Computer Standards & Interfaces 40 (2015) 42–52

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www elsevier com/locate/csi

Improved batch veri cation of signatures using generalized sparse exponents

Jung Hee Cheon [a], Mun-Kyu Lee [b,*]

[a] Department of Mathematical Sciences, Seoul National University, Seoul 151-742, Republic of Korea
[b] Department of Computer and Information Engineering, Inha University, Incheon 402-751, Republic of Korea

RTICLE INFO

Article history:
Received 19 August 2014
Accepted 2 December 2014
Available online 30 January 2015

Keywords:
Public key cryptosystem
Digital signature
Elliptic curve
Batch veri cation
Non-adjacent form

BSTR CT

We propose an ef cient method for batch veri cation of exponentiation using width-w Non-Adjacent Forms (w-NAFs), which can be applied to modi ed DSA and ECDSA signatures. We further generalize this method to use tau-adic w-NAF scalars on elliptic curves with complex multiplication such as Koblitz curves. The theoretical analyses and experimental results show that our method accelerates the individual veri cation by a factor of up to 7.49 in the single-signer case and by up to 1.47 in the multiple-signer case for 1000 instances over a Koblitz curve K233. Our method can also be exploited to accelerate batch veri cation of pairing-based signatures.

© 2015 Elsevier B.V. All rights reserved.

1 Introduction

Batch veri cation was introduced by Naccache et al. to verify multiple signatures ef ciently [25]. Their method is to use a set of small exponents to verify multiple exponentiations simultaneously. Let G be an abelian group with a generator g. Given a batch instance of N pairs $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ with $x_i \in$ and $y_i \in G$, the algorithm checks if $g^{\sum_{i=1}^{N} x_i s_i} = \prod_{i=1}^{N} y_i^{s_i}$ for randomly chosen $s_i \in S$, where the exponent set S is taken to be the set of e-bit prime integers for small e. This test was improved by adopting a small exponent set $\{0, 1\}^{\ell}$ for small ℓ by Yen and Laih [30] and Bellare et al. [4]. Another improvement [11] was obtained by taking longer integers with small Hamming weights, so called *sparse exponents*, as elements of S rather than small integers.

In this paper, we improve the previous results by adopting generalized sparse exponents which we call *width-w Non-Adjacent Form* (w-NAF for short). First, we de ne a *w-NAF of weight t with digit set D* as a radix-2 representation satisfying that; (1) each nonzero digit is an element in D, (2) at most one of any w consecutive digits is nonzero, and (3) the number of nonzero digits is t. Its radix- analog, i.e., a -adic w-NAF, is de ned similarly for a complex number . Then, by taking random s_is from a set of sparse (-adic) w-NAFs of a xed small weight t, we signi cantly reduce the cost for signature veri cation.

We rst provide ef cient sampling algorithms for w-NAF exponents, which are based on the unique representations for integer w-NAFs and -adic w-NAFs, respectively. By using w-NAF exponents together with the legacy simultaneous exponentiation technique, we can show that N exponentiations can be veri ed with approximately $21N$ multiplications under a medium level of security de ned by NIST and ECRYPT II [17], i.e., assuming that the error probability of a batch veri cation is $1/2^{112}$. Note that the costs are $57N$ and $27N$ in the small exponent test [4] and the sparse exponents test [11], respectively. Over an elliptic curve group where a point subtraction is as ef cient as a point addition, our veri cation cost becomes approximately $19N$ point additions for batch veri cation of N scalar multiplications. This gure drops further over a Koblitz curve. Using sparse -adic w-NAFs and our parallelized version of the BGMW scalar multiplication [8], we reduce the veri cation cost to $8N$ over the standard curve K233.

We also present asymptotic complexity analyses for our methods. According to these analyses, the number of multiplications per exponentiation in our method is sublinear in the size of security parameter, l, i.e., $O(l/logl)$, which is smaller by a factor of $\log l$ than that of an individual veri cation, $O(l)$.

We remark that the practical performance can be different from the theoretical estimation. For the basic small exponent tests, some implementations have been reported in the context of pairing-based signatures [16], after the conference version of this paper [12] has been published, and the results comply with the theoretical estimation. For the sparse exponent test, however, it was not clear if there is any gap between theory and practice. In this paper, we attempt to verify the performance of our methods by a real implementation of modi ed ECDSA [3]. This is the rst implementation of batch veri cation with sparse

* Corresponding author.
E-mail addresses: jhcheon@snu.ac.kr (J.H. Cheon), mklee@inha.ac.kr (M.-K. Lee).

http://dx.doi.org/10.1016/j.csi.2014.12.004
0920-5489/© 2015 Elsevier B.V. All rights reserved.

exponents, and also the first implementation of (modified) ECDSA with secure batch verification.

According to our experimental results, the fastest method, i.e., batch verification using sparse τ-adic w-NAFs, accelerates the individual verification by a factor of 7.49 over K233 when the batch size is 1000 and all the signatures are signed by a single party. This is possible thanks to the surprising result that only 8 elliptic curve additions are sufficient for the proposed method to verify one signature. We obtain similar performance gains for K163 and K283, where our method requires only 6 and 10 point additions per signature, respectively.

It should be noted that through our implementation, we verified that the Frobenius map computation is not negligible, especially in the case that the number of point additions per scalar is extremely small as in batch verification. Therefore, the number of Frobenius maps should be considered as another criterion when we select a verification method from many possible candidates. This observation led us to develop a new batch verification algorithm (Algorithm 4) using τ-adic w-NAFs, which significantly reduces the number of Frobenius maps compared to the algorithm in the conference version of this paper [12]. Another finding is that the secondary operations such as message hashing and big integer operations modulo the subgroup order also have a minor but non-negligible effect on performance.

In this paper, we also speed up the batch verification of signatures generated by multiple signers. The batch verification problem for signatures on different messages generated by different signers was initiated by the conference version of this paper [12]. According to our complexity analysis, our sparse integer w-NAF method is more than 4 times faster than individual verification when the batch size is 1000. Our experimental results show that even for Koblitz curves where the individual verification is already fast due to the dramatic reduction in the number of point additions, our τ-adic w-NAF method accelerates the individual verification by 1.47, 1.47, and 1.55 times in K163, K233, and K283, respectively.

Finally, we show that the generalized sparse exponents can be applied to accelerate the batch verification of various pairing-based signatures, too. To be precise, we consider two example schemes, BLS [6] and -IBS [9], and verify that their batch verification can be improved by 13% and 10%, respectively, if we use generalized sparse exponents instead of small exponents.

1.1. Applications

Batch verification will be useful in any settings where multiple signatures need to be verified at once. We have a variety of applications to which our proposed method may be applied. For example, in e-cash applications, merchants and/or consumers need to verify the validity of lots of electronic coins signed by the bank. E-voting systems need to verify huge number of signed ballots as fast as possible. In the outsourced database applications [23], the query request messages from clients need to be authenticated by servers. Another example is authenticated routing based on public key cryptography, in which network packets are signed and verified by each node and each router has to verify many signatures in real time. We are also able to apply batch verification to Mix-Net [1] to design privacy-preserving systems or protocols. Another promising application would be VSS (Verifiable Secret Sharing) [15], which is a fundamental building block for fault-tolerant and secure distributed computations such as reliable broadcast, peer group membership management, and Byzantine agreement.

1.2. Related works

After the conference version of this paper was presented [12], Camenisch, Hohenberger, and Pedersen [9] proposed the batch verifier for pairing-based short signatures and identity-based signatures. They used the small exponent test to reduce the total number of pairings required for signature verification. Ferrara et al. showed through implementation that the technique in [9] performs well in practice [16]. Akinyele et al. proposed an automated tool for generating batch verification code from a high level representation of a pairing-based signature scheme [2]. As already mentioned above, we also improve the batching algorithm in [9] by applying our generalized sparse exponents in Section 6. On the other hand, there was an attempt [19] to batch verify the original ECDSA instead of the modified ECDSA. The authors of [19] verified the practical performance of their technique through implementation. However, this method was shown to be insecure because it did not randomize the linear combination being verified [5]. If randomization is considered, the method in [19] does not give significant performance gain compared to the individual verification, in particular in the multiple-signer case [20].

1.3. Organization

The remainder of this paper is organized as follows. In Section 2, we define batch verification and explain the modified versions of DSA [25] and ECDSA [3] for batch verification. In Section 3, we propose the new batch verification methods for exponentiation and scalar multiplication based on the sparse w-NAF representations. The proof of uniqueness of these representations is also given as well as the corresponding selection algorithms. In addition, the asymptotic complexities of the new methods are analyzed, and the optimal parameters minimizing the verification cost are presented. The new methods are applied to signature schemes in Section 4, where both the single-signer case and the multiple-signer case are considered. The performance of the proposed verification method is compared with those of the previous methods including the individual verification and the small exponent test through the experiments in Section 5, where we also analyze the various hidden factors that affect the overall performance. In Section 6, we apply our method to pairing-based signatures and estimate the performance. Finally, we conclude in Section 7.

2. Preliminaries

2.1. Notations

Notations used throughout this paper are summarized in Table 1.

Table 1
Notation.

p	Prime number
r	Prime order of a subgroup
G	Abelian group
g	Generator of a multiplicative group
P	Generator of an elliptic curve group
x	Private key
y	Public key over a multiplicative group
Q	Public key over an elliptic curve group
w	Window size
m	Length of exponents
t	Hamming weight
Mem	Number of group elements to be stored
$\mathcal{E}_f$	Finite field exponentiation
$\mathcal{M}_f$	Finite field multiplication
$\mathcal{S}_f$	Finite field squaring
$\mathcal{M}_e$	Scalar multiplication over an elliptic curve
$\mathcal{A}_e$	Elliptic curve addition
$\mathcal{D}_e$	Elliptic curve doubling
$\mathcal{F}_e$	Frobenius mapping

2.2. Batch veri cation

Let G be a cyclic group of prime order r with a generator g. Given a subset S of $_r$, we de ne a batch veri er $\mathcal{V}_S$ for exponentiation following the construction in [4,11]:

1 Input a batch instance $\{(x_i, y_i) \in {}_r \times G | i = 1, 2, \ldots, N\}$.
2 $\mathcal{V}_S$ takes N elements $c_1, c_2, \ldots, c_N$ uniformly from S.
3 $\mathcal{V}_S$ computes $c = \sum_{i=\ 1}^{N} c_i x_i$ and $z = \prod_{i=1}^{N} y_i^{c_i}$.
4 If $g^c = z$ output 1 and otherwise output 0.

We say a batch instance $\{(x_i, y_i) \in {}_r \times G | i = 1, 2, \ldots, N\}$ is correct if $g^{x_i} = y_i$ for all i, and incorrect otherwise. Note that the veri er $\mathcal{V}_S$ outputs 1 for a correct batch instance regardless of S. We de ne $Fail(\mathcal{V}_S)$ to be the maximum probability that $\mathcal{V}_S$ outputs 1 for an incorrect batch instance, where the probability is over the random choice of $c_1, \ldots, c_N$ from S. Theorem 1 in [11] shows that it is upper-bounded by $1/|S|$; that is, we have $g^{x_i} = y_i$ for all i with probability at least $1 \quad 1/|S|$ if $\mathcal{V}_S$ outputs 1.

2.3. Modi ed DSA and modi ed ECDSA

To apply the batch veri cation to signatures, one needs to modify signature schemes. Naccache et al. presented a modi ed DSA for batch veri cation [25]. Our batch veri cation is also applicable to this modi- ed DSA. But, considering the attack by Boyd and Pavlovski [7], we made a little change to the veri cation procedure.

Let p be a prime and r be a much smaller prime dividing $p \quad 1$. We assume that $(p \quad 1)/(2r)$ has no divisor less than r to resist the attack in [7]. Let g be a generator of a subgroup G of order r in $_p$. Take a random $x \in {}_r$. The private key is x and the corresponding public key is $y = g^x$ mod p. A signature for a hashed message $m \in {}_r$ is given by

$$\left(\rho = g^k mod p, \quad \sigma = k^{\ 1}(m + x\rho) mod r\right)$$

for a random $k \in {}_r$. It is veri ed by checking if $\rho = \pm g^a y^b$ mod p for $a = m\sigma^{\ 1}$ mod r and $b = \rho\sigma^{\ 1}$ mod r. Note that $\rho = ((g^k \bmod p) \bmod r)$ is used in the original DSA [13]. The veri cation admits only $\rho = g^a y^b$ mod p, but here we relax the veri cation to admit $\rho = \pm g^a y^b$ mod p due to Boyd and Pavlovski attack, in which the security loss is only one bit.

We also apply this modi cation to ECDSA [14] and use the modi ed ECDSA presented in [3], whose security is equivalent to the standard ECDSA. Let E be an elliptic curve. Assume that the order r of E is prime and $P \in E$ a generator (If E has a cofactor $\neq 1$, the signature scheme should be modi ed due to [7]). The private key is x and the corresponding public key is $Q = xP$. The signature for a given hashed message $m \in {}_r$ is

$$\left(R = kP, \quad \sigma = k^{\ 1}(m + x\rho) \quad mod r\right)$$

where $R = (x_1, y_1)$ and $\rho = x_1$ mod r for a random $k \in {}_r$. The veri cation is done by checking if $R = aP + bQ$ for $a = m\sigma^{\ 1}$ mod r and $b = \rho\sigma^{\ 1}$ mod r.

3 Batch veri cation of exponentiations using generalized sparse exponents

3.1. Batch veri cation using sparse integer w-NAFs

Let $w \geq 2$ be an integer. A radix-2 representation is called a *width-w Non-Adjacent Form* (w-NAF, for short) if it satis es that: (1) each nonzero digit is an odd integer with absolute value less than $2^{w\quad 1}$, and (2) for any w consecutive digits, at most one is nonzero [22].

Although w-NAF gives an ef cient exponentiation on a group admitting fast inversion, it is not useful for a group such as a multiplicative subgroup of a nite eld in which an inversion is much slower than a multiplication. We here introduce a generalized version of w-NAF with a digit set D. We remark that we may allow $w = 1$ by this generalization, and the sparse integer representation introduced in [11] is equivalent to 1-NAF with a digit set $D = \{1\}$ according to this de nition. However, if we consider a signed representation for elliptic curves, it is reasonable to restrict w as $w \geq 2$. The sparse signed integer representation in [11] corresponds to 2-NAF with $D = \{\pm 1\}$.

De nition 1 Let $w \geq 1$ be an integer and $D = \{\alpha_1, \alpha_2, \ldots, \alpha_{2^{w\ 1}}\}$ where α_is are nonzero odd integers and distinct modulo 2^w. A w-NAF with digit set D is a sequence of digits satisfying the following conditions:

1 Each digit is zero or an element in D.
2 Among any w consecutive digits, at most one is nonzero.

A w-NAF with digit set D is denoted by $a = (a_{m\ 1}a_{m\ 2}\ldots a_0)_2$ or $a = \sum_{i=\ 0}^{m\ 1} a_i 2^i$ where $a_i \in D \cup \{0\}$.

De nition 2 Let $a = (a_{m\ 1}a_{m\ 2}\ldots a_0)_2$ be a w-NAF with digit set D. Then the length of a, denoted by $\text{len}(a)$, is de ned to be the largest i such that $a_{i\ 1} \neq 0$. By notation, we let $\text{len}(0) = 0$. The number of nonzero digits in its representation is called the weight of a and denoted by $\text{wt}(a)$.

The uniqueness of the representation can be easily shown as follows. The argument is a generalization of Proposition 2.1 in [22].

Theorem 1 *Let r be a positive integer. All w-NAFs of length $\leq m$ with digit set D are distinct modulo r if $m \leq log_2(r/C)$ where $C = max\{|x \quad y| : x, y \in D \cup \{0\}\}$.*

Proof Suppose there are two different w-NAFs which represent the same integer. Let $(a_{\ 1}a_{\ 2}\cdots a_0)_2$ and $(b_{\ 1}b_{\ 2}\cdots b_0)_2$ are different representations such that

$$a = \sum_{i=0}^{\ 1} a_i 2^i = \sum_{i=0}^{\ 1} b_i 2^i \tag{1}$$

Assume is the smallest integer satisfying the above property.

If $a_0 = b_0$, we have two different and shorter w-NAFs which stand for the same integer. Thus it should be $a_0 \neq b_0$. If a is even, both of a_0 and b_0 should be zero and $a_0 = b_0$. It therefore should be odd and both of a_0 and b_0 should be nonzero. Since the representations are w-NAFs, $a_0 \neq 0$ and $b_0 \neq 0$ implies $a_1 = \cdots = a_w = 0$ and $b_1 = \cdots = b_w = 0$. By (1), we have $a_0 \equiv b_0$ mod 2^w. Since all elements in D are distinct modulo 2^w, we must have $a_0 = b_0$, which contradicts with the minimality of . Thus each integer has only one w-NAF with digit set D.

Moreover, let C_1 and C_2 be the maximal and minimal element in $D \cup \{0\}$. Then $C = C_1 \quad C_2$. The largest w-NAF of length $\leq m$ is less than $C_1 2^m$. The smallest w-NAF of length $\leq m$ is greater than $C_2 2^m$. Thus the difference between any two w-NAFs is less than $(C_1 \quad C_2)2^m = C2^m \leq r$ for $m \leq log_2(r/C)$. Therefore any two w-NAF of length $\leq m$ must be either distinct modulo r or identical.

Theorem 2 *The number of w-NAFs of length m and weight t with digit set D is*

$$\binom{m \quad (w \quad 1)(t \quad 1)}{t}|D|^t$$

Proof Consider an algorithm to choose t positions out of $m \quad (w \quad 1)(t \quad 1)$ positions and ll each of them with $w \quad 1$ consecutive zeros followed by an element in D. This algorithm gives a w-NAF of length $m + (w \quad 1)$. Then its rst $(w \quad 1)$ positions should be always zero since each nonzero digit is preceded by $(w \quad 1)$ consecutive nonzeros. By discarding the rst $(w \quad 1)$ zeros, we get a w-NAF of length $\leq m$. Since the algorithm covers all w-NAFs of length $\leq m$ and the algorithm outputs one of $\binom{m \quad (w \quad 1)(t \quad 1)}{t}|D|^t$ strings, we have the theorem.

Algorithm 1. Generation of w-NAF element of weight t

```
Input: $m$, $w$, $t$ and the digit set $D$
Output: $w$-NAF of length $\leq m$
1 Randomly choose $t$ positions out of $m \quad (w \quad 1)(t \quad 1)$ positions.
2 Fill each position by $(w \quad 1)$ consecutive zeros followed by a random element in $D$.
3 Discard the  rst $(w \quad 1)$ positions of the string.
4 Print the string which is a $w$-NAF of length $\leq m$.
```

From the proof of Theorem 2, we introduce Algorithm 1 to produce a random element in its w-NAF representation.

Using the set of w-NAFs with xed Hamming weight, we can perform ef cient batch veri cation of exponentiations on a group as in Algorithm 2. Here we use the simultaneous exponentiation method with online precomputation.

Algorithm 2. Batch veri cation of exponentiations using w-NAF exponents

```
Input: $m, w, t, D$, and $N$ exponentiation pairs $(x_i, y_i) \in \mathbb{Z}_r \times G$ for an abelian group $G$ of order $r$ with a generator $g$
Output: Accept or Reject
1: Take $N$ random exponents $c_1, c_2, \ldots, c_N$ from the set of $w$-NAFs of length $\leq m$ and weight $t$, where $c_i = \sum_{j=0}^{m-1} c_{ij} 2^j$ and $c_{ij} \in D \cup \{0\}$.
2: for $\alpha \in D$ do
3:     $y_{i,\alpha} \leftarrow y_i^\alpha$ for each $i$ $(1 \leq i \leq N)$ /* precomputation */
4: end for
5: $z \leftarrow 1$
6: for $j = m - 1$ downto 0 do
7:     $z \leftarrow z^2$
8:     for $i = 1$ upto $N$ do
9:         if $c_{ij} = \alpha \in D$ then
10:            $z \leftarrow z \cdot y_{i,\alpha}$
11:        end if
12:    end for
13: end for
14: Compute $g^c$ for $c = \sum_{i=1}^{N} c_i x_i \mod r$.
15: if $z = g^c$ then
16:     Accept all of $N$ instances
17: else
18:     Reject
19: end if
```

We need to take an appropriate digit set for each of speci c groups. For a multiplicative subgroup of a nite eld in which an inversion is much slower than a multiplication, we take $D = \{1, 3, \ldots, 2^w \quad 1\}$. Then its precomputation stage requires one squaring and $2^{w \quad 1} \quad 1$ multiplications for each i $(1 \leq i \leq N)$. If $w = 1$, we skip this stage. Steps 6–13 take $m \quad 1$ squarings and $tN \quad 1$ multiplications since each exponent has t nonzero digits. Hence the total complexity is $m + N \quad 1$ squarings and $N(t + 2^{w \quad 1} \quad 1) \quad 1$ multiplications plus one exponentiation using memory for $N(2^{w \quad 1} \quad 1)$ group elements. If $w = 1$, the number of squarings is only $m \quad 1$.

For an elliptic curve group in which a subtraction is as ef cient as an addition, we take $D = \{\pm 1, \pm 3, \ldots, \pm (2^{w \quad 1} \quad 1)\}$. In this case, the precomputation cost reduces to one elliptic doubling and $2^{w \quad 2} \quad 1$ elliptic additions for each i $(1 \leq i \leq N)$. Hence the total complexity is $m + N \quad 1$ elliptic doublings and $N(t + 2^{w \quad 2} \quad 1) \quad 1$ elliptic additions plus one scalar multiplication using memory for $N(2^{w \quad 2} \quad 1)$ elliptic curve points. If $w = 2$, the number of doublings is $m \quad 1$.

Table 2 presents the performance of batch veri cation over a nite eld and an elliptic curve and shows appropriate weight t for various w. For fair comparison of various veri cation methods, we set the group order r as $r \approx 2^{231}$ and the security level as around 115 bits throughout this paper. This security level corresponds to those of the standard curves B233 and K233, whose prime orders are 232 and 231 bits long, respectively. The length m is taken to be the largest integer to preserve the uniqueness of the representation. For example, we may set the parameters $(w, m, t) = (3, 228, 17)$ and verify N exponentiations using only about $20N$ multiplications and N squarings for a suf ciently large N. For an elliptic curve, we can use either $w = 3$ or $w = 4$ requiring only about $18N$ additions and N doublings. Note that *security* in Table 2 implies the security of the batch veri cation with the given parameters, which is computed as $\binom{m \quad (w \quad 1)(t \quad 1)}{t} 2^{(w \quad 1)t}$ by Theorem 2.

Table 2
Parameter selection and number of operations for batch veri cation on abelian groups.

Common				Finite eld		Elliptic curve	
w	m	t	Security	*Mem*	Complexity	*Mem*	Complexity
1	230	27	$2^{116.4}$	0	$(27N \quad 1)\mathscr{M}_f + (229)\mathcal{S}_f + 1\mathscr{E}_f$	–	–
2	230	21	$2^{116.0}$	N	$(22N \quad 1)\mathscr{M}_f + (229 + N)\mathcal{S}_f + 1\mathscr{E}_f$	0	$(21N \quad 1)\mathcal{A}_e + (229)\mathcal{D}_e + 1\mathscr{M}_e$
3	228	17	$2^{114.1}$	$3N$	$(20N \quad 1)\mathscr{M}_f + (227 + N)\mathcal{S}_f + 1\mathscr{E}_f$	N	$(18N \quad 1)\mathcal{A}_e + (227 + N)\mathcal{D}_e + 1\mathscr{M}_e$
4	227	15	$2^{116.8}$	$7N$	$(22N \quad 1)\mathscr{M}_f + (226 + N)\mathcal{S}_f + 1\mathscr{E}_f$	$3N$	$(18N \quad 1)\mathcal{A}_e + (226 + N)\mathcal{D}_e + 1\mathscr{M}_e$
5	226	13	$2^{116.0}$	$15N$	$(28N \quad 1)\mathscr{M}_f + (225 + N)\mathcal{S}_f + 1\mathscr{E}_f$	$7N$	$(20N \quad 1)\mathcal{A}_e + (225 + N)\mathcal{D}_e + 1\mathscr{M}_e$

Now we show a more general result on the ef ciency of sparse w-NAFs. To be precise, we give an asymptotic analysis on the time complexity of w-NAF-based batch veri cation for elliptic curves. First, we easily see that when the size of batch instance gets larger, the computation per one exponentiation becomes closer to $(t + 2^{w\quad 2})$ elliptic additions. Let us call it the Asymptotic Batch Complexity per Exponentiation (ABCE). For 2 security on a group of order r, we take the weight t using Theorem 2, so that

$$\binom{\log r \quad w \quad (w \quad 1)(t \quad 1)}{t} 2^{(w \quad 1)t} \geq 2 \quad , \tag{2}$$

where the length of the exponent is taken to be $m \approx \log r \quad w$ for the uniqueness as in Theorem 1. Using the inequality $\binom{2a}{t} \geq 2^{\alpha/t}$ and considering $\log r \geq 2$, we can show that Eq. (2) is satis ed by any $t \geq \quad /(w + logw)$. If we take the minimum t, the ABCE becomes $\frac{}{w+logw} + 2^{w \quad 2}$ $\leq \frac{2}{log}$, which is smaller by a factor of log than that of an individual veri cation, i.e., $O(l)$.

3.2. Batch veri cation using sparse complex w-NAFs

Consider an ordinary elliptic curve E de ned over $_q$ with $E(\quad_q) = q + 1 \quad t$ and $\gcd(q, t) = 1$, where t is the trace of E. The Frobenius map is de ned as follows:

$$E(\overline{\quad}_q) \rightarrow E(\overline{\quad}_q); (x, y) \mapsto (x^q, y^q),$$

where $\overline{\quad}_q$ is the algebraic closure of $_q$. The Frobenius map is a root of the characteristic equation $\chi_E(T) = T^2 \quad tT + q$ in the ring of endomorphisms End(E). We denote $E(\quad_{q^n})$ by the subgroup of $E(\overline{\quad}_q)$ consisting of $_{q^n}$-rational points. Let G be the subgroup of $E(\quad_{q^n})$ generated by P with a prime order r satisfying $r^2 \nmid \quad E(\quad_{q^n})$ and $r \nmid \quad E(\quad_q)$.

We now introduce a generalization of -adic NAF into -adic w-NAF, which was introduced in [27,28] on Koblitz curves. We remark that the set of sparse complex numbers, S_2, in [11] corresponds to -adic 2-NAF.

De nition 3 Let $w \geq 2$ be an integer. A -adic w-NAF is a sequence of digits satisfying the following two conditions:

1 Each nonzero digit is an integer which is not divisible by q and whose absolute value is less than $q^w/2$.
2 Among any w consecutive digits, at most one is nonzero.

A -adic w-NAF is denoted by $a = (a_m \quad _1 a_m \quad _2 \ldots a_0)$ or $a = \sum_{i=0}^{m \quad 1} a_i \quad ^i$. The length and the weight of a -adic w-NAF are de ned similarly to those of a w-NAF. Note that given a -adic w-NAF $a = (a_m \quad _1 a_m \quad _2 \ldots a_0)$ and a point Q over E, aQ is computed as $aQ = \sum_{i=0}^{m \quad 1} a_i \quad ^i(Q)$.

The following theorem shows the uniqueness of the -adic w-NAF representation.

Theorem 3 *Let $a = (a_m \quad _1, \ldots, a_0)$ and $b = (b_{m'} \quad _1, \ldots, b_0)$ be two -adic w-NAFs with $a_m \quad _1, b_{m'} \quad _1 \neq 0$. Then $aQ = bQ$ for some nonzero $Q \in G$ implies that $m = m'$ and $a_i = b_i$ for all i if*

$$max\{m, m \} \leq M_{q,r,w} = log_q \quad \frac{r}{(q^{w/2} + 1)^2} \quad (w \quad 1) \tag{3}$$

Proof Assume that there is a nonzero point $Q \in G$ such that $aQ = bQ$ for two distinct -adic w-NAFs $a = (a_m \quad _1, \ldots, a_0)$ and $b = (b_{m'} \quad _1, \ldots, b_0)$. By adding zero digits to the front of the strings, we may assume $m = m'$. Then we have $O = aQ \quad bQ = \sum_{i=0}^{m \quad 1} d_i \quad ^i(Q)$ for $d_i = a_i \quad b_i$.

Let $F(T) = \sum_{i=0}^{m \quad 1} d_i T^i$. Since $End(E)$ is an order of the imaginary quadratic eld, $F(\quad)$ can be considered as an element of $[i]$ divisible by r. Since $\chi_E(\quad) = 0$, $F(T)$ and $\chi_E(T)$ must have a common root in the algebraic closure of $_r$. Thus the resultant $R = Res(T^2 \quad tT + q, F(T))$ satis es $R \equiv 0 \bmod r$.

Let $_1$ and $_2$ be the roots of χ_E. Then $R = F(\quad_1)F(\quad_2)$ and $|\quad_1| = |\quad_2| = \sqrt{q}$. For each $\in \{\quad_1, \quad_2\}$, we have

$$|F(\quad)| \leq \sum_{i=0}^{m \quad 1} |d_i| | \quad|^i \leq \sum_{i=0}^{m \quad 1} |a_i| | \quad|^i + \sum_{i=0}^{m \quad 1} |b_i| | \quad|^i$$
$$\leq 2\left(\frac{q^w}{2} \quad 1\right)\left(\sqrt{q}^{m \quad 1} + \sqrt{q}^{m \quad 1 \quad w} + \cdots + \sqrt{q}^{m \quad 1 \quad mod \quad w}\right)$$
$$= 2\left(\frac{q^w}{2} \quad 1\right)\frac{\left(q^{(m+w \quad 1)/2} \quad 1\right)}{q^{w/2} \quad 1} \quad q^{(m+w \quad 1)/2}\left(q^{w/2} + 1\right)$$

Thus, $|R| \quad q^{m+w \quad 1}(q^{w/2} + 1)^2 \leq r$. Hence $R = 0$. Because χ_E is irreducible over , this implies $\chi_E | F(T)$ over .

Assume that d_{i_0} is the lowest nonzero coef cient of F. Then we can write

$$T \quad ^{i_0} F(T) = \left(g_0 + g_1 T + \cdots + g_{m \quad 3 \quad i_0} T^{m \quad 3 \quad i_0}\right) \chi_E(T)$$

for some $g_i \in$. By equating the coef cients, we know $d_{i_0} = qg_0$ and $d_{i_0+j} = qg_j \quad tg_{j \quad 1} + g_{j \quad 2}$ for $1 \leq j \leq w \quad 1$, where we set $g \quad _1 := 0$ by convention. Since each of a_{i_0} and b_{i_0} is not divisible by q, both a_{i_0+1} and b_{i_0+1} are nonzero. Hence $d_{i_0+1} = \cdots = d_{i_0+w \quad 1} = 0$. That is,

$$Eqn(j) = qg_j \quad tg_{j \quad 1} + g_{j \quad 2} = 0 \quad \text{for} \quad 1 \leq j \leq w \quad 1 \tag{4}$$

From $d_{i_0+1} = 0$ and $g \quad _1 = 0$, we have $q | g_0$ since $\gcd(q, t) = 1$. By repeating this procedure, we have $g_0, g_1, \ldots, g_w \quad _2$ are divisible by q.

After replacing g_i by g_i/q in the $Eqn(1), \ldots, Eqn(c \quad 1)$, we repeat the above procedure to obtain $q^2 | g_0, g_1, \ldots, g_w \quad _3$. At the end, we have $q^c | g_0$. Therefore, we have $q^w | d_{i_0}$. However, this is impossible since $|d_{i_0}| \quad q^w$.

The above theorem tells us that distinct -adic w-NAFs of length $m \quad M_{q,r,w}$ play the role of distinct group homomorphisms of G. Moreover, if $a = b$ in End(E), we have $R = Res(T^2 \quad tT + q, \sum_{i=0}^{m \quad 1}(a_i \quad b_i)T^i)$ satis es $R = 0$. By the same argument with Theorem 3, we have $a_i = b_i$ for all i regardless of k, which implies that every endomorphism of E has at most one -adic w-NAF.

Theorem 4 *The number of -adic w-NAFs of length $\leq m$ and weight t is*

$$\binom{m \quad (w \quad 1)(t \quad 1)}{t} 2^t \left(\frac{q^w}{2} \quad \frac{q^{w \quad 1}}{2}\right)^t$$

Proof As in Theorem 2, we consider an algorithm to choose t positions out of $m \quad (w \quad 1)(t \quad 1)$ positions and ll each of them by $w \quad 1$ consecutive zeros followed by an integer not divisible by q whose absolute value is less than $q^w/2$. By discarding the rst $(w \quad 1)$ zeros, we get a -adic w-NAF of length $\leq m$ with weight t. Conversely, any string with the property can be produced by the algorithm.

Now we count the number of cases. First we have $\binom{m \quad (w \quad 1)(t \quad 1)}{t}$ choices for t positions. Next, each position is lled by an integer x such that x is not divisible by q and $|x| \quad q^w/2$. The number of such integers is

$$2^t \left(\frac{q^w}{2} \quad \frac{q^{w \quad 1}}{2}\right)^t,$$

which completes the proof.

We introduce an algorithm to output a random secret exponent in a subgroup G of order r in an elliptic curve $E(\ \ _{q^n})$. Algorithm 3 produces uniformly distributed w-NAFs of length $\leq m$ with weight t if $m \leq M_{q,r,w}$.

Algorithm 3. (-adic w-NAF exponent of weight t)

Input: q, m, w, and t
Output: -adic w-NAF of length $\leq m$

1 Choose t positions out of $m \quad (w \quad 1)(t \quad 1)$ positions.
2 Fill each position by $(w \quad 1)$ consecutive zeros followed by an integer not divisible by q whose absolute value is less than $q^w/2$.
3 Discard the rst $(w \quad 1)$ positions of the string.
4 Print the string which is a -adic w-NAF of length $\leq m$.

Algorithm 4 describes a batch veri cation algorithm using -adic w-NAFs on a Koblitz curve, given (x_i, Q_i) for $1 \leq i \leq N$. For ease of notation, we only describe the algorithm for $q = 2$, but it can be easily extended to the general cases. For more details, Steps 2–12 compute

$$R[k] = \sum_{i=1}^{N} \sum_{j=0, c_{ij}=k}^{m \quad 1} sign\left(c_{ij}\right) \ ^j(Q_i)$$

for each odd integer $1 \leq k \leq 2^{w \quad 1} \quad 1$. Steps 13–19 compute

$$Q = \sum_{k=1,2\nmid k}^{2z \quad 1} kR[k] = (R[2z \quad 1] + \cdots + R[1])$$

$$+ 2((z \quad 1)R[2z \quad 1] + (z \quad 2)R[2z \quad 3] + 2R[5] + R[3])$$

for $z = 2^{w \quad 2}$, where the last term is computed using the BGMW method [8]. Steps 2–12 require $tN \quad 2^{w \quad 2}$ additions and Steps 13–19 require $2(z \quad 2) + 2 = 2z \quad 2 = 2^{w \quad 1} \quad 2$ additions plus one doubling. Hence the total complexity is $tN + 2^{w \quad 2} \quad 2$ additions plus one doubling with $2^{w \quad 2}$ memory. If $w = 2$, Steps 14–19 are not executed, and we save one doubling.

Algorithm 4 Batch veri cation using -adic w-NAFs on Koblitz curve $(q = 2)$

```
Input: $m, w, t$ and $N$ pairs $(x_i, Q_i) \in \mathbb{Z}_r \times G$ for a subgroup $G$ of order $r$ with a generator $P$
Output: Accept or Reject

1: Choose $N$ random elements $c_i = \sum_{j=0}^{m-1} c_{ij}\tau^j$ $(1 \leq i \leq N)$ from the set of $\tau$-adic $w$-NAFs of length $\leq m$ and
   weight $t$, where $c_{ij}$ is either 0 or an odd integer such that $|c_{ij}| < 2^{w-1}$.
2: for $1 \leq k \leq 2^{w-2}$ do
3:     $R[2k-1] \leftarrow O$
4: end for
5: for $j = m-1$ downto 0 do
6:     $R[2k-1] \leftarrow \tau(R[2k-1])$ for $1 \leq k \leq 2^{w-2}$
7:     for $i = 1$ upto $N$ do
8:         if $c_{ij} \neq 0$ then
9:             $R[|c_{ij}|] \leftarrow R[|c_{ij}|] + sign(c_{ij})(Q_i)$
10:        end if
11:    end for
12: end for
13: $Z \leftarrow R[2^{w-1}-1]$
14: $T \leftarrow R[2^{w-1}-1]$
15: for $k = 2^{w-2}-1$ downto 2 do
16:     $T \leftarrow T + R[2k-1]$
17:     $Z \leftarrow Z + T$
18: end for
19: $Z \leftarrow 2Z + T + R[1]$
20: Compute $cP$ for $c = \sum_{i=1}^{N} c_i x_i \mod r$.
21: if $Z = cP$ then
22:     Accept all of $N$ instances
23: else
24:     Reject
25: end if
```

Table 3
Number of point operations for batch veri cation over a Koblitz curve K233.

w	m	t	*Mem*	Complexity	Security
2	226	21	1	$(21N \quad 1)\mathcal{A}_e + 1\mathcal{M}_e$	$2^{115.4}$
3	225	18	2	$(18N)\mathcal{A}_e + 1\mathcal{D}_e + 1\mathcal{M}_e$	$2^{118.7}$
4	223	15	4	$(15N + 2)\mathcal{A}_e + 1\mathcal{D}_e + 1\mathcal{M}_e$	$2^{116.4}$
5	221	13	8	$(13N + 6)\mathcal{A}_e + 1\mathcal{D}_e + 1\mathcal{M}_e$	$2^{115.5}$
6	219	12	16	$(12N + 14)\mathcal{A}_e + 1\mathcal{D}_e + 1\mathcal{M}_e$	$2^{118.8}$
7	217	11	32	$(11N + 30)\mathcal{A}_e + 1\mathcal{D}_e + 1\mathcal{M}_e$	$2^{120.5}$
8	215	10	64	$(10N + 62)\mathcal{A}_e + 1\mathcal{D}_e + 1\mathcal{M}_e$	$2^{120.3}$
9	213	9	128	$(9N + 126)\mathcal{A}_e + 1\mathcal{D}_e + 1\mathcal{M}_e$	$2^{118.1}$
10	211	8	256	$(8N + 254)\mathcal{A}_e + 1\mathcal{D}_e + 1\mathcal{M}_e$	$2^{114.1}$
11	209	8	512	$(8N + 510)\mathcal{A}_e + 1\mathcal{D}_e + 1\mathcal{M}_e$	$2^{121.4}$
12	207	7	1024	$(7N + 1022)\mathcal{A}_e + 1\mathcal{D}_e + 1\ \mathcal{M}_e$	$2^{114.5}$

Algorithm 4 is an improved algorithm on Algorithm 4 in the conference version of this paper [12]. Although the number of point additions (and doublings) in Algorithm 4 is the same as that in [12], it signi cantly reduces the number of Frobenius maps, which is an important improvement from practical viewpoint. Note that Algorithm 4 scans the scalars c_i from left to right, i.e., from the most signi cant digit to the least significant digit. The original algorithm in [12] scans c_i in the opposite direction. As a result, in each iteration j, Algorithm 4 computes $2^{w \quad 2}$ Frobenius maps with the temporary results $R[2k \quad 1]$. Thus, the total number of Frobenius maps is $(m \quad 1)2^{w \quad 2}$. On the other hand, the algorithm in [12] should perform $\frac{(m \quad 1)tN}{2}$ Frobenius maps with points Q_i on average. If additional memory corresponding to N points is given, this number may be reduced to $(m \quad 1)N$ by storing $\ ^j(Q_i)$ in each iteration j and updating it in the next iteration with only one Frobenius map per Q_i. However, in an ordinary situation where $N \gg 2^{w \quad 2}$, Algorithm 4 has signi cant advantage over even this improved algorithm. In addition, it saves the memory for N points $\ ^j(Q_i)$. We will examine the practical meaning of this improvement in Section 5. For now, however, we will ignore the operations in complexity analysis for the sake of simplicity. Table 3 presents an appropriate weight t and the corresponding performance of batch veri cation using Algorithm 4 for various w over a Koblitz curve.

Now we compare the batch veri cation methods using sparse complex w-NAFs and sparse integer w-NAFs. The ABCE of Algorithm 4 is approximately $t + \frac{2^{w\ \ 2}}{N}$, while that of Algorithm 2 is $t + 2^{w\ \ 2}$. That is, the factor $2^{w\ \ 2}$ is amortized over the N signatures in Algorithm 4. Therefore, -adic w-NAFs guarantee a better performance than sparse integer w-NAFs. Using an asymptotic analysis similar to that given in Section 3.1, the ABCE for Algorithm 4 is $t \approx \ell/(w + log w)$, under the assumption that $2^{w\ \ 2} \ll tN$.

4 Batch veri cation of signatures

Our batch veri cation methods for exponentiations and scalar multiplications can be directly applied to batch veri cation of signatures. In this section, we describe our signature veri cation methods for two distinct cases, i.e., multiple signatures by a single signer and multiple signatures by multiple signers.

First, we consider N signatures (m_i, ρ_i, σ_i) signed by a single signer using public key y according to the notations given in Section 2.3. For batch veri cation, we take random sparse w-NAFs $c_1, \ldots, c_N$ and compute $a = \ \ \sum_{i=1}^{N} c_i m_i \sigma_i^{\ 1} mod r$ and $b = \ \ \sum_{i=1}^{N} c_i \rho_i \sigma_i^{\ 1} mod r$. Finally we compute

$$g^a y^b \prod_{i=1}^{N} \rho_i^{c_i} mod p, \tag{5}$$

and if it is 1 or p 1, we accept all N signatures. The crucial part of this veri cation procedure is the batch exponentiation, $\prod_{i=1}^{N} \rho_i^{c_i} mod p$ in Eq. (5). This term is ef ciently computed by Algorithm 2. In the modi ed ECDSA case, given N signatures (m_i, R_i, σ_i), batch veri cation is done by computing

$$aP + bQ + \sum_{i=1}^{N} c_i R_i, \tag{6}$$

where Q is the public key of the signer. If we use a Koblitz curve, c_i s are -adic w-NAFs and the batch term in Eq. (6) can be computed by Algorithm 4. We remark that the Boyd and Pavlovski attack [7] cannot be applied to the modi ed ECDSA.

Next, we explain the multiple-signer case. Given N signatures each of which is signed by a distinct signer with public key y_i, we take random w-NAFs $c_1, \ldots, c_N$ and compute $a = \ \ \sum_{i=1}^{N} c_i m_i \sigma_i^{\ 1} mod r$ and $b_i = \ \ c_i r_i \sigma_i^{\ 1} mod r$ for each i $(1 \le i \le N)$. Finally compute

$$g^a \prod_{i=1}^{N} y_i^{b_i} \prod_{i=1}^{N} \rho_i^{c_i} mod p, \tag{7}$$

and if it is 1 or p 1, accept all N signatures. In the modi ed ECDSA case, we will compute

$$aP + \sum_{i=1}^{N} b_i Q_i + \sum_{i=1}^{N} c_i R_i \tag{8}$$

Table 4
Performance of batch veri cations of modi ed DSA with an order-r subgroup ($r \approx 2^{231}$).

Method	Single signer	Multiple signers
Individual	$(77N)\mathcal{M}_f + (38N)\mathcal{S}_f$	$(92N)\mathcal{M}_f + (231N)\mathcal{S}_f$
ISET	$(30N + 92)\mathcal{M}_f + (N + 231)\mathcal{S}_f$	–
[11]	$(27N + 92)\mathcal{M}_f + 231\mathcal{S}_f$	–
Proposed (SIT)	$(20N + 92)\mathcal{M}_f + (N + 231)\mathcal{S}_f$	$(74N + 38)\mathcal{M}_f + (2N + 231)\mathcal{S}_f$

g^a: Lim–Lee, y^b: Lim–Lee (for single signer) or sliding window (for multiple signers), $\prod y_i^{b_i}$: simultaneous sliding window, $\prod \rho_i^{c_i}$: batch exponentiation.

Table 5
Performance of batch veri cations of modi ed ECDSA with a random elliptic curve (order $\approx 2^{231}$).

Method	Single signer	Multiple signers
Individual	$(77N)\mathcal{A}_e + (38N)\mathcal{D}_e$	$(84N)\mathcal{A}_e + (231N)\mathcal{D}_e$
ISET	$(26N + 84)\mathcal{A}_e + (N + 231)\mathcal{D}_e$	–
[11]	$(21N + 84)\mathcal{A}_e + 231\mathcal{D}_e$	–
Proposed (SIT)	$(18N + 84)\mathcal{A}_e + (N + 231)\mathcal{D}_e$	$(64N + 38)\mathcal{A}_e + (2N + 231)\mathcal{D}_e$

aP: Lim–Lee, bQ: Lim–Lee (for single signer) or window NAF (for multiple signers), $\sum b_i Q_i$: simultaneous window NAF, $\sum c_i R_i$: batch scalar multiplication.

Note that we may apply a traditional simultaneous exponentiation method for the second terms, i.e., $\prod_{i=1}^{N} y_i^{b_i} mod p$ in Eq. (7) and $\sum_{i=1}^{N} b_i Q_i$ in Eq. (8).

We now evaluate the batch veri cation costs of our methods and compare them with those of previous methods. Table 4 shows the number of eld operations required in various veri cation methods. We call the proposed integer w-NAF method the Sparse Integer Test (SIT). We rst consider the individual veri cation in the single-signer case. In this case, we must compute $g^a y^b$ mod p. Note that we may apply the Lim–Lee precomputation method [21] to compute g^a since g is xed. We reasonably assume $w = 6$, which means that 62 eld elements are precomputed and stored in memory. Then a modular exponentiation with a 231-bit exponent requires 39 multiplications and 38 squarings on the average. (If we do not count multiplication by 1, the number of multiplications is only 38. This reduction applies to all exponentiation algorithms and scalar multiplication algorithms throughout this paper.) For the individual veri cation with a single signer, we may also use the Lim–Lee method for y^b because the public key y will be reused multiple times. Then we amortize the cost for table construction over N veri cations, and this cost becomes negligible if N is suf ciently large. Thus the cost of a signature veri cation is two parallel executions of the Lim–Lee method plus one multiplication. Because squarings can be shared, the total cost is 77 multiplications plus 38 squarings. On the other hand, if the signatures are made by multiple signers, we cannot do off-line precomputation on y. Therefore we use a sliding window method with window size 5 for y^b, which requires 54 multiplications and 231 squarings. Then the cost for $g^a y^b$ mod p is 92 multiplications plus 231 squarings.

In batch veri cation, it is reasonable to use the Lim–Lee method for g^a, the sliding window method for y^b, and the simultaneous sliding window method for $\prod_{i=1}^{N} y_i^{b_i} mod$ p, as explained above. We assume $w = 6$ for the Lim–Lee method and $w = 5$ for the sliding window methods. We remark that at this point we ignore the costs of $_r$ operations for simplicity, because they only require signi cantly less computation compared to group operations. For the batch computation of $\prod_{i=1}^{N} \rho_i^{c_i} mod$ p, we take $w = 3$ which is the optimal choice according to Table 2.

In Table 4, we also included the small exponent test given in [30,4] and the method given in [11]. Actually, the method in [11] can be

Table 6
Performance of batch veri cations of modi ed ECDSA with a Koblitz curve (order $\approx 2^{231}$).

Method	Single signer	Multiple signers
Individual (with Frob.)	$(65N)\mathcal{A}_e$	$(78N)\mathcal{A}_e$
[11]	$(21N + 77)\mathcal{A}_e$	–
Proposed (SCT)	$(8N + 332)\mathcal{A}_e + 1\mathcal{D}_e$	$(54N + 286)\mathcal{A}_e + 1\mathcal{D}_e$

aP: xed-point -adic window NAF, bQ: -adic win. NAF (for multiple signers) or its xed-point version (for single signer), $\sum b_i Q_i$: simultaneous -adic window NAF, $\sum c_i R_i$: batch scalar multiplication.

Table 7
Implemented curves $y^2 + xy = x^3 + ax^2 + b$.

Name	(a, b)	Prime order	Cofactor	Field polynomial
K163	$(1, 1)$	163 bits	2	$f(z) = z^{163} + z^7 + z^6 + z^3 + 1$
K233	$(0, 1)$	232 bits	4	$f(z) = z^{233} + z^{74} + 1$
K283	$(0, 1)$	281 bits	4	$f(z) = z^{283} + z^{12} + z^7 + z^5 + 1$

viewed as a special case of the proposed method with $w = 1$, and the proposed method is its generalization. Although the original version of the small exponent test [30,4] only uses the binary method with short randomizers c_i for batch exponentiation, it can be easily improved to use simultaneous sliding window method. The figures displayed in Table 4 are estimated values using this improvement with $w = 4$, which requires only 30 multiplications and 1 squaring per signature. Let us call this method the Improved Small Exponent Test (ISET).

According to the table, verification of single-signer signatures using the proposed method is estimated to be faster than an individual verification and the best known method [11] by 5.48 and 1.29 times, respectively, under the assumption that N is sufficiently large and the cost of a squaring is almost the same as that of a multiplication. In the multiple-signer case, the proposed method is 4.24 times faster than an individual verification. However, we remark that these gains may be slightly reduced if there is sufficient memory for precomputed values and larger w is used for the Lim–Lee method in individual verification.

Table 5 compares the performance of various verification methods over a random elliptic curve with a 231-bit group order. As in the modified DSA, we use the Lim–Lee method with $w = 6$ for a scalar multiplication by a fixed point such as aP. If the point is not fixed, the window NAF method with $w = 5$ is the optimal choice for a 231-bit scalar, which requires 46 point additions and 231 doublings. For $\sum b_iQ_i$, we may share the doublings in the main loop of the window NAF method by adopting its simultaneous version. To compute $\sum c_iR_i$ by ISET, the simultaneous window NAF method with $w = 4$ is the optimal choice, because the length of a scalar is about a half of ordinary one as in the small exponent test [30,4], i.e., a scalar is 115 bits long. It requires only 26 point additions plus 1 doubling per signature. On the other hand, the proposed method computes $\sum c_iR_i$ using Algorithm 2. The optimal choice for w is $w = 4$ according to Table 2. We remark that the method in [11] can be viewed as a special case of the proposed method with $w = 2$.

A comparison using data in Table 5 reveals that batch verifications for a single signer and multiple signers using the proposed method are faster than individual verifications by 6.05 and 4.77 times, respectively, assuming that a point doubling has the same complexity as a point addition and N is sufficiently large. Thus the gains of batch verification over individual verification are slightly greater in the modified ECDSA than the modified DSA. However, the gain of the proposed method over [11] is greater in the modified DSA case.

Finally, Table 6 shows the results for a Koblitz curve. We call the proposed τ-adic w-NAF method the Sparse Complex Test (SCT). In this case, we use the τ-adic window NAF method instead of the window NAF method, which eliminates doubling operations resulting in 46 point additions for a general scalar multiplication with $w = 5$. In addition, we use the fixed-point τ-adic window NAF method instead of the Lim–Lee method for fixed points. If we set $w = 6$ so that 15 elements are precomputed, the cost for a scalar multiplication is only 33 point additions. For the proposed method, larger values of w decrease t and thus the tN term in the number of point additions according to Table 3. However, they exponentially increase the 2^{w-2} term at the same time. Therefore, the choice of w should be customized according to N as analyzed in Section 3.2. We reasonably assume that N has an order of 10^2, and choose $w = 10$. As a result, the number of point additions is reduced by up to 8.13 and 1.44 times in the signle-signer and the multiple-signer cases, respectively. We remark, however, that this reduction in the number of point additions does not always guarantee the same amount of speed-up in real implementations. We will discuss this issue in the following section.

5. Performance verification through implementation

In this section, we present experimental results to verify the performance gain of our batch verification methods. To directly compare the effects of sparse integer w-NAFs and τ-adic w-NAFs, we implemented both the SIT and SCT methods on Koblitz curves. We implemented three standard Koblitz curves K163, K233 and K283 on a system with an Intel Core i7 1.6 GHz CPU and 4 GB RAM. The underlying finite field arithmetic operations and big integer operations were done using MIRACL version 5.3.2 [26], and the elliptic curve arithmetic operations, signature generation and signature verification were customized for the verification methods explained in the previous section. We used SHA-1 as the hash function for signature, which is already implemented in MIRACL. We also used K163, K233 and K283 given by MIRACL, which are the implementation of NIST standard curves. The specific parameters are given in Table 7.

We obtained three major results from our experiments. First, the proposed methods significantly reduce the timings for signature verification as anticipated in the analysis given in the previous section. Another finding is that we must consider the cost of Frobenius maps in precise performance analysis, especially if the curves are implemented over finite fields that are represented using polynomial bases. Finally, the costs for secondary operations, such as hashing on each message, the $\mathbb{F}_r$ arithmetic including the inversion σ^{-1} mod r, and the complex arithmetic operations to deal with τ-expansions, cannot be ignored completely, especially in the case that the point operation cost per scalar is relatively small as in batch verification.

To explain these findings effectively, we begin with presenting a more detailed table for the amount of operations. Table 8 expands Table 6 and compares the individual verification method and batch verification methods, showing the numbers of required Frobenius maps as well as those of additions and doublings. This table also includes the figures for SIT to directly compare our two proposed methods, i.e., the SIT and SCT methods. Note that the number of operations in the SIT method in Table 8 is slightly smaller than that analyzed in Table 5, because the computation of bQ and $\sum_{i=1}^{N} b_iQ_i$ in Eqs. (6) and (8) can be improved

Table 8
Number of point operations required in batch verification of signatures (order $\approx 2^{231}$).

Method	Single signer	Multiple signers
Individual (with Frob.)	$(65N)\mathcal{A}_e + \left(460N^{\dagger}\right)\mathcal{F}_e$	$(78N)\mathcal{A}_e + \left(466N^{\dagger}\right)\mathcal{F}_e$
ISET	$(26N + 78)\mathcal{A}_e + (N + 115)\mathcal{D}_e + 466\mathcal{F}_e$	–
Proposed (SIT)	$(18N + 78)\mathcal{A}_e + (N + 226)\mathcal{D}_e + 466\mathcal{F}_e$	$(64N + 38)\mathcal{A}_e + (N + 226)\mathcal{D}_e + \left(236N^{\dagger} + 230\right)\mathcal{F}_e$
Proposed (SCT)	$(8N + 332)\mathcal{A}_e + 1\mathcal{D}_e + 54226^{*}\mathcal{F}_e$	$(54N + 286)\mathcal{A}_e + 1\mathcal{D}_e + \left(236N^{\dagger} + 53990^{*}\right)\mathcal{F}_e$

* For m = 211 and w = 10, $(m - 1) \times 2^{w-2} + 466 = 54{,}266$, $(m - 1) \times 2^{w-2} + 230 = 53{,}990$.
† We can reduce this term by 230N if simultaneous scalar multiplications can share Frobenius maps.

Table 9
Timings for signature verification on Koblitz curves (millisec).

Curves	Methods	Single signer			Multiple signers		
		$N=$ 100	$N=$ 500	$N=$ 1000	$N=$ 100	$N=$ 500	$N=$ 1000
K163	Individual (with Frob.)	221.6	1124.7	2241.7	260.6	1343.1	2561.5
	ISET	73.3	346.3	705.1	–	–	–
	Proposed (SIT)*	57.7	276.1	503.9	201.3	1010.9	1956.3
	Proposed (SCT)†	56.2	160.7	293.3	193.5	879.9	1739.4
K233	Individual (with Frob.)	441.5	2199.6	4492.8	503.9	2502.3	5055.9
	ISET	154.5	725.4	1441.4	–	–	–
	Proposed (SIT)*	124.8	530.4	1021.8	396.3	1932.8	3809.5
	Proposed (SCT)†	124.8	330.7	589.6	394.6	1723.8	3396.1
K283	Individual (with Frob.)	650.5	3297.8	6918.6	744.1	3686.2	7704.9
	ISET	223.1	1068.6	2123.2	–	–	–
	Proposed (SIT)*	173.1	758.2	1514.7	586.6	2812.6	5620.7
	Proposed (SCT)†	168.5	491.4	870.5	583.5	2522.5	4968.6

* $(w, m, t) = (3,157,12),(4,227,15)$ and $(4,277,18)$ for K163, K233 and K283, respectively.
† $(w, m, t) = (9,144,6),(10,211,8)$ and $(10,261,10)$ for K163, K233 and K283, respectively.

Table 10
Gap between the estimated and the measured timings (K233, $N = 1000$).

	Single signer			Multiple signers	
	Ind. (Frob.): SCT	ISET: SCT	SIT: SCT	Ind. (Frob.): SCT	SIT: SCT
Estimated ratio (Table 8)	7.89	2.71	1.92	1.46	1.16
Measured ratio (Table 9)	7.49	2.48	1.74	1.47	1.11

using τ-adic expansions even though the batch computation $\sum_{i=1}^{N} c_iR_i$ is still done based on the SIT method.

In the ISET method, we might use the τ-adic simultaneous sliding window method to compute $\sum_{i=1}^{N} c_iR_i$ so that c_is are expanded into a τ-adic form. However, it is easy to see that this modification does not give any performance gain, because a small integer exponent does not guarantee a short τ-adic expansion. It is well known that by converting a binary representation into a τ-adic representation, the sequence length is approximately doubled [27, 28]. In general, this problem is solved by using partial reduction with $\delta = \tau^{n-1} + \tau^{n-2} + \cdots + \tau + 1$, which reduces back the size of the τ-adic expansion into almost the same one as the bit length of original binary representation. This reduction is meaningless for small exponents because the norm of a small exponent is already small and its τ-adic expansion is almost not changed even after a reduction by δ. Therefore, we do not expand c_i into a τ-adic form, but leave it as an integer representation.

Now, we consider the influence of Frobenius maps on the overall performance. According to our experiments, one evaluation of the Frobenius map consumes approximately 0.031 times the time for a point addition, which is not completely negligible.[1] We first examine the single-signer case in Table 8, which shows that for sufficiently large N, the performance gain of SCT over the individual verification will be somewhat greater than the gain estimated using only the numbers of point additions. This is because the number of Frobenius maps in the individual verification is linear in N, while that in SCT is independent of N. However, because this fixed value is very large, the Frobenius maps are not in favor of SCT for small N. The break-even point seems to be around $N = 1000$. But it is easy to modify the loop structure of Algorithm 4 so that it may perform $(m - 1)N$ Frobenius maps instead of $(m - 1)2^{w-2}$, which is in fact the original algorithm in [12]. As a result, we may use the algorithm in [12] for small N, i.e., $N \leq 2^{w-2}$, while we use Algorithm 4 for large N. However, even after this optimization taking the sizes of N and w into account, the performance gain of SCT over SIT is expected to be slightly smaller than the value estimated based only on the numbers of point additions, because the SIT method requires only a small constant number of Frobenius maps. We can see a similar phenomenon in the multiple-signer case. Note that MIRACL adopts polynomial basis representations for the underlying binary field elements, which is a typical choice for software implementation of elliptic curves. If we may use normal basis representations, especially in hardware, the ratio of a Frobenius map to a point addition will be closer to 0, and the terms related to $\mathscr{F}_e$ in Table 8 may be ignored.

Table 9 summarizes our experimental results and compares the performance of batch verification methods. According to Table 9, larger values of N are favorable for batch verification as expected in Table 8. Table 9 shows that the proposed batch verification methods are significantly faster than individual verification and the small exponent test. To be precise, the fastest method, i.e., the batch verification using sparse τ-adic w-NAFs, accelerates the individual verification by a factor of 7.49 in the single-signer case, and by 1.47 in the multiple-signer case for $N = 1000$ over K233. We also see that our w-NAF method is up to 2.48 times faster than the ISET method in K233. Comparing our two proposed methods, we see that SCT outperforms SIT by 1.74 and 1.11 times in the single-signer and multiple-signer cases, respectively. We observe similar speed-ups by our methods in K163 and K283.

Even though the above results are almost consistent with the estimation given by Table 8, there are small gaps as shown in Table 10. According to our precise analysis, this phenomenon is explained by the behaviors of two overhead factors. The first factor is the fixed overhead for signature verification which includes hashing on each message and the $\mathbb{F}_r$ arithmetic operations such as multiplications and inversions in $m_i\sigma_i^{-1} mod\ r$ and $r_i\sigma^{-1} mod\ r$. The time consumed by these operations is almost independent of verification methods, and it is approximately 0.1 ms per signature, i.e., 100 ms for $N = 1000$, over K233. Therefore, it is a nonnegligible bottleneck in fast schemes such as SCT for a single signer. But this negative effect is not so significant for multiple signers.

The second factor to be considered is the complex arithmetic operations, i.e., the computation of τ-adic expansions of a and b for aP and bQ. The individual verification of N signatures requires $2N$ expansions in total, while a batch verification involves only two expansions in the single-signer case. In the multiple-signer case, SIT or SCT needs N expansions, which is only a half of the individual verification. Therefore, SIT and SCT gain some advantage over the individual verification.

In summary, the complex arithmetic operations have positive effects on the proposed schemes, while the fixed operations for signature verification have negative effects. Because the negative effects of the first factor dominate the positive effects in most cases, the measured gains are slightly smaller than the estimated gains except one case in Table 10. We remark that the ratio between the main operations and overheads may vary according to the implementation details such as the underlying big integer package as well as the level of optimization. However, the above analysis may be useful to identify the bottlenecks and the target for further optimization in each verification method.

6 Extensions

In this section, we show that the batch verifiers for pairing-based signatures can be further improved by applying our generalized sparse

[1] The ratio 0.031 was obtained by separately measuring the average timing for a Frobenius map and a point addition, which were 48.4 μsec and 1.5 μsec, respectively. Note that the literature already recognizes this ratio, although most papers ignore the cost for Frobenius maps. For example, we may recall a rough estimation, field inversion/multiplication = 8 and multiplication/squaring = 7, given in [18]. Because we are using affine coordinates, a Frobenius map is composed of two squaring operations over the binary field, and a point addition is composed of one inversion, two multiplications and one squaring. Therefore, the ratio of a Frobenius map to a point addition is $2/(8 \times 7 + 2 \times 7 + 1) \approx 0.028$.

exponents instead of small exponents. This improvement does not reduce the number of pairings, but it reduces the underlying elliptic curve operations. In this paper, we consider batching two pairing-based signatures dealt with in [9] and [16], i.e., the BLS signature [6] and the -IBS [9]. Although the description of these schemes was done with the symmetric bilinear map setting in [9], we will use a more general notation from [16] which uses asymmetric bilinear maps. For easier understanding of the underlying operations, we will use additive notation for the underlying groups for pairings. That is, a bilinear map is de ned as $e\ \ \mathbb{G}_1\times\mathbb{G}_2\rightarrow\mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are the additive groups generated by P_1 and P_2, respectively, and $\mathbb{G}_T$ is a multiplicative group.

6.1. Batching single-signer BLS signatures

Multiple instances of BLS signatures can be veri ed using only two pairings regardless of the number of signatures if they were signed by the same signer [6,9]. The equation for an individual veri cation of the BLS signature σ for a message M with public key Q is

$$e(\sigma,P_2)=e(H(M),Q), \tag{9}$$

where $H\ \ \{0,1\}^*\rightarrow\mathbb{G}_1$ is a hash function. Given N signatures σ_1, …, σ_N for messages M_1, …, M_N, the batch veri cation is done by verifying

$$e\ \ \sum_{i=1}^{N}c_i\sigma_i,P_2)=e\ \ \sum_{i=1}^{N}c_iH(M_i),Q), \tag{10}$$

where c_i are the random exponents. In [9] and [16], c_i were selected as small exponents, and its performance gain over individual veri cation was veri ed by implementation in [16].

Now, we estimate the expected performance for the case where c_i are chosen as generalized sparse exponents. We begin by counting the number of operations in Eqs. (9) and (10). We see that N iterations of Eq. (9) requires $2N$ pairings and N hashes. Note that we cannot ignore the overheads for hash computation, because these hashes are maps to the points over an elliptic curve, which involve nonnegligible square root computations over the underlying elds. On the other hand, Eq. (10) requires two pairings, two simultaneous scalar multiplications with N components, and N hashes. In other words, the batch veri cation signi cantly reduces the number of pairings at the cost of additional point operations. For example, for the case $N = 200$ over a curve MNT160, the veri cation time per signature is reduced from 47.6 ms to 2.28 ms over a 3.0 GHz Pentium D according to [16]. Because the time for one pairing was measured as 23.3 ms, we may conjecture that the time for a hash over the MNT160 curve is $47.6\quad 2\times 23.3 = 1.0$ ms. Then, the total time required for scalar multiplications in Eq. (10) will be $2.28\times 200\quad (2\times 23.3+200\times 1.0)=209.4$ ms, which is approximately 46% of the batch veri cation time and is not negligible. Now our task is to reduce this time using generalized sparse exponents. If the best algorithm, i.e., the simultaneous sliding window method, has been used to compute $\sum c_i\sigma_i$ and $\sum c_iH(M_i)$, the best choice would be $w = 4$ for 80-bit small exponents c_i, which requires approximately $2\times 20N$ point additions. If we use sparse exponents with $(w, m, t) = (3, 159, 12)$ which has the same security, this is reduced to approximately $2\times 14N$. As a result, we may expect that the overall time for batch veri cation should be reduced by approximately 13%. The above estimation is suf cient to show the usefulness of sparse exponents.

6.2. Batching multiple-signer Π-IBS

Camenisch, Hohenberger, and Pedersen [9] proposed a batch veri er for -IBS, an identity-based signature scheme secure in the standard model. This signature scheme, which was originally proposed by Waters [29] and extended by Chatterjee and Sarkar [10], was rediscovered in [9]. The equation for individual veri cation of the -IBS signature (S_1, S_2, S_3) for a message M with identity ID is

$$e(S_1,P_2)\cdot e\ \ S_2,\widehat{U}_1+\sum_{i=1}^{z}k_i\hat{U}_i)\cdot e\ \ S_3,\widehat{U}_2+\sum_{i=1}^{z}m_i\hat{U}_i)=A, \tag{11}$$

where $\widehat{U}_1$, $\widehat{U}_2$ and A are the public parameters, and m_i and k_i are obtained using the technique given in [24,10]. That is, the message M is rst hashed to a xed-length binary string, and the result is partitioned into z blocks as $h(M) = m_1||m_2||\cdots||m_z$. For example, in [16], h was implemented using SHA-1 and $z = 5$ was used, following [24]. The identity ID is similarly partitioned into $ID = k_1||k_2||\cdots||k_z$. Given N signatures $(S_{j,1}, S_{j,2}, S_{j,3})$ for messages M_j with identity ID_j for $j = 1, …, N (N \geq z/2)$, the batch veri cation equation is

$$e(\sum_{j=1}^{N}c_jS_{j,1},P_2)\cdot e(\sum_{j=1}^{N}c_jS_{j,2},\widehat{U}_1)\cdot e\sum_{j=1}^{N}c_jS_{j,3},\widehat{U}_2) \tag{12}$$
$$\cdot\prod_{i=1}^{z}e(\sum_{j=1}^{N}c_j\left(k_{j,i}S_{j,2}+m_{j,i}S_{j,3}\right),\hat{U}_i)=A^{\sum_{j=1}^{N}c_j},$$

where c_j are the random exponents, and $k_{j,i}$ and $m_{j,i}$ are obtained from ID_j and M_j, respectively.

The dominant operations in Eq. (12) are $z + 3$ pairings, $z + 3$ simultaneous scalar multiplications with N components using random exponents c_j, and zN simultaneous scalar multiplications $k_{j,i}S_{j,2} + m_{j,i}S_{j,3}$. (For simpler analysis, we ignore some point additions and multiplications over $\mathbb{G}_T$.) These two kinds of simultaneous scalar multiplications require $8\times 20N$ and $5N\times 50$ point additions for $z = 5$, respectively, if c_j are small exponents with 80 bits. Note that according to [16], the time for $z + 3 = 8$ pairings is 186.4 ms, which is only 10% of the batch veri cation of 200 signatures, 1888 ms, and most of the veri cation time is consumed by elliptic curve operations. By applying sparse exponents, we can reduce the number of point additions in the rst kind of scalar multiplication from $8\times 20N$ to $8\times 14N$. Thus we may estimate that the total batch veri cation time will be reduced by approximately 10%.

7 Conclusion

In this paper, we proposed an ef cient batch veri cation method of exponentiation and scalar multiplication using generalized sparse representations. By applying the proposed method, we signi cantly improved batch veri cation of multiple signatures. This improvement can be applied to not only the single-signer case, but also the multiple-signer case. We veri ed the performance of proposed methods through a full implementation as well as a theoretical analysis. It would be an interesting research direction to apply our result to various real-world applications involving many exponentiations, such as Mix-Net [1], proof of knowledge, anonymous authentication, and authenticated routing. In addition, it would also be an interesting issue to develop a fast and provable batch veri cation method for standard ECDSA signatures which guarantees a suf cient performance gain for single-signer and multiple-signer cases.

Acknowledgments

This research was supported in part by the MSIP, Korea, under the ITRC support program (NIPA-2014-H0301-14-1010) supervised by the NIPA and in part by Next-Generation Information Computing Development Program through the NRF funded by the MSIP, Korea (No. 2011-0029925).

References

[1] M. Abe, Mix-networks on permutation networks, Advances in Cryptology — Asiacrypt'99 LNCS, vol. 1716, Springer-Verlag, 1999. 258–273.

[2] J. Akinyele, M. Green, S. Hohenberger, M. Pagano, Machine-generated algorithms, proofs and software for the batch veri cation of digital signature schemes, ACM CCS '12, 2012, pp. 474–487.
[3] A. Antipa, D. Brown, R. Gallant, R. Lambert, R. Struik, S. Vanstone, Accelerated veri -cation of ECDSA signatures, SAC 2005 LNCS, vol. 38972006. 307–318.
[4] M. Bellare, J. Garay, T. Rabin, Fast batch veri cation for modular exponentiation and digital signatures, Proc. of Eurocrypt '98, LNCS, vol. 1403, Springer-Verlag, 1998, pp. 236–250 (Full version is available via http://cseweb.ucsd.edu/mihir/papers/batch.html).
[5] D. Bernstein, J. Doumen, T. Lange, J. Oosterwijk, Faster batch forgery identi cation, Proc. of Indocrypt, LNCS, vol. 7668, Springer-Verlag, 2012. 454–473.
[6] D. Boneh, B. Lynn, H. Shacham, Short Signatures from the Weil Pairing, J. Cryptol. 17 (2004) 297–319;
A preliminary version appeared, Proc. of Asiacrypt 2001, LNCS, vol. 2248, Springer-Verlag, 2001, pp. 514–532.
[7] C. Boyd, C. Pavlovski, Attacking and repairing batch veri cation schemes, Proc. of Asiacrypt 2000, LNCS, vol. 1976, Springer-Verlag, 2000, pp. 58–71.
[8] E. Brickell, D. Gordon, K. McCurley, D. Wilson, Fast exponentiation with precomputation, Eurocrypt '92 LNCS, vol. 658, Springer-Verlag, 1993. 200–207.
[9] J. Camenisch, S. Hohenberger, M. Pedersen, Batch veri cation of short signatures, J. Cryptol. 25 (2012) 723–747;
A preliminary version appeared, Proc. of Eurocrypt, LNCS 2007, vol. 4515, Springer-Verlag, 2007, pp. 573–590.
[10] S. Chatterjee, P. Sarkar, HIBE with short public parameters without random oracles, Proc. of Asiacrypt 2006, LNCS, vol. 4284, Springer-Verlag, 2006, pp. 145–160.
[11] J. Cheon, D. Lee, Use of sparse and/or complex exponents in batch veri cation of exponentiations, IEEE Trans. Comput. 55 (12) (2006).
[12] J. Cheon, J. Yi, Fast batch veri cation of multiple signatures, Proc. of PKC 2007, LNCS, vol. 4450, Springer-Verlag, 2007, pp. 442–457.
[13] Digital Signature Standard (DSS) (DSA, RSA, and ECDSA Algorithms)Available at http://csrc.nist.gov/cryptval/dss.htm.
[14] Public key cryptography for the nancial services industry: the elliptic curve digital signature algorithm (ECDSA), ANSI X9.62, 1999 (approved January 7, 1999).
[15] P. Feldman, A practical scheme for non-interactive veri able secret sharing, IEEE Symposium on Foundations of Computer Science, 1987, pp. 427–437.
[16] A. Ferrara, M. Green, S. Hohenberger, M. Pedersen, Practical short signature batch veri cation, Proc. of CT-RSA 2009, LNCS, vol. 5473, Springer-Verlag, 2009, pp. 309–324.
[17] D. Giry, J.-J. Quisquater, Cryptographic Key Length Recommendation, http://keylength.com/2011.
[18] D. Hankerson, A. Menezes, S. Vanstone, Guide to Elliptic Curve Cryptography, Springer, 2004.
[19] S. Karati, A. Das, D. Roychowdhury, B. Bellur, D. Bhattacharya, A. Iyer, Batch veri ca-tion of ECDSA signatures, Proc. of Africacrypt 2012, LNCS, vol. 7374, Springer-Verlag, 2012, pp. 1–18.
[20] S. Karati, A. Das, D. Roychowdhury, B. Bellur, D. Bhattacharya, A. Iyer, New algorithms for batch veri cation of standard ECDSA signatures, J. Cryptogr. Eng. 4 (2014) 237–258.
[21] C.H. Lim, P.J. Lee, More flexible exponentiation with precomputation, Proc. of Crypto '94, LNCS, vol. 839, Springer-Verlag, 1994, pp. 95–107.
[22] J. Muir, D. Stinson, Minimality and other properties of the width-w non-adjacent form, Math. Comput. 75 (2006) 369–384.
[23] E. Mykletun, M. Narasimha, G. Tsudik, Authentication and integrity in outsourced databases, Proc. of ISOC Symposium on Network and Distributed Systems Security (NDSSA04)2004.
[24] D. Naccache, Secure and practical identity-based encryption, IACR Cryptology ePrint Archive: Report 2005/3692005.
[25] D. Naccache, D. M'Raithi, S. Vaudenay, D. Raphaeli, Can D.S.A. be improved? Complexity trade-offs with the digital signature standard, Proc. of Eurocrypt '94, LNCS, vol. 950, Springer-Verlag, 1994, pp. 77–85.
[26] Certivox, MIRACL: Multiprecision Integer and Rational Arithmetic Cryptographic Library, http://certivox.com/ (We obtained the version we used for the implementation of this paper from the previous website of Shamus Software at http://www.shamus.ie/).
[27] J. Solinas, An improved algorithm for arithmetic on a family of elliptic curves, Proc. of Crypto 97, Springer-Verlag, 1997, pp. 357–371 (Full version is avaiable at http://www.cacr.math.uwaterloo.ca/techreports/).
[28] J. Solinas, Ef cient arithmetic on elliptic curves, Des. Codes Crypt. 19 (3) (2000) 195–249.
[29] B. Waters, Ef cient identity-based encryption without random oracles, Proc. of Eurocrypt 2005, LNCS, vol. 3494, Springer-Verlag, 2005, pp. 320–329.
[30] S. Yen, C. Laih, Improved digital signature suitable for batch veri cation, IEEE Trans. Comput. 44 (7) (1995) 957–959.