Batch Verification and Finding Invalid Signatures in a Group Signature Scheme

Kitae Kim[1,3], Ikkwon Yie[1], Seongan Lim[2], and Daehun Nyang[3]

(Corresponding author: Kitae Kim)

Department of Mathematics, Inha University[1]

253, Yonghyun dong, Nam gu, Incheon, 402-751, Korea (Email: ktkim@inha.ac.kr)

Department of Mathematics, Ewha Womans University, Korea[2]

ISRL, Graduate School of Information Technology and Telecommunications, Inha University, Incheon, Korea[3]

(Received May 20, 2009; revised and accepted Mar. 24, 2010)

Abstract

Batch cryptography has been developed into two main branches - batch verification and batch identification. Batch verification is a method to determine whether a set of signatures contains invalid signatures, and batch identification is a method to find bad signatures if a set of signatures contains invalid signatures. Recently, some significant developments appeared in such field, especially by Ferrara et al. [10] and Law et al. [15], respectively. However, no batch identification method for group signature have been suggested. In this paper, by exploiting earlier works on standard signature schemes, we propose methods of batch verification and batch identification of a group signature scheme.

Keywords: Batch identification, batch verification, group signature

1 Introduction

Currently, digital signatures have been adapted in many industrial applications such as electronic payment system, electronic voting system and so on. Some of the applications require multiple signatures to be verified faster than individual verification of the signatures. For instance, in electronic payment system, typically customers interact with a banking server, and then the banking server must verify a large number of signatures. Concerning verification of multiple signatures there are mainly two questions: given signature/message pairs, without checking the validity of individual signatures, (1) determine efficiently whether an instance contains invalid signatures; (2) identify efficiently invalid signatures, if any, in an instance.

A batch verification of signature scheme provides a solution of the first question. A batch verification algorithm (or a batch verifier) of signatures is defined as follows. A batch verifier of signatures is a probabilistic algorithm that takes as input a security parameter ℓ and a batch instance (signature/message pairs), satisfying (1) if all the members of an instance are valid then it returns true, (2) if there are invalid signatures then the probability that it returns true is at most $2^{-\ell}$. A batch verification method verifies multiple signatures altogether at once and reduces verification time compared with individual verification. The concept of batch cryptography was introduced by Fiat in 1984 for an RSA-type signature [11], and the first efficient batch verifier was proposed by Naccache, Raihi, Vaudenay, and Raphaeli in 1994 for DSA-type signatures [19]. Since then several batch verification methods have been proposed for DSA-type, RSA-type, and Pairing based systems. In particular, Ferrara, Green, Hohenberger, and Pedersen [10] proposed a first batch verifier for a (short) group signature scheme.

To the second question, a few methods have been proposed to identify bad signatures efficiently. In 2000, Pastuszak, Michalek, Pieprzyk, and Seberry [22] proposed a divide and conquer verifier, which splits an instance into sub-instances and applies the generic test to each sub-instance recursively until all bad signatures are identified. Later, Lee, Cho, Choi, and Cho [16] proposed new method for identifying bad signatures efficiently in RSA-type batch signatures. Stanek showed that this method was flawed and proposed an improved method to repair his attack [23]. In 2006, Law and Matt [15] proposed new methods, quick binary and exponentiation method, for finding invalid signatures in some pairing based signature schemes in which the verification algorithm has a special form, specifically for Cha-Cheon signature scheme [8]. Their method shares the basic idea with Lee et al.'s method [16], and the authors did not consider in group signature schemes.

Recently, Ferrara et al. [10] presented a general framework of batch verification of pairing based signature schemes including a group signature scheme, specifically Boneh-Boyen-Shacham short group signature scheme (BBS for short) [4], and investigated an implemental result of finding invalid signatures in Boneh-Lysyanskaya-

Shachaum short signature scheme (BLS for short) [5], which is not a group signature scheme. Especially, to find invalid signatures in batches of BLS signature scheme, the authors employed the Patuszak et al.'s *divide-and-conquer* approach [22]. Though the authors suggested a batch verifier for BBS group signature scheme, they did not considered how can one construct batch identification algorithms for group signature schemes. Indeed, we can exploit the divide-and-conquer method to find invalid signatures in obvious way.

On the other hand, as noted in [10], Law et al.'s methods [15] can be applied to BLS short signature scheme because the verification algorithm of BLS short signature has almost the same form with those of the signature schemes in [15] in the sense that given signature is valid if and only if two pairings are equal. However, as well as BBS group signature scheme, Delerablee-Pointcheval (dynamic) group signature scheme (namely XSGS scheme) [9] has slightly different verification algorithm - there is an extra non-pairing multiplication.

In this paper, by exploiting Ferrara et al.'s general framework [10], we construct a batch verification of Delerablee-Pointcheval group signature scheme [9], for which no batch verifier has been considered so far. Furthermore, we propose a batch identification method for XSGS by exploiting Law et al.'s method [15]. Our batch identification method can be directly applied to BBS group signature scheme (Ferrara et el.'s batch verifier for BBS group signature scheme) due to the similarity of BBS group signature and XSGS scheme. To the best of our knowledge, no batch identification methods have been suggested in group signature schemes.

2 Preliminaries

2.1 Batch Verification

Batch verification of digital signatures was introduced by Naccache et al. [19] to verify multiple signatures in DSA signature scheme. The definition of batch verification and the weaker notion, say screening, were formalized by Bellare et al. [3]. The following definition is due to Camenisch et al. [7] in which they extended the definition of Bellare et al. to deal with multiple signers.

Definition 1 (Batch verification of signatures). *Let ℓ be the security parameter. Suppose that (pk_1, sk_1), $\cdots$, (pk_n, sk_n) are generated independently according to* $\mathsf{Gen}(1^\ell)$. *Then we call probabilistic* Batch *a batch verification when following conditions hold:*

- *If* $\mathsf{Verify}(pk_{t_i}, m_i, \sigma_i) = 1$ *for all* $i \in [1, n]$, *then* $\mathsf{Batch}((pk_{t_1}, m_1, \sigma_1), \ldots, (pk_{t_n}, m_n, \sigma_n))$=1.
- *If* $\mathsf{Verify}(pk_{t_i}, m_i, \sigma_i) = 0$ *for any* $i \in [1, n]$, *then* $\mathsf{Batch}((pk_{t_1}, m_1, \sigma_1), \ldots, (pk_{t_n}, m_n, \sigma_n))$=0 *except with probability negligible in k, taken over the randomness of* Batch.

2.2 Review of Extremely Short Group Signature Scheme

Delerabee and Pointcheval [9] proposed a dynamic group signature scheme (XSGS) that provides the strongest security level in the random oracle model - full anonymity, full traceability, and non-frameability, while BBS group signature scheme is proved in a weaker security model, say full-cpa-anonymity and full-traceability.

2.2.1 Bilinear Pairing

The XSGS scheme uses bilinear pairing on cyclic groups, which can be described briefly as follows: For two additive cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$, and a multiplicative cyclic group $\mathbb{G}_T$ of order p, we assume that there is an efficiently computable function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, with following properties:

- Bilinear: for all $A, B \in G$ and $a, b \in \mathbb{Z}_p$, $e(aA, bB) = e(A, B)^{ab}$.
- Non-degeneracy: For two non-identity elements $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$, $e(A, B)$ is a generator of the group $\mathbb{G}_T$.

2.2.2 Delerablee-Pointcheval's Group Signature Scheme (XSGS)

XSGS scheme [9] is composed of algorithms and protocols. We briefly review their scheme.

Setup(1^ℓ): The input is a security parameter ℓ. Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be groups of prime order p in bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We assume that we have a computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. Note that choice of all parameters depend on the input security parameter.

- *GM (issuer)* does the following:
  1) Generate prime p, pairing group $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e$, and a hash $\mathcal{H} : \{0, 1\}^* \to \mathbb{Z}_p$.
  2) Select a generator $G_2 \in \mathbb{G}_2$ at random, and set $G_1 \leftarrow \psi(G_2)$.
  3) Select $K \in \mathbb{G}_1$ and $W \in \mathbb{G}_2$.
  4) Choose $\gamma \in \mathbb{Z}_p^*$ at random, and set $\mathtt{ik} = \gamma$.
- *Opener* does the following:
  1) Choose $\xi_1, \xi_2 \in \mathbb{Z}_p$ at random.
  2) Set $H = \xi_1 K$ and $G = \xi_2 K$.

The group public key (or public parameters) `gpk` of the system, the group manager's secret key `ik`, and the opener's secret key `ok` are then given by

$$\begin{aligned} \mathtt{gpk} &= (G_1, K, H, G, G_2, W) \\ \mathtt{ik} &= \gamma \quad \text{which is the group manager's issuing key} \\ \mathtt{ok} &= (\xi_1, \xi_2) \quad \text{which is the opener's opening key.} \end{aligned}$$

User $(\mathsf{upk}, \mathsf{usk})$		GM $(\mathsf{ik} = \gamma, \mathsf{gmsk})$
$y \in \mathbb{Z}_p, C = yH$ $U = \begin{pmatrix} c = \text{Ext-Commit}(y) \\ \text{NIZKPEqDL}(c, C, H) \end{pmatrix}$	$\xrightarrow{C,U}$	Verify $C \in \mathbb{G}_1$, Checks U $x \in \mathbb{Z}_p$, $A = (\frac{1}{\gamma+x})(G_1 + C)$ $B = e(G_1 + C, G_2)/e(A, W)$ $D = e(A, G_2)$
	$\xleftarrow{A,V}$	$V = \text{NIZKPoKDL}(B, D)$
$B = e(G_1 + C, G_2)/e(A, W)$ $D = e(A, G_2)$ Verifies $A \in \mathbb{G}_1$, Checks V $S = Sign_{\mathsf{usk}}(A)$	$\xrightarrow{S}$	Check S w.r.t (upk, A)
	$\xleftarrow{x}$	Adds (upk, A, x, S) in reg.
Checks $(x + \gamma)A \overset{?}{=} G_2 + yH$ ie., $e(A, G_2)^x \cdot e(A, W) \cdot e(H, G_2)^{-y} \overset{?}{=} e(G_1, G_2)$		

reg: GM manages this registration table, meaning the user is a group member.
Ext-Commit(y): extractable commitment (c is commitment of y).
NIZKPEqDL(c,C,H): proof of equality of DL of C in base H with the committed val. in c.
NIZKPoKDL(B,D): proof of knowledge of DL of B in base D.
gmsk: some secret information used in Ext-Commit, NIZKPEqDL, and NIZKPoKDL.

Figure 1: A user performs the protocol with the key issuer

The public key, gpk, also implicitly include λ, κ, m, and a description of $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$.

Join Protocol. We assume that there is a PKI environment, and the PKI is separated from the group environment. The certification authority will be assumed fully trusted (the only one).

We also assume that each user $\mathcal{U}_i$, before joining the group, obtains a personal public key $\mathsf{upk}[i]$ and the associated secret key $\mathsf{usk}[i]$ in the PKI.

To get a membership certificate, a user performs the protocol with the key issuer (namely, Issuer or GM) shown in Figure 1.

At the end of the protocol, the user becomes a group member and obtains a membership certificate, which is the group signing key $\mathsf{gsk}= (A, x, y)$. That is, gsk=membership certificate = (A, x, y). We note that (1) (A, x) is known to GM and user, (2) y is known to the user only.

Sign(gpk,gsk[i],M). A signer who possesses a certificate (group signing key) gsk$[i]$ = (A, x, y) to sign on a message $M \in \{0, 1\}^*$ as follows:

1) Compute (T_1, T_2, T_3, T_4) as follows:

   a. Choose α, β.

   b. Set $(T_1, T_2, T_3, T_4) = (\alpha K, A+\alpha H, \beta K, A+\beta G)$.

2) Select $r_\alpha, r_\beta, r_x, r_z = \mathbb{Z}_p^*$ at random.

3) Compute

$$\begin{aligned} & R_1 = r_\alpha K, \\ & R_2 = e(T_2, G_2)^{r_x} \cdot e(H, W)^{-r_\alpha} \cdot e(H, G_2)^{-r_z}, \\ & R_3 = r_\beta K, \\ & R_4 = r_\alpha H - r_\beta G. \end{aligned}$$

4) Compute $c = \mathcal{H}(M, T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4)$.

5) Compute

$$\begin{aligned} s_\alpha &= r_\alpha + c\alpha \bmod p, & s_\beta &= r_\beta + c\beta \bmod p, \\ s_x &= r_x + cx \bmod p, & s_z &= r_z + cz \bmod p. \end{aligned}$$

The signature on message M is $\sigma = (T_1, T_2, T_3, T_4, c, s_\alpha, s_\beta, s_x, s_z)$.

Verify(gpk,M,σ). To verify a signature $\sigma = (T_1, T_2, T_3, T_4, c, s_\alpha, s_\beta, s_x, s_z)$ signed on message M, one performs the following:

1) Compute

$$\begin{aligned} R_1 &= s_\alpha K - cT_1, \\ R_2 &= e(T_2, G_2)^{s_x} \cdot e(H, W)^{-s_\alpha} \\ &\quad \cdot e(H, G_2)^{-s_z} \cdot \left(\frac{e(G_1, G_2)}{e(T_2, W)} \right)^{-c}, \\ R_3 &= s_\beta K - cT_3, \\ R_4 &= s_\alpha H - s_\beta G - c(T_2 - T_4). \end{aligned}$$

2) Check $c \overset{?}{=} \mathcal{H}(M, T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4)$.

Open(gpk,ok, (M,σ)). To trace the actual signer of a given signature σ, the open manager OM does the following:

1) Recover A: using the opening key $\mathsf{ok}=(\xi_1,\xi_2)$,

$$T_2 - \xi_1 T_1 = (A+\alpha H) - \xi_1 \alpha K = A + \alpha H - \alpha H = A,$$
$$T_4 - \xi_2 T_3 = (A+\beta H) - \xi_2 \beta K = A + \beta H - \beta H = A.$$

2) Find the actual signer, using the read-access to the registration table reg.

3) Provide a publicly verifiable proof τ that
   - he did well in the step 1 - which is a simple proof of equality of discrete logarithms in $\mathbb{G}_1$.
   - the designated user has not be framed, using the corresponding $S = Sign_{\mathsf{usk}}(A)$ in reg.

3 A Batch Verification Method for XSGS Scheme

Now we describe the batch verification of XSGS Scheme. First of all, we observe that the XSGS scheme was derived from a signature of knowledge on the underlying zero-knowledge protocol of knowledge. To shorten the lengths of the signature, the authors slightly modified the signature of knowledge instead of using the (standard) signature of knowledge. In order to efficiently batch the verification, we modify the signature and the verification algorithms without comprise the security of the scheme. We note that the technique in this section is almost the same as Ferrara et al.'s method used for batch BBS group signature scheme.

New-Sign(gpk,gsk[i],M).

A signer who possesses a certificate gsk$[i] = (A,x,y)$ to sign on a message $M \in \{0,1\}^*$ as follows:

1) Compute (T_1,T_2,T_3,T_4) from $DELG(A)$:
   a. Choose α,β.
   b. Set $(T_1,T_2,T_3,T_4) = (\alpha K, A+\alpha H, \beta K, A+\beta G)$.
2) Select $r_\alpha, r_\beta, r_x, r_z \in \mathbb{Z}_p^*$ at random.
3) Compute

$$\begin{aligned}
&R_1 = r_\alpha K,\\
&R_2 = e(T_2,G_2)^{r_x}\cdot e(H,W)^{-r_\alpha}\cdot e(H,G_2)^{-r_z},\\
&R_3 = r_\beta K,\\
&R_4 = r_\alpha H - r_\beta G.
\end{aligned}$$

4) Compute $c = \mathcal{H}(M, T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4)$.
5) Compute

$$s_\alpha = r_\alpha + c\alpha \bmod p, \quad s_\beta = r_\beta + c\beta \bmod p,$$
$$s_x = r_x + cx \bmod p, \quad s_z = r_z + cz \bmod p.$$

The signature is $\sigma = (T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4, s_\alpha, s_\beta, s_x, s_z)$.

New-Verify(gpk,M,σ). To verify a signature σ of a message M with σ=$(T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4, s_\alpha, s_\beta, s_x, s_z)$, one performs the following:

1) Compute $c = \mathcal{H}(M, T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4)$.
2) Check that

$$\begin{aligned}
R_1 &\overset{?}{=} s_\alpha K - cT_1,\\
R_3 &\overset{?}{=} s_\beta K - cT_3,\\
R_4 &\overset{?}{=} s_\alpha H - s_\beta G - c(T_2 - T_4).
\end{aligned}$$

3) Check

$$e(T_2,G_2)^{s_x}\cdot e(H,W)^{-s_\alpha}\cdot e(H,G_2)^{-s_z} \overset{?}{=} R_2 \cdot \left(\frac{e(G_1,G_2)}{e(T_2,W)}\right)^c.$$

As previously mentioned, these modified algorithms are from direct application of Fiat-Shamir transformation on the zero-knowledge proof of knowledge in [9]. The security proofs are the same as in the paper, because XSGS scheme was, in fact, proved in this setting using the fact that R_i can derived from T_i, c and $s_\alpha, s_\beta, s_x, s_z$. Thus, we conclude this modified version is also a secure group signature scheme.

3.1 A Batch Verifier of XSGS Scheme

Let $\mathsf{gpk} = (\mathbb{G}_1, \mathbb{G}_2, G_T, e, \psi, G_1, K, H = \xi_1 K, G = \xi_2 K, G_2, W = \gamma G_2)$ be the group public key, and let σ_j be the j'th signature on message M_j for each $j = 1,\ldots,N$, where $\sigma_j = (T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, R_{j,1}, R_{j,2}, R_{j,3}, R_{j,4}, s_{j,\alpha}, s_{j,\beta}, s_{j,x}, s_{j,z})$. Then to batch the signatures, our XSGS Batch does the following:

1) Compute for all $j = 1,\ldots,N$

$$c_j = \mathcal{H}(M, T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, R_{j,1}, R_{j,2}, R_{j,3}, R_{j,4}).$$

2) Check, for each $j = 1,\ldots,N$, whether the following non-pairing equations are satisfied or not:

$$\begin{aligned}
s_{j,\alpha}K &\overset{?}{=} R_{j,1} + c_j T_{j,1},\\
s_{j,\beta}K &\overset{?}{=} R_{j,3} + c_j T_{j,3},\\
s_{j,\alpha}H - s_{j,\beta}G &\overset{?}{=} R_{j,4} + c_j(T_{j,2} - T_{j,4}).
\end{aligned}$$

If the equations are satisfied then go to next; Otherwise, return 0 and exit.

3) Choose a random vector $(\delta_1, \ldots, \delta_\ell)$ of ℓ_b bit elements from $\mathbb{Z}_p$.

4) Check that

$$\prod_{j=1}^{N} R_{j,2}^{\delta_j} \overset{?}{=} e\left(\sum_{j=1}^{N} \delta_j \left(s_{j,\alpha}T_{j,2} - s_{j,z}H - c_jG_1\right), G_2\right) \cdot e\left(\sum_{j=1}^{N} \delta_j \left(-s_{j,\alpha}H + c_jT_{j,2}\right), W\right)$$

If this is satisfied then return 1 and exit. Otherwise, return 0 and exit.

Theorem 1. *For security level ℓ_b, the above algorithm is a batch verifier for the XSGS group signature scheme, where the probability of accepting an invalid signature is 2^{ℓ_b}.*

Proof. Since the algorithm XSGS Batch performs the first three non-pairing tests in $\mathbb{G}_T$ separately, it suffice to consider for the pairing test.

Now, suppose that is N valid signatures corresponding messages M_j to be batched where $\sigma_j = (T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, R_{j,1}, R_{j,2}, R_{j,3}, R_{j,4}, s_{j,\alpha}, s_{j,\beta}, s_{j,x}, s_{j,z})$. Then

$$e(T_{j,2}, G_2)^{s_{j,x}} \cdot e(H, W)^{-s_{j,\alpha}} \cdot e(H, G_2)^{-s_{j,z}} = R_{j,2} \cdot \left(\frac{e(G_1, G_2)}{e(T_{j,2}, W)}\right)^{c_j},$$

and so for any random vector $(\delta_1, \ldots, \delta_N)$ of ℓ_b bit elements from $\mathbb{Z}_q$, we get

$$\left(e(T_{j,2}, G_2)^{s_{j,x}} \cdot e(H, W)^{-s_{j,\alpha}} \cdot e(H, G_2)^{-s_{j,z}}\right)^{\delta_j} = \left(R_{j,2} \cdot \left(\frac{e(G_1, G_2)}{e(T_{j,2}, W)}\right)^{c_j}\right)^{\delta_j}.$$

That is,

$$R_{j,2}^{\delta_j} = \left(e(T_{j,2}, G_2)^{s_{j,x}} \cdot e(H, W)^{-s_{j,\alpha}} \cdot e(H, G_2)^{-s_{j,z}}\right)^{\delta_j} \cdot e(T_{j,2}, W)^{c_j\delta_j} \cdot e(G_1, G_2)^{-c_j\delta_j}$$

Then we combine and simplify as

$$e\left(\delta_j \left(s_{j,\alpha}T_{j,2} - s_{j,z}H - c_jG_1\right), G_2\right) \cdot e\left(\delta_j \left(-s_{j,\alpha}H + c_jT_{j,2}\right), W\right) = R_{j,2}^{\delta_j}.$$

Multiplying every N equations, we finally have

$$\prod_{j=1}^{N} R_{j,2}^{\delta_j} = e\left(\sum_{j=1}^{N} \delta_j \left(s_{j,\alpha}T_{j,2} - s_{j,z}H - c_jG_1\right), G_2\right) \cdot e\left(\sum_{j=1}^{N} \delta_j \left(-s_{j,\alpha}H + c_jT_{j,2}\right), W\right)$$

Thus XSGS Batch $= 1$ if input signatures are all valid.

Now, we show that XSGS Batch $= 1$ implies that there is no invalid signature in the input signatures except with negligible probability. Suppose that $\sigma_j = (T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, R_{j,1}, R_{j,2}, R_{j,3}, R_{j,4}, s_{j,\alpha}, s_{j,\beta}, s_{j,x}, s_{j,z})$ are N signatures such that XSGS Batch $= 1$. Then

$$\prod_{j=1}^{N} R_{j,2}^{\delta_j} = e\left(\sum_{j=1}^{N} \delta_j \left(s_{j,\alpha}T_{j,2} - s_{j,z}H - c_jG_1\right), G_2\right) \cdot e\left(\sum_{j=1}^{N} \delta_j \left(-s_{j,\alpha}H + c_jT_{j,2}\right), W\right)$$

It can be rewritten as

$$\prod_{j=1}^{N} \left(\underbrace{E_1 \cdot E_2 \cdot R_{j,2}^{-1}}\right)^{\delta_j} = 1.$$

$$E_1 = e\left(s_{j,\alpha}T_{j,2} - s_{j,z}H - c_jG_1, G_2\right);$$
$$E_2 = e\left(-s_{j,\alpha}H + c_jT_{j,2}, W\right).$$

In the above equation, the braced element can be expressed as $e(G_1, G_2)^{\beta_j}$ since it is in G_T, and so $\prod_{j=1}^{N} e(G_1, G_2)^{\beta_j\delta_j} = 1$. On the other hand, since $e(G_1, G_2)$ is a generator of G_T, we have $e(G_1, G_2)^{\sum_{j=1}^{N} \beta_j\delta_j} = 1$ and $\sum_{j=1}^{N} \beta_j\delta_j \equiv 0 \pmod p$.

Now assume that there is at least one invalid signature, say σ_1. Then, $\beta_1 \neq 0$ because, if not then

$$e\left(s_{1,\alpha}T_{1,2} - s_{1,z}H - c_1G_1, G_2\right) \cdot e\left(-s_{1,\alpha}H + c_1T_{1,2}, W\right) \cdot R_{1,2}^{-1},$$

and this means

$$e(T_2, G_2)^{s_{1,x}} \cdot e(H, W)^{-s_{1,\alpha}} \cdot e(H, G_2)^{-s_{1,z}} = R_{1,2} \cdot \left(\frac{e(G_1, G_2)}{e(T_2, W)}\right)^{c_1}.$$

That is, σ_1 is a valid signature, which leads to a contradiction.

Now since $\beta_1 \not\equiv 1 \pmod p$, $\beta_1\gamma_1 \equiv 1 \pmod p$ for some γ_1. Hence we have an equation

$$\delta_1 \equiv -\gamma_1 \sum_{j=2}^{N} \beta_j\delta_j \pmod p.$$

Let E be an event that occurs if New-Verify$(\sigma_1) = 0$ but XSGS Batch$(\sigma_1, \ldots, \sigma_N) = 1$, and let $\widehat{X_1}$ be denote the set of all vectors $< \delta_2, \ldots, \delta_N >$ consisted with the last $N-1$ values of Δ in XSGS Batch. Then $\Pr[E|\Delta_1] = 2^{\ell_b}$ for each $\Delta_1 \in \widehat{X_1}$. So we have

$$\begin{aligned}\Pr[E] &\leqslant \sum_{\Delta_1 \in \widehat{X_1}} \Pr[E|\Delta_1] \Pr[\Delta_1] \\ &= \sum_{i=1}^{2^{\ell_b(N-1)}} 2^{\ell_b} 2^{-\ell_b(N-1)} \\ &= 2^{-\ell_b}.\end{aligned}$$

□

4 Finding Invalid Signatures in XSGS Scheme

In this section, we show how to find invalid signatures in XSGS group signature when batch instance contains bad signatures. The technique we use is due to Law and Matt [15], or Lee et al. [16]. While Lee et al.'s method is to find bad signatures in RSA signature scheme, Law and Matt's methods are to find invalid signatures in pairing based standard signature schemes in which the verifications are of the form $e(X_i, P) = e(Y_i, R)$. Our method can be thought of slight extension of the Law et al.'s exponentiation method [15].

Suppose that we have an instance of N signatures σ_j on messages M_j where $\sigma_j = (T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, R_{j,1}, R_{j,2}, R_{j,3}, R_{j,4}, s_{j,\alpha}, s_{j,\beta}, s_{j,x}, s_{j,z})$, $j = 1,\ldots, N$, and that the instance contains w invalid ones. To find the invalid signatures, we first compute, for each $j = 1, \ldots, N$, $c_j \leftarrow \mathcal{H}\ (M_j, T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, R_{j,1}, R_{j,2}, R_{j,3}, R_{j,4})$. We then verify the validity of the non-pairing equations:

$$\begin{aligned} s_{j,\alpha}K &\overset{?}{=} R_{j,1} + c_jT_{j,1}, \quad s_{j,\beta}K \\ &\overset{?}{=} R_{j,3} + c_jT_{j,3}, s_{j,\alpha}H - s_{j,\beta}G \\ &\overset{?}{=} R_{j,4} + c_j(T_{j,2} - T_{j,4}). \end{aligned}$$

All j for which the equation is not satisfied indicate the positions of invalid signatures. Now, excluding the bad ones, we assume that there is no such index, i.e., for all $j = 1, \ldots, N$ the signatures hold the non-pairing equations. Let $j_1, \ldots, j_w$ denote the indexes of the invalid signature and I be the set $\{j_t \mid t = 1, \ldots, w\}$. We will find the set I consisting of bad signatures. For each $j = 1, \ldots, N$, we choose $\delta_j \in \mathbb{Z}_p^*$ and set

$$\begin{aligned} X_j &\leftarrow \delta_j\,(s_{j,\alpha}T_{j,2} - s_{j,z}H - c_jG_1)\,, \\ Y_j &\leftarrow \delta_j\,(-s_{j,\alpha}H + c_jT_{j,2})\,, \\ Z_j &\leftarrow R_{j,3}^{\delta_j}. \end{aligned}$$

We define $A_k := \prod_{j=1}^{N} e(j^kX_j, U) \cdot e(j^kY_j, V) \cdot Z_j^{j^k}$ for each $k = 0, \ldots, w$. If $A_0 = 1$ then this means there is no bad signatures, i.e. $w = 0$. Otherwise, the instance should have invalid signature(s) and $w \geqslant 1$. Also for each $k = 0, \ldots, w$,

$$\begin{aligned} A_k &= \prod_{j=1}^{N} \left(e(X_j, U) \cdot e(Y_j, V) \cdot Z_j\right)^{j^k} \\ &= \prod_{j \in I} \left(e(X_j, U) \cdot e(Y_j, V) \cdot Z_j\right)^{j^k}. \end{aligned}$$

By Newton formula [17],

$$A_w = A_{w-1}^{f_1} \cdot A \cdots A_1^{(-1)^{w-2}f_{w-1}} \cdot A_0^{(-1)^{w-1}f_w}, \quad (1)$$

where $f_i = f_i(j_1, \ldots, j_w)$ is ith elementary symmetric polynomial over field $\mathbb{Z}_p$ in w variables $j_1, \ldots, j_w$. Thus, if we compute $f_1, \ldots, f_w$ (without knowing the positions of bad ones) for which Equation (1) holds then we have found the positions of invalid signatures. If no match is found, then there are more bad signatures than w in the batch instance.

Algorithm. (Batch Identifier)

```
1) Compute $c_j \leftarrow \mathcal{H}(M_j, T_{j,1}, T_{j,2}, T_{j,3}, R_{j,1}, R_{j,2}, R_{j,3}, R_{j,4})$ for each $j$.
2) For $j = 1$ to $N$ do
    a. Check the following non-pairing equations EQ(j):
           $s_{j,\alpha}K \overset{?}{=} R_{j,1} + c_jT_{j,1}$,
           $s_{j,\beta}K \overset{?}{=} R_{j,3} + c_jT_{j,3}$,
           $s_{j,\alpha}H - s_{j,\beta}G \overset{?}{=} R_{j,4} + c_j(T_{j,2} - T_{j,4})$.
    b. If EQT(j) fails then return $j$. (In this step, we will discard the invalid signatures. So, in what follows, we assume that the input signatures pass this step.)
3) Choose a random vector $(\delta_1, \ldots, \delta_\ell)$ of $\ell_b$ bit elements from $\mathbb{Z}_p$.
4) For $j = 1$ to $N$ do
    a. Compute $X_j \leftarrow \delta_j\,(s_{j,\alpha}T_{j,2} - s_{j,z}H - c_jG_1)$,
    b. Compute $Y_j \leftarrow \delta_j\,(-s_{j,\alpha}H + c_jT_{j,2})$,
    c. Compute $Z_j \leftarrow R_{j,3}^{\delta_j}$.
5) Compute $U \leftarrow -G_2$ and $V \leftarrow -W$.
6) Compute $A_0 \leftarrow e(\sum_{j=1}^{N} X_j, U) \cdot e(\sum_{j=1}^{N} Y_j, V) \cdot \prod_{j=1}^{N} Z_j$.
7) If $A_0 = 1$ then output "valid instance", and exit.
8) Set $k \leftarrow 1$.
9) While $k < N$ do
    a. Compute
           $A_k = e(\sum_{j=1}^{N} j^kX_j, U) \cdot e(\sum_{j=1}^{N} j^kY_j, V) \cdot \prod_{j=1}^{N} Z_j^{j^k}$
    b. Find $j_1, \ldots, j_k$ such that
           $A_k = \prod_{t=1}^{k} A_{k-t}^{(-1)^{t-1}f_t}$,
       where $f_t$ is $i$th elementary symmetric polynomial in $j_1, \ldots, j_k$
    c. If such $j_1, \ldots, j_k$ exist then output $j_1, \ldots, j_k$. as the position of invalid signatures and exit.
    d. Otherwise, set $k \leftarrow k + 1$.
10) Output"all the input signatures are invalid" and exit.
```

4.1 Efficiency Analysis

Following earlier batch identification methods [15, 16, 22], we assume that the number of invalid signatures w is small. The analysis in this subsection is similar to that of Law and Matt [15].

As well as our method, Individual test and Quick Binary test cannot avoid the non-pairing computations of Step 2. Also note that all of the compared methods will be launched when initial batch verification fails and Step 1-6 are batch verification steps. So, to simplify the cost comparison, we don't count the cost of non-pairing equation tests and Steps 1-6, and assume that w is the number of invalid signatures contained in the batch instance (of course w is unknown).

In Step 9, we first need to compute $\sum_{j=1}^{N} j^k X_j$ and $\sum_{j=1}^{N} j^k Y_j$. By Law and Matt's method [15], due to Solinas, we know that $\sum_{j=1}^{N} j^k X_j (k = 1, \ldots, w)$ can be computed during $w(N-1)$ addition in $\mathbb{G}_1$, and so $\sum_{j=1}^{N} j^k X_j$ and $\sum_{j=1}^{N} j^k Y_j (k = 1, \ldots, w)$ can be computed in $2w(N-1)$ addition in $\mathbb{G}_1$. Similarly, we can compute $\prod_{j=1}^{N} Z_j^{j^k} (k = 1, \ldots, w)$ in $w(N-1)$ multiplications in $\mathbb{G}_2$ which we will see in theorem 2 below. Additionally, to compute A_k $(k = 1, \ldots, w)$ we require $2w$ pairings and $2w$ multiplication in $\mathbb{G}_T$.

Using Shanks' Baby-step Giant-step and Law et al.'s method, the step 9-(b) (for all iteration up to w) can be done during $2\sqrt{N}$ or $\frac{8}{(w-1)!} N^{w-1} + O(N^{w-2})$ multiplications in $\mathbb{G}_T$ according to $w = 1$ or $w \geqslant 2$, respectively. For $w \geqslant 2$, we require $w-1$ inverses in $\mathbb{G}_T$ additionally. (The computation 9-(b) is exactly the same with Law et al.'s algorithm).

Theorem 2. *$\prod_{j=1}^{N} Z_j^{j^k} (k = 1, \ldots, w)$ in $w(N-1)$ multiplications in $\mathbb{G}_2$.*

Proof. First, by Stirling formula, we know that for $j = 1, \ldots, N$.

$$j^k = \sum_{i=1}^{k} S_{k,i}\ (j)_i = \sum_{i=1}^{k} (-1)^{k-i} i!\ S_{k,i} \left(\begin{array}{c} j+i-1 \\ i \end{array} \right)$$

where $S_{k,i}$ is the Stirling number of the second kind, $(j)_i = j \cdot (j-1) \cdots (j-i+1)$, and $k \geqslant 1$ is an integer. Thus, we have

$$\begin{aligned} \prod_{j=1}^{N} Z_j^{j^k} &= \prod_{j=1}^{N} Z_j^{\sum_{i=1}^{k} (-1)^{k-i} i!\ S_{k,i} \left(\begin{array}{c} i+j-1 \\ i \end{array} \right)} \\ &= \prod_{j=1}^{N} \prod_{i=1}^{k} Z_j^{\left(\begin{array}{c} i+j-1 \\ i \end{array} \right) \left((-1)^{k-i} i! S_{k,i}\right)} \\ &= \prod_{i=1}^{k} \left(\prod_{j=1}^{N} Z_i^{\left(\begin{array}{c} i+j-1 \\ i \end{array} \right)} \right)^{(-1)^{k-i} i! S_{k,i}} \end{aligned}$$

Letting $U_i = \prod_{j=1}^{N} Z_j^{\left(\begin{array}{c} i+j+1 \\ i \end{array} \right)}$, we have

$$\prod_{j=1}^{N} Z_k^{j^k} = \prod_{i=1}^{k} U_i^{(-1)^{k-i} i!\ S_{k,i}}$$

In order to compute U_i, consider

$$\begin{aligned} U_i &= \prod_{j=1}^{N} Z_j^{\binom{i+j-1}{i}} = Z_1^{\binom{i}{i}} \cdot Z_2^{\binom{i+1}{i}} \cdots Z_N^{\binom{N+i-1}{i}} \\ &= Z_1^{\binom{i}{i}} \cdot \\ &\quad Z_2^{\binom{i}{i-1}} \cdot Z_2^{\binom{i-1}{i-1}} \cdot \\ &\quad \vdots \qquad \vdots \\ &\quad Z_N^{\binom{N+i-2}{N-1}} \cdot Z_N^{\binom{N+i-3}{N-1}} \cdot Z_N^{\binom{Z+i-4}{N-1}} \cdots Z_N^{\binom{N-1}{N-1}} \\ &= Z_N^{\binom{N-1}{N-1}} \cdot \left(Z_{N-1}^{\binom{N-1}{N-1}} \cdot Z_N^{\binom{N}{N-1}} \right) \cdots \\ &\qquad \left(Z_1^{\binom{i}{i}} \cdot Z_2^{\binom{i}{i-1}} \cdots Z_N^{\binom{N+i-2}{N-1}} \right) \end{aligned}$$

Thus, by defining $U_0 = \prod_{j=1}^{N} Z_j$, we can compute these w values $U_1, \ldots, U_w$ as follows.

```
1) For $j$ from 1 to $N$ do
       $V_j \leftarrow Z_j$
2) For $k = 0$ to $w$
       For $j$ from $N-1$ down to 1 do
           $V_j \leftarrow V_j \cdot V_{j+1}$
           $j \leftarrow j-1$
       $U_k \leftarrow V_1$
   $k \leftarrow k+1$
```

The computations for $k = 0$ can be ignored because the values were already computed during Step 6 of the Batch Identifier. So the cost of this algorithm is $w(N-1)$ multiplications in $\mathbb{G}_T$.

Finally, we compute $prod_k := \prod_{j=1}^{N} Z_j^{j^k}$ for $k = 1, \cdots, w$ as follows.

```
1) $prod_1 \leftarrow U_1$;  $s_0 \leftarrow 0$;  $s_1 \leftarrow 1$;  $s_2 = \cdots s_w \leftarrow 0$
2) For $k = 2$ to $w$
       $prod_k \leftarrow 1$
       For $m = k$ down to 1 do
           $s_m \leftarrow s_{m-1} + m s_m$
           $prod_k \leftarrow \left( U_m^{(-1)^{k-m} s_m} \cdot prod_m \right)^m$
           $m \leftarrow m-1$
       $k \leftarrow k+1$
```

Note that this second part does not depend on N, but only on the number w of invalid signatures contained in batch. Also note that the algorithm runs with only $O(w^2)$ operations. Since, as most batch verifier or batch identifier, we assume that N is large and w is small, we can ignore the cost $O(w^2)$ operation. Therefore, we conclude that the cost of computing $\prod_{j=1}^{N} Z_j^{j^k}$ $(k = 1, \ldots, w)$ is approximately $w(N-1)$ multiplications in $\mathbb{G}_T$. $\square$

Thus, we have approximate cost of the algorithm to find w invalid signatures in a batch of N signature - $2w(N-1)$ addition in $\mathbb{G}_1$, $2w$ pairings, N exponentiations in $\mathbb{G}_T$, $w-1$ inverses in $\mathbb{G}_T$, and $N+1+2\sqrt{N}$ multiplications in $\mathbb{G}_T$ if $w=1$ or $w(N+1)+\frac{8}{(w-1)!}N^{w-1}$ multiplications if $w \geqslant 2$.

Table 1 summarizes cost comparison of our method with other batch identification methods where we use our batch verifier as the underlying batch verification algorithm of Quick binary.

The notions mult and mult-exp denote multiplications and multi-exponentiations, respectively. In the Individual test, to reduce the number of pairings, we use the equation

$$e(T_2, s_x G_2 - cW)\cdot e(H,W)^{-s_\alpha}\cdot e(H,G_2)^{-s_z}\cdot e(G_1,G_2)^{-c} \stackrel{?}{=} R_2.$$

By cashing the $e(H,W)$, $e(H,G_2)$ and $e(G_1,G_2)$, one can verify the equation during 1 multi-exponentiation with two exponents in $\mathbb{G}_2$, 1 multi-exponentiation in $\mathbb{G}_T$, 1 pairing, and 1 multiplications in $\mathbb{G}_T$. We note that this property is mentioned in the paper [9]. Table 1 shows our method reduces the pairing computations significantly.

To evaluate concrete performance, we compare the timings under the following settings [2, 10, 15]. Let p be a prime of 160 bits. We assume that E be a non-supersingular elliptic curve defined over F_p with embedding degree 6 for which Tate pairing is constructed (ie., MNT-curve or Class A curve [12, 15]). Then from Barreto et al. [2] and Avanzi [1], we first estimate each 1-operation in terms of the number of multiplications of F_p as summarized in Table 2.

Table 2: The number of multiplications in F_p

pairing	add in $\mathbb{G}_1$	multi-add in $\mathbb{G}_2$
$9120m$	$11m$	$9253m$
mult in $\mathbb{G}_T$	multi-exp in $\mathbb{G}_T$	inv in $\mathbb{G}_T$
$15m$	$1839m$	$44m$

In Table 2, m denote the number of multiplications in the prime field F_p and the notions mult, multi-exp, and inv denote multiplications, multi-exponentiations, and inverses, respectively. For the multi-exponentiation, we use the Joint Sparse Form like Matt et al. did.

Plugging Table 2 to Table 1, we can approximate timing results in Figure 2. From the results, we can observe that our extended method shows better performance than

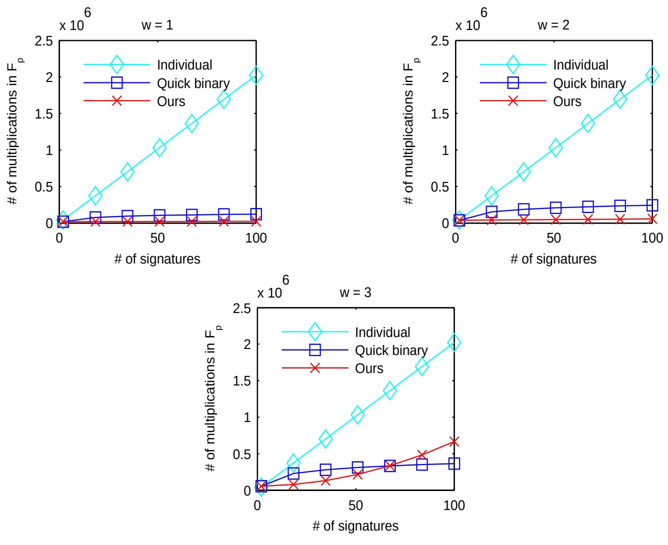

Figure 2: Cost comparison to other methods

Individual test and Quick Binary in case when the underlying batch verifier is the XSGS Batch proposed in the previous section.

In practical situations and literatures, a set of signatures to be batched contain very small number of invalid signatures. By shuffling and then partitioning into several blocks, we may assume that the number of signatures are not too large (e.g., 1024) and that the number of bad signatures contained in each partition is at most very small (e.g., 0, 1 or 2).

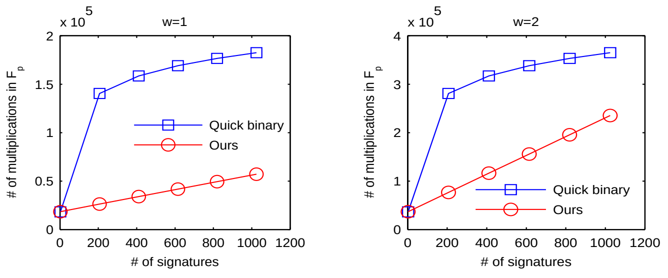

Figure 3: Comparison to Quick Binary method

Figure 3 shows the performance comparison between Quick binary and our method in terms of the number of multiplications when the number of signatures is less than or equal 1024 and the number of invalid signatures contained in is 1 or 2, respectively. From the results, we can conclude that our method is reduces considerably the computational cost compared with Individual test and Quick binary.

5 Conclusion

In group oriented signature schemes, there are a large number of signatures to be verified. For such signature

Table 1: Comparison to other methods

	Individual	Quick Binary	Our Method
pairing	N	$2w\log_2 N$	$2w$
addition in $\mathbb{G}_1$	-	-	$2w(N-1)$
mult in $\mathbb{G}_T$	N	$w\log_2 N$	$w(N+1)+2\sqrt{N}$ if $w=1$ $w(N+1)+\frac{8}{(w-1)!}N^{w-1}$ if $w \geqslant 2$
multi-add in $\mathbb{G}_2$	N	-	-
multi-exp in $\mathbb{G}_T$	N	-	-
inverse in $\mathbb{G}_T$	-	-	$w-1$

schemes, designing batch verification methods and finding invalid signatures (when batch test fails) are important issues.

In this paper, by exploiting Ferrera *et al.'s* batch verifier of short signature scheme, we have proposed a batch verification scheme of an extremely short dynamic group signature scheme. In addition, by extending a batch identifier of Law et al. designed for special types of signature scheme (not for group signature schemes), we presented an batch identification method of the extremely short group signature scheme.

Acknowledgements

This work was supported by Priority Research Centers Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (2009-0093827).

References

[1] R. Avanzi, "On the complexity of certain multi-exponentiation techniques in cryptography," In *Journal of Cryptology*, vol. 18, no. 4, *Spinger-Verlag*, pp. 357-373, 2005.

[2] P. Barreto, H. Kim, B. Lynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems," *Crypto'02*, LNCS 2442, *Springer-Verlag,* pp. 354-368, 2002.

[3] M. Bellare, J. Garay, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures," *Eurocrypt'98*, LNCS 1403, *Springer-Verlag*, pp. 236-250, 1998.

[4] D.Boneh, X.Boyen, and H.Shacham, "Short group signatures," *Crpyto'04*, LNCS 3152, pp. 41-55, 2004.

[5] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *Asiacrypt'2001*, LNCS 2248, pp. 514-532, 2001.

[6] C.Boyd, and C. Pavlovski, "Attacking and repairing batch verification schemes," *Asiacrypt'00*, LNCS 1976, *Springer-Verlag*, pp. 58-71, 2000.

[7] J. Camenisch, S. Hohenberger, and M. Pedersen, "Batch verification of short signatures," *Eurocrypt'07*, LNCS 4515, *Springer-Verlag*, pp. 246-263, 2007.

[8] J. Cha, and J. Cheon, "An identity-based signature from gap diffie-hellman groups," *PKC'03*, LNCS 2567, *Springer-Verlag*, pp. 18-30, 2003.

[9] C. Delerablee, and D. Pointcheval, "Dynamic fully anonymous short group signatures," *VietCrypt 2006*, LNCS 4341, *Springer-Verlag*, pp. 193-210, 2006.

[10] A. Ferrara, M. Green, S. Hohenberger, M. Pedersen, "On the Practicality of Short Signature Batch Verification," *CT-RSA 2009*. (http://eprint.arcr.org/2008/015)

[11] A. Fiat, "Batch RSA," *Crypto'89*, LNCS 435, *Springer-Verlag*, pp. 175-185, 1989.

[12] R. Granger, D. Page, and N.P. Smart, "High security pairing-based cryptography revisited," *ANTS VII*, LNCS 4075, *Springer-Verlag*, pp. 480-494, 2006.

[13] R. Gennaro, H. Krawczyk, and T. Rabin, "RSA-based undeniable signatures," *Crypto'97*, LNCS 1294, *Springer-Verlag*, pp. 132-149, 1997.

[14] M. S. Hwang, C. C. Lee, and Y. L. Tang, "Two simple batch verifying multiple digital signatures," *Proceedings of Information and Communications Security*, LNCS 2229, *Springer-Verlag*, pp. 233-237, 2001.

[15] L. Law, and B. Matt, "Finding invalid signatures in pairing-based bathes," *Cryptography and Coding 2007*, LNCS 4887, *Springer-Verlag*, pp. 34-53, 2007.

[16] S. Lee, S. Cho, J. Choi, and Y. Cho, "Efficient identification of bad signatures in RSA-Type batch signature," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E89-A, no. 1, pp. 74-80, 2006.

[17] Lidl, and Niderriter, *Finite Fields*, Encyclotopedia of Mathematics and its Applications, Cambridge University Press, Cambridge, UK, 2nd ed., 1997.

[18] B. Matt, "Identification of multiple invalid signatures in pairing-based batched signatures," *Advances in Cryptology - PKC 2009*, LNCS, *Springer-Verlag*, 2009.

[19] Naccache, M'Raihi, Vaudenay, and Raphaeli, "Can DSA be improved? complexity trade-offs with the digital signature standard," In *EUROCRYPT'94*, LNCS 0950, pp. 77-85, 1994.

[20] L. Nguyen, and R. Safavi-Naini, *Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings*, *Asiacrypt' 2004*, LNCS 3329, *Springer-Verlag*, pp. 372-386, 2004.

[21] H. Park, S. Lim, I. Yie, K. Kim, and J. Song, "Strong unforgeability in group signature schemes", *Computer Standards & Interfaces*, vol. 31, pp. 856-862, Elsevier, 2009.

[22] J. Pastuszak, D. Michalek, J. Pieprzyk, and J. Seberry, "Identification of bad signatures in batches", *PKC 2000*, *LNCS 1751*, *Springer-Verlag*, pp. 28-45, 2004.

[23] M. Stanek, "Attacking LCCC batch verification of RSA signatures", *International Journal of Network Security*, vol. 6, no. 3, pp. 255-257, 2008.

Kitae Kim received the B.S. degree in Mathematics from Konyang University, and the M.S and the Ph.D degrees in Mathematics from Inha University, Korea. He is a postdoctoral researcher at Graduate School of Information Technology and Telecommunications in Inha University. His current research interests include algebraic/algorithmic number theory, elliptic curves, privacy enhanced signatures and homomorphic encryption.

Ikkwon, Yie recieved the B.S. and M.S. degrees in Mathematics from the Seoul National University, Seoul, Korea, and the Ph.D. degree in Mathematics from the Purdue University. He is currently a professor of Department of Mathematics in Inha University. His main research interests include Galois theory and Digital signatures.

Seongan Lim received her B.S. degree in Mathematics from the Dongguk University, Korea, in 1985. In 1987, she received her M.S. degree in Mathematics from the Seoul National University, Korea. In 1995, she received her PhD degree in Mathematics from Purdue University, USA. She is a research professor of Department of Mathematics in Ewha Womans University, Korea. Her current research interests include cryptography, fast computer arithmetic, computer algorithms, mathematics.

Daehun Nyang received his BS degree in electronic engineering from Korea Advanced Institute of Science and Technology (KAIST) in 1994, the MS and the PhD degrees in computer science from YONSEI University, Korea, in 1996 and 2000, respectively. From 2000 to 2003, he had worked for Electronics and Telecommunications Research Institute (ETRI) as a senior researcher. Since 2003, he has been with INHA University, Korea, where he is currently a professor in Graduate School of Information Technology and Telecommunications. His research interests include cryptography and network security including WPAN, WLAN, MANET, RFID, and WSN.