efficient scheme KY05+ (Section 4.8.1), with group signature length 1190 bits shorter.

Because there does not exist a one way function in [Wat05]+ signature, and no other substitutions are known in this case, the construction can not be applied to it right now.

### 4.8.1 Group Signature KY05+

Public parameters and algorithms Setup, Join (Table 4), Open are exactly as [KY05], except that key-setup for linear ElGamal encryption is eliminated.

| User | | Issue Authority |
|---|---|---|
| $x = x_1x_2$, where $x \in\overset{\$}{\leftarrow} S'$ and $x_1 \in\overset{\$}{\leftarrow} S''$, | $\xrightarrow{x}$ | If $x \in S'$, $r \overset{\$}{\leftarrow} Z_p^*$, $s \overset{\$}{\leftarrow} Z_p^*$ |
| $e(\sigma, wg_2^xv^r)? = e(g_1, g_2)^s$ | $\xleftarrow{(r,s,\sigma)}$ | $\sigma \leftarrow g_1^{\frac{s}{\gamma+x+\delta r}}$ |
| $cert = (x, r, s, \sigma)$, $msk = (x_1, x_2)$ | | |

Table 4: Algorithms Join of [KY05].

**GSig.** A member firstly calculates ($\sigma'$, $s'$, $W_1$, $W_2$, $C_0$, $C_1$, $C_2$), then generates a proof of knowledge $\pi$ of $(\theta_x, \theta_{x_1}, \theta_{x_2}, \theta_y, \theta_{y'}, \theta_{yx_2}, \theta_t, \theta_r, \theta_{s'})$ that satisfies some specified relations (Table 5). The generation method is quite similar to GSig in [KY05] except that some witnesses and relations are different, e.g., $\sigma', W_1, W_2$ are newly introduced into KY05+, while $C_0, C_1, C_2$ are inherited from [KY05]. A major difference is that the number of witnesses that need proving is fewer than that of [KY05]. Thus a group signature of KY05+ is $(\sigma', W_1, W_2, C_0, C_1, C_2, \pi)$, about $7|p| = 1190$ bits shorter than [KY05].

If we view $x = x_1x_2$ 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 [KY05].

| | |
|---|---|
| $\sigma' = \sigma^{r'}$, $s' = r's$ | $r' \xleftarrow{\$} Z_p^*$ in $G_1$ |
| $W_1 = g^y f_1^{x_1}$ | $y \xleftarrow{\$} S(1, 2^{l_n - 2})$ in $QR(n)$ |
| $W_2 = g^{y'} f_2^{x_2} f_3^t$ | $y' \xleftarrow{\$} S(1, 2^{l_n - 2})$ 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 $Z_{N^2}^*$ |
| $C_2 = \Vert (H_2 H_3^{\mathcal{H}(hk, C_0, C_1)})^t \Vert$ | in $Z_{N^2}^*$ |
| $g^{\theta_y} f_1^{\theta_{x_1}} = W_1$, | $g^{\theta_{y'}} f_2^{\theta_{x_2}} f_3^{\theta_t} = W_2$, |
| $W_1^{-\theta_{x_2}} g^{\theta_{yx_2}} f_1^{\theta_x} = 1$, | $e(\sigma', w g_2^{\theta_x} v^{\theta_r}) = e(g_1, g_2)^{\theta_{s'}}$, |
| $G^{\theta_t} = C_0$, | $H_1^{\theta_t} (1+N)^{\theta_x} = C_1$, |
| $(H_2 H_3^{\mathcal{H}(hk, C_0, C_1)})^{2\theta_t} = C_2^2$, | $\theta_x \in S'$, $\theta_{x'} \in S''$. |

Table 5: Algorithm GSig of KY05+.

## 5 Conclusion

By the help of a kind of weakly unforgeable signatures, i.e., those that are not strongly unforgeable and are indirectly signable, perfectly unlinkable (the latter important feature has not been identified before), we have obtained group signatures with shorter signature length or simpler signature generation. We propose improvements to the group signature schemes of CL04, NSN04, KY05.

WoUF signatures may be also advantageous in other cases, which may needs more investigation.

## References

[ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO'00*, LNCS 1880, pages 255–270. Springer-Verlag, 2000.

[ADR02] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In *EUROCRYPT'02,*, LNCS 2332, pages 83–107. Springer-Verlag Berlin Heidelberg, 2002.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *EUROCRYPT'04*, LNCS 3027, pages 56–73. Springer-Verlag, 2004.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO'04*, LNCS 3152, pages 45–55. Springer-Verlag, 2004.

[BG90] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In *CRYPTO'89*, LNCS 435, pages 194–211. Springer-Verlag, 1990.

[BS04] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *CCS'04*, LNCS 3108, pages 168–177. ACM Press, 2004.

[BSW06] Dan Boneh, Emily Shen, and Brent Waters. Strongly unforgeable signatures based on computational difie-hellman. In *PKC'06*, LNCS 3958, pages 229–240. Springer Berlin Berlin Heidelberg, 2006.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA'05*, LNCS 3376, pages 136–153. Springer-Verlag, 2005. Full Paper at http://www-cse.ucsd.edu/ mihir/papers/dgs.html.

[CD95] Ronald Cramer and Ivan Damgård. Secure signature schemes based on interactive protocols. In *CRYPTO'95*, LNCS 963, pages 297–310. Springer-Verlag, 1995.

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO'02*, LNCS 2442, pages 61–76. Springer-Verlag, 2002.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO'04*, LNCS 3152, pages 56–72. Springer-Verlag, 2004.

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *CRYPTO'06*, LNCS 4117, pages 78–96. Springer-Verlag, 2006.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signatures schemes for large groups. In *CRYPTO'97*, LNCS 1296, pages 410–424. Springer-Verlag, 1997.

[CvH91] D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT'91*, LNCS 547, pages 257–265. Springer-Verlag, 1991.

[DK01] Hans Delfs and Helmut Knebl. *Introduction to Cryptography : Principles and Applications.* Springer, December 2001.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractor. In *Crypto'05*, LNCS 3621, pages 152–168. Springer-Verlag Berlin Heidelberg, 2005.

[FS87] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *CRYPTO'86*, LNCS 263, pages 186–194. Springer, 1987.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. 17(2):281–308, 1988.

[Gol01] Oded Goldreich. *Foundations of Cryptography*, volume Basic Tools. Cambridge University Press, 2001.

[Hes03] Florian Hess. Efficient identity based signature schemes based on pairings. In *SAC'02: Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography*, pages 310–324, London, UK, 2003. Springer-Verlag.

[HSM05] Xinyi Huang, Willy Susilo, and Yi Mu. Breaking and repairing trapdoor-free group signature schemes from asiacrypt 2004. Cryptology ePrint Archive, Report 2005/122, 2005.

[KY04] Aggelos Kiayias and Moti Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. In *Cryptology ePrint Archive*, Report 2004/076, 2004.

[KY05] Aggelos Kiayias and Moti Yung. Group signatures with efficient concurrent join. In *EUROCRYPT'05*, LNCS 3494, pages 198–214. Springer-Verlag, 2005.

[MvOV96] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[Ngu05] Lan Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA'05*, LNCS 3376, pages 275–292. Springer-Verlag, 2005. A modified version is available at Cryptology ePrint Archive: Report 2005/123.

[NSN04] Lan Nguyen and Rei Safavi-Naini. Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings. In *ASIACRYPT'04*, LNCS 3329, pages 372–386. Springer-Verlag, 2004.

[Pet97] Holger Peterson. How to convert any digital signature scheme into a group signature scheme. In *Proc. Security Protocols Workshop, Paris*, LNCS 1361, pages 177–190. Springer-Verlag, 1997.

[Sch91] C. P. Schnorr. Effcient signature generation by smart cards. In *Journal of Cryptology*, volume 4, pages 161–174, 1991.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT'05*, LNCS 3494, pages 114–127, 2005.

[ZC05] Fangguo Zhang and Xiaofeng Chen. Cryptanalysis and improvement of an id-based ad-hoc anonymous identification scheme at ct-rsa 05. Cryptology ePrint Archive, Report 2005/103, 2005.

[ZL06a] Sujing Zhou and Dongdai Lin. Group signatures with reduced bandwidth. *IEE Proceedings Information Security*, 2006. To appear.

[ZL06b] Sujing Zhou and Dongdai Lin. Shorter verifier-local revocation group signatures from bilinear maps. In *CANS 06*, LNCS 4301, pages 126–143. Springer-Verlag, 2006.

## A Proof of Lemma 3.1

*Proof.* Suppose there exists an adversary $\mathcal{B}$ to the signature, we now construct an adversary $\mathcal{A}$ to resolve $q$-SDH problem ([BB04]) in $G$: to calculate $(c, \frac{1}{z+c}Q), c \in Z_p^*$ given a random tuple $(Q, zQ, ..., z^qQ)$.

$\mathcal{B}$ 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 $(m_i, a_i, b_i, c_i, A_i), i = 1, ..., q_{sig}$, $\mathcal{B}$ wins by outputting a forgery, i.e., a new message-signature $(m^*, a^*, b^*, c^*, A^*)$ that $m^* \notin \{m_1, ..., m_{q_{sig}}\}$. 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 $(Q, zQ, ..., z^qQ)$.

When $\mathcal{B}$ queries about a signature on $m_i$, $\mathcal{A}$ firstly selects $b_i, c_i \xleftarrow{\$} Z_p^*$, calculates $A_i = \frac{1}{z+a_i}[m_iP + (b_i - a_ic_i)P_{pub} + P_0] + c_iP_{pub}$, which is computable from $(Q, zQ, ..., z^qQ)$ 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 $m^*+s = (b^*-a^*c^*)a^*$ is negligible otherwise $\mathcal{B}$ can be invoked to solve discrete logarithm problem in $G$ if $z$ is chosen by $\mathcal{A}$ and $sQ$ is given as a discrete logarithm challenge. Then there exist $g(z)$, $r \neq 0 \bmod p$ 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 $(Q, zQ, ..., z^qQ)$, 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 $(Q, zQ, ..., z^qQ)$.

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 $A_i = \frac{1}{z-a_l+a_i}[m_iP + (b_i - a_ic_i)P_{pub} + P_0] + c_iP_{pub}$, which is computable from $(Q, zQ, ..., z^qQ)$ 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 $b_l - a_lc_l = (d + m_l)a_l^{-1}$, and $s = t + (d + m_l)a_l^{-1}$, then it can be verified that $m_lP + (b_l - a_lc_l)P_{pub} + P_0 = szP$, so $A_l = \frac{1}{\gamma+a_l}[m_lP + (b_l - a_lc_l)P_{pub} + P_0] + c_lP_{pub} = sP + c_lP_{pub}$ 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 $m^*-a_l(b^*-a^*c^*)+d = 0 \bmod p$ is negligible otherwise $\mathcal{B}$ can be invoked to solve discrete logarithm problem in $G$ if $z$ is chosen by $\mathcal{A}$ and $dQ$ is given as a discrete logarithm challenge. Then there exist $g(z)$, $r \neq 0 \bmod p$ 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 $(Q, zQ, ..., z^qQ)$, is a resolution to the $q$-SDH challenge. Note that any algorithm for $\frac{1}{z}Q$ can be used to calculate a $(c \neq 0, \frac{1}{z+c}Q)$. $\square$

## B A Formal Model of Group Signature - A Variant of [BSZ05]

[BSZ05]'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 [BSZ05]: 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 judgers (the executors of algorithm Judge).

We define the oracles similar to [BSZ05]. 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.

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

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

Table 6: Correctness.

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

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

where experiments $Exp^{anon-b}_{GS,\mathcal{A}}(k)$ 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* [BS04].

If $\{i_0, i_1\} \subseteq CU$, and *CrptIA* is not queried, the group signature is *anonymous* in the sense of [KY04].

If $\{i_0, i_1\} \subseteq HU$, and *CrptIA* is queried, the group signature is *anonymous* in the sense of [BSZ05].

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 7.

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

Table 7: 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 $\text{Adv}^{trace}_{GS,\mathcal{A}}$ defined below is negligible.

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

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

If *CrptOA* is not queried, the group signature is secure against *misidentification attack* [KY04].

If *CrptOA* is queried, the group signature is *traceable* in the sense of [BSZ05].

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 8.

```
Experiment $Exp^{trace}_{GS,A}(k)$
$(gpk, ik, ok) \xleftarrow{\$} \text{Setup}(1^k)$; $CU \leftarrow \varnothing$, $HU \leftarrow \varnothing$;
$(m, \sigma) \xleftarrow{\$} A(gpk : CrptOA, CrptU, SndToI, RReg)$.
If $\text{GVer}(gpk, m, \sigma) = 0$, 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 8: 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 $CrptU(i_h)$, $SndToI(i_h,.)$, $USK(i_h)$.

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

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

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

If *CrptIA* and *CrptOA* are queried, the group signature is secure against *framing attack* [KY04] or non-frameable [BSZ05].

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 9.

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

Table 9: Non-frameability.

**Definition 7.** *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 4.1

Note that the difference between ours construction 4.1 and the generic construction in [BSZ05] is that, our ultimate group signature is $\sigma = (C, \Sigma', \pi_1) = (Enc(pk_e, pk_i, r_i), \Sigma', \pi_1)$, where $\pi_1$ is a proof of knowledge of $(sk_i, \Upsilon', r_i)$ satisfying $Ver(pk_s, sk_i, (\Upsilon', \Sigma')) = 1$ and $C = Enc(pk_e, f(sk_i), r_i)$; while the ultimate group signature of [BSZ05] is $\sigma = (C, \pi_1) = (Enc(pk_e, < i, pk_i, \Upsilon, \Sigma, s >, r_i), \pi_1)$, where $s = S(sk_i, m)$ and $\pi_1$ is a proof of knowledge of $(pk_i, \Upsilon, \Gamma, s, r_i)$ satisfying $Ver(pk_s, < i, pk_i >, (\Upsilon, \Sigma)) = 1$, $C = Enc(pk_e, < i, pk_i, \Upsilon, \Sigma, s >, r_i)$, 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 [BSZ05], i.e., $\Sigma'$, 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 [BSZ05].

The proof follows [BSZ05]. 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 $Ch_{PE}(b, ., .)$ and $Dec(sk_e, .)$.

$\mathcal{A}_c$ 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 $(m, \sigma) = (m, C, \Sigma', \pi_1)$ is valid and $C$ is not returned by $Ch(c, ., .)$, queries oracle $Dec(sk_e, .)$, and generates a simulation proof for $\rho_2$.

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

*USK*$(i)$: Returns $(pk_i, sk_i, \Upsilon, \Sigma)$, 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 $Ch_{PE}(b, M_0, M_1)$ ($M_c = (pk_{i_c})$, $M_{\overline{c}} = (0^{|M_c|})$), 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 $(m, \hat{\sigma})$=$(m, C, \hat{\Sigma}, \hat{\pi})$ from the challenge $(m, \sigma)$=$(m, C, \Sigma', \pi_1)$, 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 $(m, \sigma) = (m, C, \Sigma', \pi_1)$, is valid and $C$ is not returned by $Ch(b, ., .)$, runs algorithm Open since $\mathcal{A}_s$ knows $ok (= sk_e)$, and generates a simulation proof for $\rho_2$.

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

*USK*$(i)$: Returns $(pk_i, sk_i, \Upsilon, \Sigma)$, 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 $(C, \Sigma', \pi_1)$.

*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 $(m, \hat{\sigma}) = (m, C, \hat{\Sigma}, \hat{\pi})$ from the challenge $(m, \sigma) = (m, C, \Sigma', \pi_1)$, in which case $\mathcal{A}_s$ outputs $(pk_e, pk_s, m, C, \hat{\Sigma})$ and $\hat{\pi}$.

**Description of $\mathcal{D}_1$.** $\mathcal{D}_1$ is given the common reference string $R_1$, and access to oracle $Prove_1(.)$ 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*$(i,.)$: Runs as algorithm Join, adds $i$ to the honest member set $HU$.

*USK*$(i)$: Returns $(pk_i, sk_i, \Upsilon, \Sigma)$, 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 $(gpk, gsk_{i_b}, m)$ 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 $Prove_2(.)$ 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*$(i,.)$: Runs as algorithm Join, adds $i$ to the honest member set $HU$.

*USK*$(i)$: Returns $(pk_i, sk_i, \Upsilon, \Sigma)$, 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 $(gpk, gsk_{i_b}, m)$.

*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 [BSZ05] that

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

## C.2 Proof of Lemma 4.2

The proof follows [BSZ05]. 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 $Sig(sk_s, .)$.

$\mathcal{A}_{ds}$ 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*$(i,.)$: 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 $(m, \sigma) = (m, C, \Sigma', \pi_1)$ 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 [KY04], (GVer be the predicate), in random oracle model, there exist $(m, C, \Sigma', c, s)$, $(m, C, \Sigma', c', s')$ from which $(w, \Upsilon', r)$ can be extracted, $(\Upsilon', \Sigma')$ is a valid $DS$ signature on $w$, and $w$ is not queried to $Sig(sk_s, .)$.

It follows from the same analysis in [BSZ05] that

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

## C.3 Proof of Lemma 4.3

The proof follows [BSZ05]. 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*$(i,.)$: 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 $(pk_i, sk_i, \Upsilon, \Sigma)$, 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 $(\iota, m, C, \Sigma', c, s)$, then there exist $(\iota, m, C, \Sigma', c, s)$, $(m, C,\ \Sigma', c', s')$ in random oracle model according to generalized forking lemma [KY04], (GVer be the predicate), so $(w, \Upsilon', r)$ can be extracted, where $(\Upsilon', \Sigma')$ is a valid $DS$ signature on $w$, and $f(w) = y$.

It follows from a similar analysis in [BSZ05] that $\mathrm{Adv}^{nf}_{GS,\mathcal{B}}(k) \leq \varepsilon(k) + n(k)\mathrm{Adv}^{ow}_{f,\mathcal{A}_f}(k)$, where $\varepsilon(k)$ is negligible.

## D An Improvement to the Group Signature in [NSN04]

We present an improvement to [NSN04] by producing shorter group signature length while maintaining a similar computation.

The improvement is based on the basic proof of knowledge: $PK\{X : e(X, Q) = e(P_1, Q_1)\}$, $(e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_M,\ ord(\mathbb{G}_M) = p, \mathbb{G}_1 = \langle P \rangle)$, the pairing version of Schnorr's identification scheme [Sch91], which has been widely adopted in identification based signature schemes from pairings, e.g., [Hes03]. The details of the proof of knowledge are as follows:

1. Prover generates $R_X \in_R \mathbb{G}_1$, computes $R = e(R_X, Q)$,
2. Prover $\longrightarrow$ Verifier: $R$.
3. Verifier $\longleftarrow$ Prover: $c \in_R \mathbb{Z}_p$.
4. Prover computes in $S = R_X - cX$.
5. Prover $\longrightarrow$ Verifier: $S$.
6. Verifier verifies that if $R = e(S, Q)e(P_1, Q_1)^c$.

Note that the scheme of [NSN04] were found flawed in [HSM05], so we compare our improvement with [HSM05] and the scheme adopted from applying the modified version of [Ngu05]. We follow the the original notations to facilitate the comparison.

To improve the group signature in [NSN04], we generate the following signature of knowledge:

$$\begin{aligned} SK\{(a_i, ta_i, x_i, t, S_i) :& \Lambda_a^{a_i}\Theta_a^{-ta_i}e(S_i, P_{pub}) = e(P, P)^{x_i}e(P, P_0), \\ & E_a = tG, a_iE_a = ta_iG, \Lambda_a = e(S_i, P)\Theta_a^t\}\{m\} \end{aligned}$$

The details: Select $r_1, r_2, r_3, r_4 \in_R \mathbb{Z}_p$, $R_s \in_R \mathbb{G}_1$, calculate

$$\begin{aligned}
&R_1 = \Lambda_a^{r_1}\Theta_a^{-r_2}e(P,P)^{-r_3}e(R_s,P_{pub}),\\
&R_2 = t_4G, \quad R_3 = r_1E_a - r_2G, \quad R_4 = e(R_s,P)\Theta_a^{r_4},\\
&c = \mathcal{H}_2(P\|P_0\|P_{pub}\|H\|G\|\Theta\|E_a\|\Lambda_a\|E_b\Lambda_b\|\varsigma\|R_1\|R_2\|R_3\|m),\\
&s_1 = r_1 - ca_i, \quad s_2 = r_2 - cta_i, \quad s_3 = r_3 - cx_i,\\
&s_4 = r_4 - ct, S = R_s - cS_i.
\end{aligned}$$

The group signature of $m$ is $\sigma = (E_a, \Lambda_a, E_b, \Lambda_b, \varsigma, s_1, s_2, s_3, s_4, S, c)$.
To verify the above group signature, check if

$$\begin{aligned}
c =&\mathcal{H}_2(P\|P_0\|P_{pub}\|H\|G\|\Theta\|E_a\|\Lambda_a\|E_b\Lambda_b\|\varsigma\|\Lambda_a^{s_1}\Theta_a^{-s_2}e(P,P)^{-s_3}e(S,P_{pub})\\
&e(P,P_0)^c\|s_4G + cE_a\|s_1E_a - s_2G\|e(S,P)\Theta_a^{s_4}\Lambda_a^c\|m).
\end{aligned}$$

**Explanation for $SK$.** Consider the underlying interactive proof of the above $SK$. Its zero-knowledge is easy to see. Its soundness: By resetting Prover under the same random inputs, an honest verifier can get $(s_1, s_2, s_3, s_4, S, c)$ and $(s_1', s_2', s_3', s_4', S', c')$ where $s_j' \neq s_j$, $j = 1, 2, 3, 4$, $S' \neq S$, $c' \neq c$, then

$$\begin{aligned}
&\Lambda_a^{s_1}\Theta_a^{-s_2}e(P,P)^{-s_3}e(S,P_{pub})e(P,P_0)^c = \Lambda_a^{s_1'}\Theta_a^{-s_2'}e(P,P)^{-s_3'}e(S',P_{pub})e(P,P_0)^{c'},\\
&s_4G + cE_a = s_4'G + c'E_a, \quad s_1E_a - s_2G = s_1'E_a - s_2'G,\\
&e(S,P)\Theta_a^{s_4}\Lambda_a^c = e(S',P)\Theta_a^{s_4'}\Lambda_a^{c'}.
\end{aligned}$$

Let $\Delta s_j = s_j - s_j'$, $j = 1, 2, 3, 4$, $\Delta S = S - S'$, $\Delta c = c' - c$, then

$$\begin{aligned}
&\Lambda_a^{\Delta s_1}\Theta_a^{-\Delta s_2}e(P,P)^{-\Delta s_3}e(\Delta S,P_{pub}) = e(P,P_0)^{\Delta c},\\
&\Delta s_4G = \Delta cE_a = 0, \quad \Delta s_1E_a - \Delta s_2G = 0,\\
&e(\Delta S,P)\Theta_a^{\Delta s_4} = \Lambda_a^{\Delta c}.
\end{aligned}$$

Set $S_i = \frac{1}{\Delta c}\Delta S$, $t = \Delta s_4/\Delta c \bmod p$, $a_i = \Delta s_1/\Delta c \bmod p$, $x_i = \Delta s_3/\Delta c \bmod p$, then $\Delta s_2/\Delta c = ta_i \bmod p$.

**Efficiency Comparison.** The length of the group signature $\sigma$ is shorter compared with [HSM05] ($\sigma = (E_a, \Lambda_a, E_b, \Lambda_b, \varsigma, U, V, W, X, \Gamma_1, \Gamma_2, s_0, s_1, s_2, s_3, s_4, s_5, c)$ there) or applying the modified version of [Ngu05] to this group signature (see Section 4.7).