INFORMATICA, 2011, Vol. 22, No. 2, 189–201



# A Method of Finding Bad Signatures in an RSA-Type Batch Verification

Kitae KIM[1], Ikkwon YIE[1], Seongan LIM[2*], Haeryong PARK[3]
[1]*Department of Mathematics, Inha University*
*Yonghyun-dong 253, Nam-gu, Incheon, 402-751, Korea*
[2]*Department of Mathematics, Ewha Womans University*
*Daehyun-dong 11-1, Seodaemun-gu, Seoul, 120-750, Korea*
[3]*KISA, IT Venture Tower*
*Garak-dong 78, Songpa-gu, Seoul, 138-950, Korea*
*e-mail: ktkim@inha.ac.kr, ikyie@inha.ac.kr, seongannym@yahoo.co.kr, snupark@kisa.or.kr*

Received: July 2009; accepted: January 2011

**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 Lee *et al.*, Ferrara *et al.* and Law *et al.*, respectively. In this paper, we address some weakness of Lee *et al.*'s earlier work, and propose an identification method in an RSA-type signature. Our method is more efficient than the well known divide and conquer method for the signature scheme. We conclude this paper by providing a method to choose optimal divide and conquer verifiers.

**Keywords:** batch verification, batch identification, RSA-type signature.

## 1. Introduction

Currently, digital signatures have been adapted in many industrial applications such as electronic payment system, electronic voting system, etc. 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, (i) determine efficiently whether an instance contains invalid signatures; (ii) identify efficiently invalid signatures, if any, in an instance.

A batch verification of a signature scheme provides a solution of the first question. A batch verification algorithm (or a batch verifier) of signatures is defined as follows.

*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).

A batch verifier of signatures is a probabilistic algorithm that takes as input a security parameter $\ell$ and a batch instance (signature/message pairs), satisfying (i) if all the members of an instance are valid then it returns true, (ii) 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 (Fiat, 1989), and the first efficient batch verifier was proposed by Naccache *et al.* (1994) for DSA-type signatures. Since then several batch verification methods have been proposed for DSA-type, RSA-type, and pairing based systems. In particular, Ferrara *et al.* (2009) 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. Pastuszak *et al.* (2004) 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 (2006) proposed new methods $DBI_{\mathrm{basic}}$ and $DBI_{\alpha}$ for identifying bad signatures efficiently in RSA-type batch signatures. $DBI_{\mathrm{basic}}$ is aimed to find bad one in case when there is one bad signature in input instance. Stanek (2008) showed that this method was flawed and proposed an improved method to repair his attack.

In this paper, we show that a theorem used by Lee *et al.* is incorrect and Stanek's improvement is also miss-leaded. Indeed, Stanek's improved method (Lee *et al.*, 2006) does not identify bad signatures. We give attacks against the batch identification method proposed by Lee *et al.* and Stanek's improvement. Furthermore, we propose an identification method on a modified RSA signature scheme introduced by Gennaro *et al.* The proposed identification method is a batch identification for the Boyd *et al.*'s batch verification.

## 2. Preliminaries

### 2.1. *Batch Verification and Batch Identification*

A batch verification of digital signatures was introduced by Naccache *et al.* (1994) 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.* (1998). The following definition is due to Camenisch *et al.* (2007) in which they extended the definition of Bellare *et al.* to deal with multiple signers.

DEFINITION 1 (Batch verification of signatures). Let $\ell$ be the security parameter. Suppose that $(pk_1, sk_1), \ldots, (pk_n, sk_n)$ are generated independently according to $\mathsf{Gen}(1^{\ell})$. Then we call probabilistic $\mathsf{Batch}$ a batch verifier (or 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 $\ell$, taken over the randomness of $\mathsf{Batch}$.

DEFINITION 2 (Batch identification). Under the same assumption as in the above definition, a probabilistic polynomial time algorithm is secure batch identifier (or batch identification) relative to a batch verifier if it satisfies the followings: for any input instance $x$,

- if $x$ has no invalid signature then it outputs "true", meaning every signature in $x$ is valid.
- if $x$ contains invalid signatures, then it outputs the invalid ones contained in the instance except with probability negligible in $\ell$.

### 2.2. *Divide and Conquer Approach*

The most popular batch identification method is so-called the divide and conquer verifier (DCV) suggested by Pastuszak *et al.* (2000), which splits an instance into sub-instances and applies the generic test (GT) to each sub-instance recursively until all bad signatures are identified. In more details, $DCV_\alpha$ is defined as follows:

Algorithm $DCV_\alpha(x, t)$

Given a batch instance $x = ((s_1, m_1), \ldots, (s_t, m_t))$,

1. If $t = 1$ then
   (a) If GT$(x, 1)$ is "true" then return "true" and exit.
   (b) If GT$(x, 1)$ is "false" then return $x$ as a bad signature.
2. If GT$(x, t)$ is "true", then return "true" and exit.
3. Divide $x$ into $\alpha$ sub-instances $(x_1, \ldots, x_\alpha)$ containing $\frac{t}{\alpha}$ signatures each.
4. Run $DCV_\alpha(x_1, \frac{t}{\alpha}), \ldots, DCV_\alpha(x_\alpha, \frac{t}{\alpha})$.

## 3. Lee *et al.*'s Batch Identification and Stanek's Improvement

Lee *et al.* (2006) proposed two efficient methods $DBI_{\text{basic}}$ and $DBI_\alpha$ of identifying bad signatures in the batch verifier of the RSA signature scheme. Later, Stanek (2006) showed that their methods were flawed and suggested a possible improvement without giving its analysis. In this section, we briefly overview Lee *et al.*'s method and the Stanek's improved version.

The methods $DBI_{\text{basic}}$ and $DBI_\alpha$ assume a generic test (GT; Bellare, 1998) to check validity of a batch instance $(s_1, m_1), \ldots, (s_t, m_t)$. The authors claimed that their method could not only detect the existence of bad signatures but also identify the positions of bad signatures in a batch instance. The algorithms are described as follows:

Algorithm $DBI_{\text{basic}}(x, t)$

Given a batch instance $x = ((s_1, m_1), \ldots, (s_t, m_t))$,

1. If GT$(x, t)$ is "true" then return "true".
2. $M \leftarrow \prod_{i=1}^{t} m_i$; $M^* \leftarrow \prod_{i=1}^{t} m_i^i$.
3. $S \leftarrow \prod_{i=1}^{t} s_i$; $S^* \leftarrow \prod_{i=1}^{t} s_i^i$.
4. Find $k \in \{1, \ldots, t\}$ such that $(\frac{S^e}{M})^k = \frac{(S^*)^e}{M^*}$.

5. If $k$ does not exist then return "false".
6. If GT($x \backslash (s_k, m_k), t-1$) is "true" then return $k$.
7. Return "false".

In Lee *et al.* (2006), it was claimed that $DBI_{\text{basic}}$ returns "true" when all signatures are valid, the position of bad signature when there is only one bad signature, and "false", when there are two or more bad signatures, respectively.

Algorithm $DBI_\alpha(x, t)$

Given a batch instance $x = ((s_1, m_1), \ldots, (s_t, m_t))$,

1. If $t = 1$ then
   (i) If GT($x, t$) is "true" then return "true".
   (ii) Else return 1.
2. If $t = 2$ then
   (i) If GT($x, t$) is "true" then return "true".
   (ii) Else find $k \in \{1, 2\}$ such that

$$\left(\frac{(s_1 s_2)^e}{m_1 m_2}\right)^k = \frac{(s_1 s_2^2)^e}{m_1 m_2^2}.$$

   – If $k = 1$ return 1.
   – If $k = 2$ return 2.
   – Else return 1,2.
3. If GT($x, t$) is "true" then return "true".
4. $M \leftarrow \prod_{i=1}^{t} m_i$; $M^* \leftarrow \prod_{i=1}^{t} m_i^i$.
5. $S \leftarrow \prod_{i=1}^{t} s_i$; $S^* \leftarrow \prod_{i=1}^{t} s_i^i$.
6. Find $k \in \{1, \ldots, t\}$ such that $(\frac{S^e}{M})^k = \frac{(S^*)^e}{M^*}$.
7. If $k$ exists then
   – If GT($x \backslash (s_k, m_k), t-1$) is "true" then return $k$.
8. Divide $x$ into $\alpha$ batch instances $(x_1, \ldots, x_\alpha)$ containing (approximately) $\frac{t}{\alpha}$ signatures each.
9. Return $DBI_\alpha(x_1, \frac{t}{\alpha}) \cup \cdots \cup DBI_\alpha(x_\alpha, \frac{t}{\alpha})$.

The algorithm $DBI_\alpha$ uses $DBI_{\text{basic}}$ as a subroutine and its correctness and complexity.

## 4. Analysis of Lee *et al.*'s Scheme and Stanek's Improvement

As mentioned previously, Lee *et al.* claimed that their algorithms can detect the existence of bad signatures and, in addition, identify the the position of the bad signatures in a batch instance. But, Stanek pointed out a weakness of the methods and suggested an improvement. In this section, we point out another problem of Lee *et al.*'s methods, and then show that Stanek's improvement also suffers from an attack.

### 4.1. *On the Usage of Small Exponent Test for GT*

It was claimed that GT in the above algorithms could be initiated as small exponent test or random subset test described in Lee *et al.* (2006) and that the presented tests were suggested by Bellare *et al.* In fact, these are different from the tests of Bellare *et al.* (1998) and, in particular, the Lee *et al.*'s small exponent test is not batch verifier for RSA signature scheme.

Algorithm (Lee *et al.*'s small exponent test)

On input $(s_1, m_1), \ldots, (s_t, m_t)$ and a security parameter $\ell$,

1. Pick $h_1, \ldots, h_t \in \{0,1\}^\ell$ at random.
2. Compute $m = \prod_{i=1}^{t} m_i^{h_i}$ and $s = \prod_{i=1}^{t} s_i^{h_i}$.
3. If $m = s^e \pmod N$ then return "true"; else return "false".

Given a valid batch instance $(s_1, m_1), \ldots, (s_t, m_t)$, we let $\bar{s}_1 = -s_1$, $\bar{s}_i = s_i$ $(i = 2, \ldots, t)$. Then $(\bar{s}_1, m_1), \ldots, (\bar{s}_t, m_t)$ passes their small exponent test with probability $1/2$. That is, the test is not a batch verifier. In addition, $DBI_\alpha$ could not return any bad signature(s) even though there are invalid signatures. Therefore we conclude that GT must not be initiated as their small exponent test.

### 4.2. *Stanek's Improvement and an Essential Problem*

The algorithm $DBI_{\text{basic}}$ and Stanek's improvement find the bad signature using Theorem 1 in Lee *et al.* (2006). And the underlying idea of Theorem 1 in Lee *et al.* (2006) is the following.

Given an RSA modulus $N$ and $g \in \mathbb{Z}_N^*$, $g^k = g^j \pmod N$ implies $k = j$. (†)

Now we shall show that the statement (†) is not true. In practice, the condition $g^k = g^j \pmod N$ is equivalent to that the order of $g$ is a divisor of $j - k$. Especially, when $g = -1$, we have $g^k \equiv g^j \pmod N$ if and only if $j$ and $k$ have the same parity. Therefore, if we use only odd exponents as Stanek suggested to repair the problem of $DBI_{\text{basic}}$, Theorem 1 is of no use.

Moreover, if one has a set of pairs of valid signature and message, then it is possible to generate a set of pairs of signature and message with one bad signature where the $DBI_{\text{basic}}$ and the Stanek's improvement outputs "false".

For instance, we consider the Stanek's improvement. Stanek's improvement is almost the same as the Lee *et al.*'s method except it uses $M^* \leftarrow \prod_{i=1}^{t} m_i^{2i-1}$, $S^* \leftarrow \prod_{i=1}^{t} s_i^{2i-1}$, and $(\frac{S^e}{M})^{2k-1} \equiv \frac{(S^*)^e}{M^*}$. That is, it uses the following equation

$$\left( \frac{\left(\prod_{i=1}^{t} s_i\right)^e}{\prod_{i=1}^{t} m_i} \right)^{2k-1} = \frac{\left(\prod_{i=1}^{t} s_i^{2i-1}\right)^e}{\prod_{i=1}^{t} m_i^{2i-1}}. \tag{1}$$

Given a batch instance $(s_1, m_1), \ldots, (s_t, m_t)$ where all signatures are valid, we let $\bar{s}_i = s_i$ for $i = 1, \ldots, t-1$ and $\bar{s}_t = -s_t$. Clearly, the set of pairs $(\bar{s}_1, \bar{m}_1), \ldots, (\bar{s}_t, \bar{m}_t)$

has one bad signature $(\bar{s}_t, \bar{m}_t)$. But for all $k(1 \leqslant k \leqslant t)$,

$$\left(\frac{\left(\prod_{i=1}^{t} \bar{s}_i\right)^e}{\prod_{i=1}^{t} m_i}\right)^{2k-1} = \left(\frac{(-s_t)^e}{m_t}\right)^{2k-1} \equiv -1 \pmod{N},$$

$$\frac{\left(\prod_{i=1}^{t} \bar{s}_i^{2i-1}\right)^e}{\prod_{i=1}^{t} m_i^{2i-1}} = \frac{\left((-s_t)^{2k-1}\right)^e}{{m_t}^{2k-1}} \equiv -1 \pmod{N}.$$

Since $(\bar{s}_t, \bar{m}_t)$ is invalid, GT($x \backslash (\bar{s}_k, \bar{m}_k)$) outputs "false" for all $k$ $(k \neq t)$. If the first tested value $k$ is less than $t$ then the improved method by Stanek will return "false" meaning the input instance contains two or more bad signatures. That is, it outputs "false" even when there is only one bad signature in the instance of $t$ pairs of signature and message. Hence the output "false" of Stanek's method as well as $DBI_{\text{basic}}$ does not imply the existence of two or more invalid signatures for the given instance.

Since improved $DBI_\alpha$ uses improved $DBI_{\text{basic}}$ as subroutine and the subroutine $DBI_{\text{basic}}$ doesn't correctly identify the bad signature when there is only one bad signature, the resultant complexity of the improved $DBI_\alpha$ is not better than Divide and Conquer Verifier method (Pastuszak *et al.*, 2000).

**Remark 1.** *In order to correct Theorem* 1*, one needs a condition for the above statement (†) '$g \in \mathbb{Z}_N^*$, $g^k = g^j$ implies $k = j$' to be true.*

## 5. Finding Invalid Signatures in Boyd *et al.*'s Batch Verifier

As we have shown, Lee *et al.*'s method of identifying bad signatures is not correct in their context. Though, since the underlying idea is useful, we adopt their idea for an efficient identifying method of bad signatures in a batch verification designed for Gennaro *et al.*'s modified RSA signature scheme after correcting the theorem in our case.

### 5.1. *Gennaro et al.'s Signature Scheme and Boyd et al.'s Batch Verifier*

Gennaro *et al.* (1997) introduced a modified RSA signature scheme that defined as follows:

(i) Assume that we have a hash function $h()$ and an RSA modulus $N = pq$ with safe primes $p$ and $q$ (that is, $p,\ q,\ (p-1)/2$, and $(q-1)/2$ are odd primes).

(ii) A signature of a message $m$ is $\sigma = \alpha h(m)^d \pmod{N}$ for some $\alpha$ with $\text{ord}(\alpha) \leqslant 2$, where $\text{ord}(\alpha)$ denote the multiplicative order of $\alpha$ modulo $N$.

(iii) $(\sigma, h(m))$ is a valid signature if $\frac{\sigma^e}{h(m)}$ is an element in $Z_N^*$ of order $\leqslant 2$, i.e., $\sigma^{2e} = h(m)^2 \pmod{N}$.

For the notational simplicity, we shall write the input message for the signature as $m$ instead of $h(m)$.

Boyd and Pavlovski (2000) proposed a batch verifier of Gennaro *et al.*'s modified RSA signature scheme. We describe the batch verification proposed by Boyd *et al.*

Assume $p$ and $q$ are safe primes with $p-1=2p',\ q-1=2q'$ and let $N=pq$ and $e,d$ be parameters of a signer. Suppose $\ell$ is the security parameter and assume that $2^{\ell}<\min(p',q')$. For given $x$ of $t$ pairs of signature and message the algorithm of batch verification is denoted by $BP(x,t)$ or simply $BP$.

Algorithm $BP(x,t)$

Given an instance $(s_1,m_1),\ldots,(s_t,m_t)$,

1. Check that $\gcd(s_i,N)=1$ for all $i=1,\ldots,t$.
2. Pick $h_1,\ldots,h_t\in\{0,1\}^{\ell}$ at random.
3. Compute $s=(\prod_{i=1}^{t}s_i^{h_i})^e \pmod N$ and $m=\prod_{i=1}^{t}m_i^{h_i} \pmod N$.
4. If $s^2=m^2 \pmod N$ then accept, else reject.

### 5.2. *The Proposed Method $DBI_{bp}$ of Identifying Bad Signature in $BP$*

In this section, we construct an efficient identification method in the modified RSA signature scheme suggested by Gennaro *et al.* and, then, show the correctness of our method by modifying the theorem we have shown in the above section.

We shall describe our proposed batch identification method and call our method as $DBI_{bp}(x,t)$ or simply $DBI_{bp}$.

Algorithm $DBI_{bp}(x,t)$

Given a batch instance $x=((s_1,m_1),\ldots,(s_t,m_t))$,

1. If $BP(x,t)$ is "accept" then return "true".
2. $M\leftarrow\prod_{i=1}^{t}m_i;\ M^*\leftarrow\prod_{i=1}^{t}m_i^i$.
3. $S\leftarrow\prod_{i=1}^{t}s_i;\ S^*\leftarrow\prod_{i=1}^{t}s_i^i$.
4. Find $k\in\{1,\ldots,t\}$ such that $(\frac{S^e}{M})^{2k}=(\frac{(S^*)^e}{M^*})^2$.
5. If $k$ does not exist then return "false".
6. If $BP(x\backslash(s_k,m_k),t-1)$ is "true" then return $k$.
7. Return "false".

$DBI_{bp}$ returns "true" if all signatures are valid, the position of bad signature if there is only one bad signature, and "false" if there are two or more bad signatures. Note that for the RSA-type signature scheme, the batch verifier $BP$ can be done only for the signatures with the same signer. So it is reasonable to assume that the number $t$ of the signatures in the batch instance is less than $p$ and $q$.

Before showing the correctness of our method, let us consider the case when a batch instance $(s_1,m_1),\ldots,(s_t,m_t)$ has exactly one bad signature $(s_k,m_k)$. In this case, since

$s_j^e = m_j$ for all $j \neq k$, we obtain the following equation:

$$\left(\frac{\left(\prod_{i=1}^{t} s_i\right)^e}{\prod_{i=1}^{t} m_i}\right)^{2k} = \left(\frac{\left(\prod_{i=1}^{t} s_i^i\right)^e}{\prod_{i=1}^{t} m_i^i}\right)^2. \tag{2}$$

**Theorem 1.** *Suppose that* $p \equiv 3 \pmod 4$ *and* $q \equiv 3 \pmod 4$*, for example* $p$ *and* $q$ *are safe primes,* $t$ *be an integer less than the smallest odd prime factor of* $\phi(N)$*, and* $\gcd(\phi(N), e) = 1$*. If* $(s_1, m_1), \ldots, (s_t, m_t)$ *has only one bad signature and if* $k$ *is a positive integer satisfying* (2) *then* $(s_k, m_k)$ *is the bad signature.*

*Proof.* Let $(s_j, m_j)$ be the bad signature and $k$ be the smallest positive integer satisfying (2). Then $s_j^{2e} \neq m_j^2$, and so we have two nonidentity quantities:

$$\left(\frac{\left(\prod_{i=1}^{t} s_i\right)^e}{\prod_{i=1}^{t} m_i}\right)^{2k} = \left(\frac{s_j^e}{m_j}\right)^{2k},$$

and

$$\left(\frac{\left(\prod_{i=1}^{t} s_i^i\right)^e}{\prod_{i=1}^{t} m_i^i}\right)^{2} = \left(\frac{s_j^e}{m_j}\right)^{2j}.$$

By the assumption on $k$, the two equations are equal. Letting $\alpha_j = \frac{s_j^e}{m_j} \neq 1$, we have $\alpha_j^{2(j-k)} \equiv 1 \pmod N$.

Suppose that $j-k > 0$. Then the order of $\alpha_j$ must be a divisor of $\gcd(\phi(N),\ 2(j-k))$. Note that $j - k$ is less than the smallest odd prime factor of $\phi(N)$ and $\phi(N) = 4a$ for some odd number $a$. So, the order of $\alpha_j$ must be $1, 2$ or $4$. Since $p, q \equiv 3 \pmod 4$, $\mathbb{Z}_N^\times$ has no element of order $4$ and thus the order of $\alpha$ is $1$ or $2$. This contradicts to the fact $(s_j, m_j)$ is bad signature. Thus we conclude $j - k = 0$ and $j = k$. □

In order to identify multiple bad signatures in batch instances, one can generalize the $DBI_{bp}$ by employing the divide and conquer approach as Lee *et al.* (2006) did. However, the straightforward generalization is not as efficient as the Pastuszak *et al.*'s DCV when an instance contains multiple bad signatures, because such a generalization deals with entire signatures without considering efficiency. Hence, instead of using direct generalization, we use the basic idea of the divide and conquer in a slightly different approach. Note that in practical situations and literatures, a set of signatures to be batched contains very small number of invalid signatures. For given an instance, we first randomly shuffle the signatures, and then splits the instance into several blocks depending on the number of input signatures before calling batch verifier. That is, we take the size of each block such that the probability that each block contains two or more bad signatures is negligible and the efficiency is assured over well known methods assuming each block has at most one bad signature. To claim this method is sufficiently efficient, it is enough to find the

suitable size of blocks to run $DBI_{bp}$ in the presence only one bad signature. We will investigate the efficiency issues in the next section.

We now show the proposed method is a batch identifier for the RSA-type signature scheme with respect to the batch verifier.

**Theorem 2.** *The proposed method is a secure batch identifier relative to the batch verifier $BP$. That is, given an instance $x$, if $x$ contains invalid signatures then $DBI_{bp}$ finds the invalid ones.*

*Proof.* As we have shown in the above, it is enough to show the method correctly find the invalid signature in the presence of single one. We assume that an instance $x = ((s_1, m_1), \ldots, (s_t, m_t))$ contains one bad signature, say $(s_k, m_k)$. Since $BP$ is a secure batch verifier (Boyd and Pavlovski, 2000), $BP(x)$ returns "reject" except with negligible probability. Now, by applying Theorem 1 into Steps 2–4 in $DBI_{bp}$, we have the invalid signature $(s_k, m_k)$. This completes the proof. □

## 6. Efficiency Analysis

In this section, we estimate the computational cost of $DBI_{bp}$ in terms of the required number of modular multiplications, and compare $DBI_{bp}$ with Pastuszak et al.'s divide and conquer verifier method.

### 6.1. *Complexity of $DBI_{bp}$*

In Boyd (2000), it was proved that the cost of the batch verifier $BP(x, t)$ is approximately $\ell(t + 2) + \frac{3}{2}|e| + t - 1$ modular multiplications. It was noted that the algorithm is faster than individual verification when $e$ is large enough and satisfies $|e| \gg \frac{3\ell(t+2)}{2(t-1)} + \frac{2}{3}$. So $BP(x, t)$ is useful if the exponent $e$ is chosen to be $|e| \geqslant 2\ell$ for reasonable size $\ell$, and we assume so.

$DBI_{bp}$ requires the same number of modular multiplications as in $DBI_{\mathrm{basic}}$. The difference occurred is from the cost of underlying batch verifier and 2 modular squares additionally needed in $DBI_{bp}$. According to Lee (2006), $DBI_{\mathrm{basic}}$ takes one modular exponentiation plus $2t + \frac{3}{2}\sqrt{t}$ modular multiplications (excluding the cost of the batch verifier). And thus $DBI_{bp}$ requires to perform $BP(x, t)$, $BP(x \backslash (s_k, m_k)), t - 1)$, and $2t + \frac{3}{2}\sqrt{t} + 3/2|e| + 2$ modular multiplications. We let $V_t$ be the number of modular multiplications of $BP(x, t)$ and $F_t$ be $2t + \frac{3}{2}\sqrt{t} + \frac{3}{2}|e| + 2$. Then $V_t = \ell(t+2) + \frac{3}{2}|e| + t - 1$ and the computational cost of $DBI_{bp}$ is as follows:

$$V_t + F_t + V_{t-1} = (2\ell + 4)t + \frac{3}{2}\sqrt{t} + \frac{9}{2}|e| + 3\ell - 1.$$

### 6.2. *$DCV_\alpha$ and Efficient Choices of $\alpha$ in $DCV_\alpha$*

Pastuszak *et al.* proposed a method so called divide and conquer verifier $DCV_\alpha$ (or $DCV_\alpha(x, t)$) to identify bad signatures in a batch instance in Pastuszak *et al.* (2000).

The computational cost of $DCV_\alpha$ was evaluated in terms of the number of GT call and estimated for the number of signatures in batch instances. It was also noted that 2 and 4 are optimal $\alpha$ when there is single bad signature. In practice, their choice of $\alpha$ may not be optimal by considering the computational cost of underlying verifiers. Most known batch verifiers have linear complexity with respect to the number of signatures (Camenish *et al.*, 2007; Naccache *et al.*, 1994; Pastuszak *et al.*, 2000) and we assume a batch verifier takes computational cost $V_t = at + b$ for some constant $a, b$, where $t$ is the number of input signatures. When there is one bad signature in a batch instance, $DCV_\alpha$ takes the cost:

$$\begin{aligned} V_t + \alpha V_{t/\alpha} + \cdots + \alpha V_{t/\alpha^\delta} &= (at+b) + \frac{\alpha^\delta - 1}{\alpha^{\delta-1}(\alpha-1)} at + b\alpha\delta \\ &\approx (at+b) + \frac{t-1}{\frac{t}{\alpha}(\alpha-1)} at + b\alpha \log_\alpha t \\ &= (at+b) + A\frac{\alpha}{\alpha-1} + B\frac{\alpha}{\ln \alpha}, \end{aligned} \tag{3}$$

where $A = a(t-1),\ B = b\ln t$, and $\delta = \lceil \log_\alpha t \rceil$.

Let $\phi_t(x) = A\frac{x}{x-1} + B\frac{x}{\ln x}$ and $\alpha_t$ be the ceiling of the value $x$ at which $\phi_t(x)$ is minimum. Then $DCV_{\alpha_t}$ is almost optimal method among the divide and conquer verifiers $DCV_\alpha$.

For instance, we investigate the optimal choice of $\alpha$ in $DCV_\alpha$ and compare our choice of $\alpha$ with the Pastuszak *et al.*'s optimal divide and conquer method ($\alpha = 2, 4$) in the Boyd *et al.*'s batch verifier $BP(x,t)$. We assume that the exponent $e$ is large so that $|e| \approx 2\ell$ in our selection of efficient choice of $\alpha$. Then $V_t = (\ell+1)t + 5\ell - 1$ and $F_t = 2t + \frac{3}{2}\sqrt{t} + 3\ell + 2$. Thus the $DBI_{bp}$ is done in $(2\ell+4)t + \frac{3}{2}\sqrt{t} + 12\ell - 1$ modular multiplication. And the cost of $DCV_\alpha$ can be computed by the above equation (3) with $a = \ell + 1$ and $b = 5\ell - 1$. The security parameter $\ell$ of the batch verifier was set to 60 in Pastuszak *et al.* (2000), but we choose $\ell$ to be 30 which is enough for practical requirement meaning the error probability will be $2^{-30}$. All the results are shown in Tables 1 and 2.

From the results, we can observe that $DCV_{\alpha_t}$ shows better performance than $DCV_2$ and $DCV_4$ in the case when the underlying batch verifier is $BP(x,t)$. In fact, $DCV_{\alpha_t}$ is more efficient than $DCV_\alpha$ for $\alpha(\neq \alpha_t)$, and the same results can be made for large number of signatures.

Table 1

Optimal choice of $\alpha$ for each $t$ signatures

| $t$ | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |
|---|---|---|---|---|---|---|---|---|
| $\alpha_t$ | 6 | 4 | 6 | 7 | 8 | 11 | 13 | 18 |
| # mult | 4086 | 6525 | 11,539 | 20,446 | 37,701 | 71,702 | 138,195 | 269,593 |

Table 2
Number of multiplication for $\alpha = 2, 4$

| $t$ | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |
|---|---|---|---|---|---|---|---|---|
| $\alpha = 2$ | 4553 | 7827 | 14,077 | 26,279 | 50,385 | 98,299 | 19,3829 | 384,591 |
| $\alpha = 4$ | 4231 | 6525 | 11,771 | 21,009 | 40,143 | 77,157 | 151,843 | 299,961 |

Table 3
The number of multiplications in $DCV_{\alpha_t}$ and $DBI^*_{\text{basic}}$

| $t$ | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |
|---|---|---|---|---|---|---|---|
| $DCV_{\alpha_t}$ | 6525 | 11,539 | 20,446 | 37,701 | 71,702 | 138,195 | 269,593 |
| $DBI^*_{\text{basic}}$ | 4467 | 8568 | 16,767 | 33,161 | 65,943 | 131,499 | 262,599 |

6.2.1. *Comparison of $DBI_{bp}$ with $DCV_\alpha$ in the Presence of One Bad Signature*
Finally, we compare our method $DBI_{bp}$ with $DCV_\alpha$ when an input instance contains one bad signature out of $t$ signatures and $BP$ is used as underlying batch verifier. $DBI_{bp}$ requires $V_t + F_t + V_{t-1}$ multiplications. On the other hand, $DCV_\alpha$ takes $V_t + \alpha V_{t/\alpha} + \cdots + \alpha V_{t/\alpha^\delta}$ multiplications where $\delta = \lceil \log_\alpha t \rceil$.

We estimated the number of multiplications of $DBI_{bp}$ and optimal divide and conquer verifiers. The results are shown in Table 3. In the table, $t$ is the number of signatures to be verified, and $DCV_{\alpha_t}$ is the optimal choice of $\alpha$ in divide and conquer verifier taken by the method with the underlying batch verifier $BP(x, t)$ that was computed in the previous section. The results are evaluated for the number of signatures $\leqslant 10000$. Since most known RSA-type batch verifiers as well as Pastuszak's $BP$ are for single signer, it is reasonable to assume that $t$ is less than or equal to 10 thousands practically.

As we note in Table 3, we can observe that $DBI_{bp}$ reduces the number of modular multiplications by 2971, 5759, and 6994 compared with the optimal $DCV_{\alpha_t}$ when $t = 128$, 1024, and 4096, respectively. In fact, $DBI_{bp}$ takes less modular multiplication than $DCV_{\alpha_t}$ for all $t \leqslant 10000$, and so our method $DBI_{bp}$ is reduces the computational cost compared with the divide and conquer verifier $DCV_\alpha$ for any $\alpha$ in the case when there is one bad signature in the instance. As mentioned previously, by shuffling and partitioning, we may assume that each partitioned block contains at most one invalid signature. Therefore, we can conclude that our method is better than the well known divide and conquer verifier in the Gennaro *et al.*'s signature scheme.

## 7. Conclusion

In this paper, we have pointed out a flaw (different from the Stanek's attack) on the Lee *et al.*'s batch identification method and described an attack on the Stanek's improved method. From the analysis, their methods could not find out bad signatures even if batch

instances contain invalid signatures. Furthermore, we have proposed a batch identification method to identify the bad signatures in Gennaro *et al.*'s signature scheme. Our method might be thought of as a modification of Lee et.'s scheme so as to apply Gennaro *et al.*'s signature scheme. The proposed method is more efficient than the divide and conquer verifier for the signature scheme.

**Acknowledgements.** The authors would like to thank anonymous reviewers for their valuable comments.

## References

Bellare, M., Garay, J., Rabin, T. (1998). Fast batch verification for modular exponentiation and digital signatures. In: *Eurocrypt'98, Lecture Notes in Computer Science*, Vol. 1403. Springer, Berlin, pp. 236–250.

Boyd, C., Pavlovski, C. (2000). Attacking and repairing batch verification schemes. In: *Asiacrypt'00, Lecture Notes in Computer Science*, Vol. 1976. Springer, Berlin, pp. 58–71.

Camenisch, J., Hohenberger, S., Pedersen, M. (2007). Batch verification of short signatures. *Eurocrypt'07, Lecture Notes in Computer Science*, Vol. 4515. Springer, Berlin, pp. 246–263.

Ferrara, A., Green, M., Hohenberger, S., Pedersen, M. (2009). On the practicality of short signature batch verification. *CT-RSA* (to appear). `http://eprint.arcr.org/2008/015`.

Fiat, A. (1989). Batch RSA. In: *Crypto'89, Lecture Notes in Computer Science*, Vol. 435. Springer, Berlin, pp. 175–185.

Gennaro, R., Krawczyk, H., Rabin, T. ( 1997). RSA-based undeniable signatures. In: *Crypto'97, Lecture Notes in Computer Science*, Vol. 1294. Springer, Berlin, pp. 132–149.

Gennaro, R., Krawczyk, H., Rabin, T. (2000). RSA-based undeniable signatures. *Journal of Cryptology*, 13(4), 397–416.

Hwang, M.-S., Lee, C.-C., Tang, Y.-L. (2001). Two simple batch verifying multiple digital signatures. In: *Proceedings of Information and Communications Security, Lecture Notes in Computer Science*, Vol. 2229. Springer, Berlin, pp. 233–237.

Law, L., Matt, B. (2007). Finding invalid signatures in pairing-based bathes. In: *Cryptography and Coding 2007, Lecture Notes in Computer Science*, Vol. 4887. Springer, Berlin, pp. 34–53.

Lee, S., Cho, S., Choi, J., Cho, Y. (2006). Efficient identification of bad signatures in RSA-type batch signature. In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E89-A(1). pp. 74–80.

Liu, J., Huang, S. (2010). Identity-based threshold proxy signature from bilinear pairings. *Informatica*, 21(1), 41–56.

Naccache, M'Raihi, Vaudenay, Raphaeli (1994). *Can DSA be improved? Complexity trade-offs with the digital signature standard*. In: *Eurocrypt'94, Lecture Notes in Computer Science*, Vol. 0950. pp. 77–85.

Pastuszak, J., Michalek, D., Pieprzyk, J., Seberry, J. (2004). Identification of bad signatures in batches. In: *PKC 2000, Lecture Notes in Computer Science*, Vol. 1751. Springer, Berlin, pp.28–45.

Stanek, M. (2008). Attacking LCCC batch verification of RSA signatures. *International Journal of Network Security*, 6(3), 255–257.

Sun, X., Li, J., Yin, H., Chen, G. (2010). Delegatability of an identity based strong designated verifier signature scheme. *Informatica*, 21(1), 117–122.

**K. Kim** received the BS degree in mathematics from Konyang University, and the MS and the PhD degrees in mathematics from Inha University, Korea. He was a postdoctoral researcher at Graduate School of Information Technology and Telecommunications in Inha University. He is currently an instructor of Department of Mathematics in Inha University. His current research interests include algebraic/algorithmic number theory, elliptic curves, and privacy enhanced signatures.

**I. Yie** received the BS and MS degrees in mathematics from the Seoul National University, Seoul, Korea, and the PhD 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.

**S. Lim** received her BS degree in mathematics from the Dongguk University, Korea, in 1985. In 1987, she received her MS 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.

**H. Park** received his BS in mathematics from Chonnam National University, Kwangju, Korea in 1999, his MS in mathematics from Seoul National Unversity, Seoul, Korea, in 2001, and his PhD in interdisciplinary program of information security from Chonnam National University, Kwangju, Korea in 2006. He is working on cryptography team of KISA (Korea Information Security Agency).

## Blogų parašų suradimo metodas RSA tipo paketinio tikrinimo sistemoje

Kitae KIM, Ikkwon YIE, Seongan LIM, Haeryong PARK

Paketinė kriptografija skirstoma į paketinę identifikaciją ir paketinę verifikaciją. Paketinė verifikacija nustato ar parašų aibėje yra negaliojančių parašų, o paketinė identifikacija suranda blogus parašus, jei parašų aibėje yra negaliojančių parašų. Neseniai šį uždavinį sprendė Lee *et al.*, Ferrara *et al.* ir Law *et al.* Šiame straipsnyje parodyti tam tikri Lee *et al.* darbo trūkumai ir pasiūlytas RSA tipo parašo identifikavimo metodas. Pasiūlytasis metodas yra efektyvesnis negu gerai žinomas parašo schemos išskaidymo metodas. Pasiūlytas optimalių išskaidymo verifikatorių parinkimo algoritmas.