Improved base-ϕ expansion method for Koblitz curves over optimal extension fields

B. Chung, H. Kim and H. Yoon

Abstract: An improved base-ϕ expansion method is proposed, in which the bit-length of coefficients is shorter and the number of coefficients is smaller than in Kobayashi's expansion method. The proposed method meshes well with efficient multi-exponentiation algorithms. In addition, two efficient algorithms based on the proposed expansion method, named ϕ-wNAF and ϕ-SJSF, are presented which significantly reduce the computational effort involved in online precomputation by using the property of Frobenius endomorphism. The proposed algorithms noticeably accelerate computation of a scalar multiplication on Koblitz curves over optimal extension fields (OEFs). In particular, for OEFs where the characteristic is close to 32 bits or 64 bits, the required number of additions is reduced up to 50% in comparison with Kobayashi's base-ϕ scalar multiplication algorithm. Finally, a method that significantly reduces the memory usage of the precomputation table at the expense of slightly more computation is presented.

1 Introduction

The elliptic curve cryptosystem (ECC) has been recognised as a secure and efficient cryptosystem, since it was introduced by Miller [1] and Koblitz [2] independently in 1985. The ECC requires shorter key length than the cryptosystem on finite fields. In general, 160-bit elliptic curves are known to be as secure as 1375-bit finite fields [3]. This advantage enhances the adaptability of the ECC to resource-limited devices such as smart cards or wireless communication devices.

The main computation of the ECC is a scalar multiplication of a point on the elliptic curve. Because scalar multiplication is performed by iterative point additions and doublings, the performance of the ECC is proportional to the number of point operations. The method to reduce the number of such operations is similar to previous efforts to reduce the number of multiplications in modular exponentiation [4].

In an elliptic curve group, there is almost no difference in the amount of computation between addition and subtraction. Thus, we can adopt other methods that are not readily applicable to modular exponentiation such as a signed binary method, addition–subtraction chains or signed window methods. Moreover, the number of doublings can be considerably reduced if base-ϕ expansion methods based on the Frobenius endomorphism [5–11] or other endomorphisms [12, 13] are applied. These methods are known to be the most efficient approaches in terms of reducing the elliptic curve operations.

The base-ϕ expansion method using the Frobenius map was first proposed by Koblitz [5]. Koblitz's method is applicable only to elliptic curves defined over $\mathbb{F}_2$, $\mathbb{F}_{2^2}$, $\mathbb{F}_{2^3}$ and $\mathbb{F}_{2^4}$ finite fields. Müller [8] and Cheon *et al.* [9] extended the base-ϕ expansion method to elliptic curves defined over $\mathbb{F}_{2^r}$, where r is less than 6. In addition, Koblitz proposed a base-ϕ expansion method defined over finite fields whose characteristic is odd such as $\mathbb{F}_3$ and $\mathbb{F}_7$. Smart generalised the base-ϕ expansion method for elliptic curves defined over finite fields of the form $\mathbb{F}_q$ whose characteristic is odd and less than 128 [10]. Such kinds of elliptic curves are especially called Koblitz curves.

Nowadays, many embedded systems (such as cellular phones or handheld computers) and most desktop PCs use 32-bit μ-processors, and many servers and new powerful desktop PCs use 64-bit μ-processors. In 1998, Bailey and Paar [14, 15] proposed optimal extension fields (OEFs) of the form $\mathbb{F}_{p^m}$, which are suitable for software implementation in 32-bit or 64-bit μ-processors. In OEFs, the characteristic p is chosen as a large prime of special form less than but close to the word size of the processor ($|p| \simeq 32$ or 64 bits) for full utilisation of its built-in operations. Because of their higher characteristic, the base-ϕ expansion methods of Müller [8] and Smart [10] are not directly applicable to elliptic curves over OEFs. Kobayashi *et al.* [11] (Kobayashi for short) extended Müller's method to be applicable to elliptic curves over OEFs with a small reference table.

Kobayashi's base-ϕ expansion method converts the computation of a scalar multiplication into the sum of multiple scalar multiplications with relatively small m coefficients. In general, it is inefficient to compute each small scalar multiplication separately and then add them. Kobayashi's approach combines the basic multi-exponentiation algorithm with an optimisation technique that minimises the total sum of the Hamming weights of the coefficients. If, however, the more efficient multi-exponentiation algorithms are employed, they can yield better performance.

Kobayashi's expansion method has several points that require improvement. First, after the expansion, the bit-length of the coefficients is longer by 2 than that of the characteristic p. This results in additional memory usage for representing the coefficients in the case of OEFs whose characteristic is very close to the word size of the

doi:10.1049/iet-ifs:20060033

Paper first received 15th February and in revised form 1st September 2006

The authors are with the Computer Science Division, Department of EECS, Korea Advanced Institute of Science and Technology (KAIST), 373-1 Guseong-Dong, Yuseong-Gu, Daejeon 305-701, Republic of Korea

E-mail: bcchung@nslab.kaist.ac.kr

processor. Moreover, it increases the number of elliptic curve operations, which is proportional to the bit-length of p. Secondly, the optimisation technique is targeted for only basic multi-exponentiation algorithm. It does not maximise the performance when the more efficient multi-exponentiation algorithms, interleaved wNAF [16] and simultaneous simple joint sparse form (SJSF) [17], are employed.

Taking the above into consideration, we design an improved base-ϕ expansion method in which the bit-length of the coefficients is the same as that of p and the number of coefficients is $m-1$. Optimisation that involves eliminating one coefficient has advantages in both interleaved wNAF and simultaneous SJSF algorithms. In the former algorithm, the number of additions decreases significantly because it is proportional to the number of coefficients. In the latter algorithm, both the number of additions required for precomputation and the total joint Hamming weight decrease.

The computational effort involved in online precomputation can be significantly reduced by employing the basic concept of using the property of Frobenius endomorphism [12, 18]. On the basis of this consideration, we present two efficient algorithms, named interleaved ϕ-wNAF and simultaneous ϕ-SJSF. The combination of the proposed expansion method with these two algorithms yields far better performance for OEFs where the characteristic is close to 32 bits or 64 bits and the extension degree is relatively small. In particular, it reduces the required number of additions by 35–50% compared with Kobayashi's base-ϕ scalar multiplication algorithm. We note that, however, Gaudry [19] has recently proposed an index calculus method to solve the elliptic curve discrete logarithm problem (ECDLP) on elliptic curves defined over small extension fields $\mathbb{F}_{p^3}$ or $\mathbb{F}_{p^4}$, which is asymptotically faster than Pollard's rho method. Hence, this weakness should be dealt carefully to provide concrete security.

Finally, we present a method that significantly reduces the memory usage of the precomputation table at the expense of slightly more computation. This method enables a scalar multiplication to be performed with a reference table of less than 10 precomputed points.

2 Preliminaries

2.1 Optimal extension fields

An OEF is a finite field $\mathbb{F}_{p^m}$ such that

1. p is a pseudo-Mersenne prime of the form $2^n \pm c$ less than but close to the word size of the processor,
2. An irreducible binomial $f(x) = x^m - w$ exists over $\mathbb{F}_p$.

OEFs are suitable for software implementation because they fully utilise the computational capability of the processor. Moreover, the following conditions yield additional arithmetic advantages for subfield modular reduction and extension field modular reduction [14, 15]

$$2^n \equiv \pm c \bmod p$$
$$x^m \equiv w \bmod f(x)$$

2.2 Koblitz curves over OEFs

Let p be a prime, where $p > 3$, and let E be a non-supersingular elliptic curve defined over $\mathbb{F}_p$

$$E: y^2 = x^3 + ax + b \tag{1}$$

where $a, b \in \mathbb{F}_p$ and $4a^3 + 27b^2 \neq 0$. The Koblitz curve E is defined over $\mathbb{F}_p$ so as to admit the use of base-ϕ expansion methods in the scalar multiplication.

$E(\mathbb{F}_{p^m})$ is defined as a set of $\mathbb{F}_{p^m}$-rational points on the Koblitz curve E defined over $\mathbb{F}_p$. $E(\mathbb{F}_{p^m})$ consists of a point at infinity $\mathcal{O}$ and points of $(x, y) \in \mathbb{F}_{p^m} \times \mathbb{F}_{p^m}$ that satisfy (1). Then, $E(\mathbb{F}_{p^m})$ forms an additive abelian group with an identity element $\mathcal{O}$.

2.3 Frobenius map

Let $P = (x, y)$ be an $\mathbb{F}_{p^m}$-rational point on the Koblitz curve E defined over $\mathbb{F}_p$. The Frobenius map ϕ is then defined as

$$\phi: (x, y) \longrightarrow (x^p, y^p)$$

The Frobenius map ϕ is an endomorphism over $E(\mathbb{F}_{p^m})$ and satisfies

$$\phi^2 - t\phi + p = 0, \quad -2\sqrt{p} \leq t \leq 2\sqrt{p}$$

In a polynomial basis representation, the Frobenius map ϕ can be evaluated using $2(m-1)$ multiplications on $\mathbb{F}_p$ [11]. This takes negligible time compared with multiplication on $\mathbb{F}_{p^m}$ that needs m^2 multiplications on $\mathbb{F}_p$.

2.4 Base-ϕ scalar multiplication

Scalar multiplication is to compute kP for a random scalar multiplier k in the range $1 \leq k \leq \mathrm{ord}(P)$ and a point P that is an $\mathbb{F}_{p^m}$-rational point and is not an $\mathbb{F}_p$-rational point on the Koblitz curve E. Since the Frobenius map ϕ satisfies $\phi^2 - t\phi + p = 0$ and $\phi^m = 1$, the multiplier k can be expressed as a series of Frobenius maps with length m, which is referred to as a Type I expansion by Kobayashi [11]

$$k = \sum_{i=0}^{m-1} d_i \phi^i, \text{ where } |d_i| < \frac{3p}{2} \tag{2}$$

A Type I expansion can be further optimised using the property that $\sum_{i=0}^{m-1} \phi^i P = \mathcal{O}$ if $\gcd(\mathrm{ord}(P), \#E(\mathbb{F}_p)) = 1$ (the proof for this is given in [20]), which is referred to as a Type II expansion by Kobayashi. We can then rewrite (2) as follows

$$\sum_{i=0}^{m-1} d_i \phi^i = \sum_{i=0}^{m-1} (d_i + \alpha \ominus z)\phi^i = \sum_{i=0}^{m-1} c_i \phi^i, \quad \text{where } |c_i| < 3p \tag{3}$$

where $\ominus$ denotes a bitwise subtraction. Here, α is an integer that makes all $(d_i + \alpha)$s be non-negative and z is an integer that minimises the sum of the Hamming weights of c_is, that is $\sum_i w_H(c_i)$. In other words, if the Hamming weight of each column of $(d_i + \alpha)$s is greater than $m/2$, then all elements in the column should be complemented to reduce the Hamming weight to less than $m/2$, that is, $1 \rightarrow 0$ and $0 \rightarrow \bar{1}$.

By (3), scalar multiplication kP can be represented as follows

$$kP = \sum_{i=0}^{m-1} c_i P_i, \text{ where } P_i = \phi^i(P) \tag{4}$$

First, $P_i = \phi^i(P)$ for $0 \leq i < m$ is precomputed. Then, kP is computed by the table lookup method. The overall procedure is described in Algorithm 1.

Let L be the bit-length of $\max\{\|c_i\|: 0 \leq i < m\}$, and let m be an odd integer. The theoretical number of operations

required for Algorithm 1 with Type II expansion is

$$\left(\frac{2}{2^m}\cdot\sum_{i=0}^{\lfloor m/2\rfloor} i\binom{m}{i}\right)L$$

additions on average and $L-1$ doublings.

Algorithm 1: Kobayashi's base-ϕ scalar multiplication algorithm

Input: k, P, E, t, p where E is an elliptic curve, $t = tr(E)$

Output: $Q = kP$

Base-ϕ expansion of k

1-1 $i \leftarrow 0$, $x \leftarrow k$, $y \leftarrow 0$, $u_j \leftarrow 0$ for all j
1-2 **while** $x \neq 0$ or $y \neq 0$ **do**
1-3 $u_i \leftarrow x \bmod p$
1-4 **if** $u_i > p/2$ **then** $u_i \leftarrow u_i - p$
1-5 $v \leftarrow (x - u_i)/p$, $x \leftarrow tv + y$, $y \leftarrow -v$, $i \leftarrow i + 1$

Optimisation of base-ϕ expansion

2-1 $d_i \leftarrow u_i + u_{i+m} + u_{i+2m}$ for $0 \le i < m$
2-2 $c_i \leftarrow d_i + \alpha \ominus z$ for $0 \le i < m$, where $\ominus$ denotes a bitwise subtraction

Table reference multiplication

3-1 $P_i \leftarrow \phi^i(P)$ for $0 \le i < m$
3-2 $Q \leftarrow \mathcal{O}$, $L = \max\{\|c_i\|: 0 \le i < m\}$
3-3 **for** $j = L - 1$ **down to** 0 **do**
3-4 $Q \leftarrow 2Q$
3-5 **for** $i = 0$ **to** $m-1$ **do**
3-6 **if** $c_i[j] = 1$ **then** $Q \leftarrow Q + P_i$
3-7 **if** $c_i[j] = \bar{1}$ **then** $Q \leftarrow Q - P_i$

3 Multi-exponentiation algorithms

Multi-exponentiation in a commutative group is a common computation in many public-key cryptosystems especially for the verification of ElGamal, DSA or ECDSA signatures. Instead of computing each exponentiation separately, computing them all together yields better performance. In the ECCs, the multi-exponentiation corresponds to the evaluation of a sum $\sum_{1\le i\le d} k_iP_i$, where $d \ge 2$, each P_i is a point on E, and each $k_i \le \mathrm{ord}(P)$ is a scalar multiplier. We assume that k_i s are uniformly distributed random integers up to a respective maximum bit-length L. The notation $k_i[j]$ denotes the jth bit of k_i.

The basic algorithm looks at one bit of each k_i in left-to-right order. First, the previous result is doubled. For each k_i, P_i is then added to the intermediate result if $k_i[j] = 1$. The subsequent bits are performed similarly. Although the number of additions does not decrease, the number of doublings is $L-1$ independently of d. This advantage mainly accounts for the enhanced performance of multi-exponentiation algorithms compared with the use of separate computations.

Algorithm 2: Straus's simultaneous multi-exponentiation

Precomputation stage

Precompute the 2^d values $\sum_{i=1}^{d}\{0, 1\}P_i$ and store them in the table.

Evaluation Stage

1 $Q \leftarrow \mathcal{O}$
2 **for** $j = L-1$ **down to** 0 **do**
3 $Q \leftarrow 2Q$
4 **if** $(k_1[j], \ldots, k_d[j]) \neq (0 \cdots 0)$ **then**
5 $Q \leftarrow Q + \sum_i k_i[j]P_i$ {by table lookup}

If additional memory is available, a table reference method reduces the number of additions. First, an auxiliary table is precomputed from the elements P_i s in the precomputation stage. The final result is then computed with the help of this table in the evaluation stage. The well-known Straus's trick (also known as Shamir's trick) is a simple example of a table reference [21].

Each scalar integer k_i can be recoded into a signed binary representation, where each bit $k_i[j] \in \{-1, 0, 1\}$. In the ECC, this representation is meaningful because addition and subtraction have almost the same computation cost. For a single scalar integer, the non-adjacent form (NAF) [22] has the minimal Hamming weight among many different representations of k_i. Since the number of non-zero bits is minimised, the number of additions and subtractions is also minimised. On average, only $L/3$ bits of an NAF are non-zero. For multiple scalars, the joint sparse form (JSF) [23] minimises the number of non-zero columns, which is called the joint Hamming weight, and consequently reduces the number of additions.

The multi-exponentiation algorithms can be classified into two approaches: simultaneous exponentiation and interleaved exponentiation [16, 24]. The former combines all P_is with each other to make a table in the precomputation stage. All k_i s are then considered simultaneously in the evaluation stage. The latter makes separate tables for each P_i in the precomputation stage. The evaluation stage then utilises interleaving of the k_i s. In this section, we present an overview of two representative algorithms: a simultaneous SJSF algorithm and an interleaved wNAF algorithm.

3.1 Simultaneous SJSF algorithm

Recently, Grabner *et al.* [17] presented another joint representation called the SJSF. Like the JSF, this representation also has a minimal joint Hamming weight, but it is constructed in a simpler manner than the JSF. Grabner *et al.* [17] also presented a general algorithm to find the SJSF, k_i' s, of d integers $k_1, \ldots, k_d$. First, A_j for $0 \le j \le L$ is defined as

$$A_j = \{1 \le i \le d \mid k_i'[j] \neq 0\} \tag{5}$$

The algorithm then finds a SJSF that satisfies the following rule

$$A_{j+1} \supsetneq A_j \text{ or } A_{j+1} = \phi, \quad j \ge 0 \tag{6}$$

The joint Hamming weight of k_i's is then minimal among all joint representations.

After finding the SJSF, Straus's trick or its variation [25] can be used to compute $\sum_{1\le i\le d} k_i'P_i$. In this case, the number of additions needed in the evaluation stage is reduced to the minimal joint Hamming weight. However, $((3^d - 1)/2) - d$ additions, which are over a hundred for $d > 4$, are needed in the precomputation stage. Instead of finding the SJSF of all k_i s, combining two or three of k_i s into a unit block and then finding the SJSF of each block gives better performance.

Furthermore, the window method can be used with the SJSF. As noted in a previous work [17], however, the number of precomputed values increases exponentially as the window size w grows. Only the case $d = 2$, $w = 2$ is reasonable in current ECCs; other parameters require more than hundreds of additions.

3.2 Interleaved wNAF algorithm

The wNAF algorithm combines an NAF algorithm and a window method using a table reference [7, 26]. Given a window size w and an integer k_i, there exists a unique width-$(w+1)$ NAF $N_i[L], \ldots, N_i[0]$ such that

(wNAF-1) $k_i = \sum_{0 \le j \le L} N_i[j] \cdot 2^j$.
(wNAF-2) Each $N_i[j]$ is 0 or odd with an absolute value less than 2^w.
(wNAF-3) Among any $w+1$ consecutive components, at most one is non-zero.

The ordinary NAF is the case $w = 1$.

The average density, that is the proportion of non-zero components in N_i, is $1/(w+2)$. Thus, the expected number of additions for a scalar multiplication is $L/(w+2)$.

The interleaved wNAF algorithm [16] finds the wNAFs, N_i s, separately for each k_i. Then, the multi-exponentiation $\sum_{1 \le i \le d} k_i P_i$ is computed concurrently with N_i s. The precomputation stage requires a total of $d \cdot 2^{w-1}$ additions to make non-trivial table entries. The evaluation stage requires $d \cdot (L/(w+2))$ additions on average and L doublings.

4 Proposed base-ϕ scalar multiplication

In this section, we propose a base-ϕ expansion method in which the bit-length of the coefficients is the same as that of characteristic p and the number of coefficients is $m-1$. In addition, we present two efficient algorithms, named interleaved ϕ-wNAF and simultaneous ϕ-SJSF. These algorithms combine the proposed expansion method with two efficient multi-exponentiation algorithms, an interleaved wNAF and a simultaneous SJSF, respectively. In order to reduce the computational effort involved in online precomputation, we employ a basic technique that uses the property of Frobenius endomorphism [12, 18].

4.1 Improved base-ϕ expansion with Frobenius map

Our proposed base-ϕ expansion method is described in Algorithm 3. For a Type I expansion, we combine two parts of Kobayashi's method: expanding a scalar k as a series of base-ϕ with length $2m+4$ and reducing the length of the series into m. This enables the Type I expansion to be performed with m words memory. We also carefully deal with $w \leftarrow w - p$ operation in consideration of d_i, which always guarantees the condition $|d_i| < p/2$. The following theorem provides the foundation for the proposed expansion.

Theorem 1: All coefficients d_i s of Type I expansion in Algorithm 3 always satisfy the condition $|d_i| < p/2$.

Proof 1: Without loss of generality, we assume that $|d_i| < p/2$ and $0 \le w < p$ in step 1-4 of Algorithm 3.

If $d_i + w < p/2$, then $w' = w$, so that $d_i + w' = d_i + w < p/2$ and $d_i + w' > -p/2$ since $w' \ge 0$ and $d_i > -p/2$. Hence, $|d_i + w'| < p/2$.

Algorithm 3: (ϕ-EXP) Proposed base-ϕ expansion method

Input: k, E, t, p where E is an elliptic curve, $t = \mathrm{tr}(E)$
Output: $\{c_0 \ldots c_{m-2}\}$, where $|c_i| < p$

[Type I] Base-ϕ expansion of k
1-1 $i \leftarrow 0$, $x \leftarrow k$, $y \leftarrow 0$, $d_j \leftarrow 0$ for $0 \le j < m$
1-2 **while** $x \neq 0$ or $y \neq 0$ **do**
1-3 $w \leftarrow x \bmod p$
1-4 **if** $(d_i + w) > p/2$ **then** $w \leftarrow w - p$
1-5 $d_i \leftarrow d_i + w$
1-6 $v \leftarrow (x - w)/p$, $x \leftarrow tv + y$, $y \leftarrow -v$
1-7 $i \leftarrow i + 1 \bmod m$

[Type II] Optimisation of base-ϕ expansion
2-1 $c_i \leftarrow d_i - d_{m-1}$ for $0 \le i \le m-2$

If $d_i + w > p/2$, then $w' = w - p$, so that $d_i + w' = d_i + w - p$. Since $d_i + w > p/2$ and $0 \le w < p$, $p/2 < d_i + w < 3p/2$. Therefore $-p/2 < d_i + w - p < p/2$. Hence, $|d_i + w'| < p/2$.

Therefore a scalar k is expanded as a series of base-ϕ with length m

$$k = \sum_{i=0}^{m-1} d_i \phi^i, \quad \text{where } |d_i| < \frac{p}{2} \tag{7}$$

Similar to Kobayashi's Type II expansion, we utilise the property that $\sum_{i=0}^{m-1} \phi^i P = \mathcal{O}$ if $\gcd(\mathrm{ord}(P), \#E(\mathbb{F}_p)) = 1$. We can then rewrite (7) as a base-ϕ series with length $m-1$

$$\sum_{i=0}^{m-1} d_i \phi^i = \sum_{i=0}^{m-1} (d_i - d_{m-1}) \phi^i = \sum_{i=0}^{m-2} c_i \phi^i, \quad \text{where } |c_i| < p \tag{8}$$

Our Type II expansion is targeted for OEFs whose characteristic is close to the word size of 32-bit or 64-bit μ-processors. In this case, the extension degree of OEFs will be relatively small due to their higher characteristic. For reasonable security, elliptic curves should have a cyclic subgroup of over 160-bit prime order [3]; hence, a reasonable extension degree of OEFs whose characteristic is close to 32 bits or 64 bits is 7 or 5, respectively, for current and near future levels of security. A smaller extension degree may be allowed for lower levels of security, depending on the application. Gaudry's index calculus method [19], however, can solve an ECDLP defined over $\mathbb{F}_{p^3}$ in time $O(p^{4/3})$ with a reasonably small constant instead of $O(p^{3/2})$ for Pollard's rho. In consideration of this weakness, the case of extension degree 3 for 64-bit μ-processors should be applied carefully.

Given that the extension degree is relatively small, we effectively focus on adapting the multi-exponentiation techniques into the scalar multiplication rather than merely minimising the total sum of the Hamming weights of the coefficients. The simultaneous SJSF multi-exponentiation algorithm computes five coefficients in the form of (2, 3) and computes seven coefficients in the form of (2, 2, 3). If, however, one coefficient is eliminated, the algorithm gains computational advantage by computing in the form of (2, 2), (3, 3) or (2, 2, 2); in other words, the joint Hamming weight decreases due to the use of small blocks of equal size. This gives another advantage when the Frobenius endomorphism is introduced for online precomputation, which is explained later. In addition, since the performance of the interleaved wNAF multi-exponentiation algorithm is proportional to the number of coefficients, it is logical to reduce one coefficient.

4.2 Scalar multiplication with interleaved ϕ-wNAF

We propose a ϕ-wNAF scalar multiplication algorithm that combines our base-ϕ expansion method with an interleaved wNAF multi-exponentiation algorithm. We also introduce a

Table 1: Expected number of operations in one block

Algorithm	Parameter		Precomputaiton		Evaluation	
	d	w	Add	ϕ	Add	Double
SJSF	2	1	2	—	$(1/2)L$	$L-1$
	2	2	10	—	$(3/8)L$	$L-1$
	3	1	10	—	$(23/39)L$	$L-1$
ϕ-SJSP	3	1	8	2[†]	$(23/39)L$	$L-1$

[†] $\phi P+\phi^2 P=\phi(P+\phi P)$, $\phi P-\phi^2 P=\phi(P-\phi P)$

method that considerably reduces the computational effort involved in online precomputation. This algorithm is described in Algorithm 4.

Algorithm 4: (ϕ-wNAF) Scalar multiplication with interleaved ϕ-wNAF

Input: k, P, E, t, p, w where E is an elliptic curve, $t=\mathrm{tr}(E)$

Output: $Q=kP$

Precomputation stage:
{For $0\le i\le m-2$, precompute $T_i[j]=j\phi^i P$ for all odd j s.t. $1\le j<2^w$.}
1-1 $T_0[1]\leftarrow P$; $X\leftarrow 2P$
1-2 **for** $j=3$ **to** 2^w-1 **by** 2 **do**
1-3 $T_0[j]=T_0[j-2]+X$
1-4 **for** $i=1$ **to** $m-2$ **do**
1-5 **for** $j=1$ **to** 2^w-1 **by** 2 **do**
1-6 $T_i[j]=\phi(T_{i-1}[j])$

Evaluation stage:
2-1 $\{k_0,\ldots,k_{m-2}\}\leftarrow\phi$-EXP($k$, E, t, p)
2-2 $\{N_0,\ldots,N_{m-2}\}\leftarrow w$NAF($\{k_0,\ldots,k_{m-2}\}$)
2-3 $Q\leftarrow\mathcal{O}$, $L=\max\{\|N_i\|: 0\le i\le m-2\}$
2-4 **for** $j=L-1$ **down to** 0 **do**
2-5 $Q\leftarrow 2Q$
2-6 **for** $i=0$ **to** $m-2$ **do**
2-7 **if** $N_i[j]>0$ **then** $Q\leftarrow Q+T_i[N_i[j]]$
2-8 **if** $N_i[j]<0$ **then** $Q\leftarrow Q-T_i[-N_i[j]]$

After the precomputation table T_0 is first computed for P, that is jP for all odd j such that $1\le j<2^w$, the other precomputation tables, T_i s, for $1\le i\le m-2$ are computed in this manner

$$T_i[j]=\phi(T_{i-1}[j])\quad\text{for all odd } j\quad\text{s.t. } 1\le j<2^w$$

Thus, only one table precomputation for P and dozens of inexpensive ϕ operations are required for all precomputations.

In the precomputation stage, the normal wNAF multi-exponentiation algorithm requires $(m-1)\cdot 2^{w-1}$ elliptic curve additions, whereas the ϕ-wNAF algorithm requires just 2^{w-1} elliptic curve additions and $(m-2)\cdot 2^{w-1}$ ϕ operations.

In the evaluation stage, the ϕ-wNAF algorithm uses $(m-1)\cdot(L/(w+2))$ additions on average and $L-1$ doublings. On the other hand, if Kobayashi's base-ϕ expansion is applied, the ϕ-wNAF algorithm uses $m\cdot((L+1)/(w+2))$ additions on average and L doublings.

4.3 Scalar multiplication with simultaneous ϕ-SJSF

We also propose a ϕ-SJSF scalar multiplication algorithm that combines our base-ϕ expansion method with a simultaneous SJSF multi-exponentiation algorithm. Like the ϕ-wNAF algorithm, the ϕ-SJSF can reduce the computational effort involved in online precomputation. It only precomputes for one block; precomputations for the other blocks are performed with the Frobenius map. In the evaluation stage, the ϕ-SJSF algorithm becomes more complicated; specifically, it entails simultaneous processing within an inner block and interleaved processing between inter blocks. This algorithm is described in Algorithm 5.

In the precomputation stage, the required number of operations varies depending on the block size and window size, as summarised in Table 1 [24]. In the evaluation stage, the ϕ-SJSF algorithm uses the sum of additions required for each block and $L-1$ doublings. The number of additions required for each block is summarised in Table 1.

5 Performance comparison

In this section, we analyse the required number of elliptic curve operations over $\mathbb{F}_{(2^{32}-5)^5}$, $\mathbb{F}_{(2^{31}-1)^7}$, $\mathbb{F}_{(2^{61}-1)^3}$, and $\mathbb{F}_{(2^{61}-1)^5}$. We also compare the expected number of operations when SJSF, ϕ-SJSF, wNAF, and ϕ-wNAF algorithms are applied to either Kobayashi's Type II expansion or the proposed expansion. Since the coefficients of Kobayashi's Type II expansion are not uniformly distributed, it is very difficult to analyse the precise complexity. Thus, we evaluated the expected number of operations through simulation, the results of which are shown in Tables 2–5. For each algorithm, we chose the parameter that minimised the number of operations. In the SJSF and ϕ-SJSF algorithms, the parameter (2,3) implies that we must find the SJSF of

Table 2: Expected number of elliptic curve operations over $\mathbb{F}_{(2^{32}-5)^5}$

Algorithm	Kobayashi's expansion					Proposed expansion				
	Parameter	Precomputation		Evaluation		Parameter	Precomputation		Evaluation	
		Add	ϕ	Add	Double		Add	ϕ	Add	Double
Basic	—	—	4	52.0	33	—	—	3	63.0	31
SJSF	(2, 3)	12	4	33.5	33	(2, 2)	4	3	32.4	32
ϕ-SJSF	(2, 3)	10	12	33.5	33	(2, 2)	2	9	32.4	32
wNAF	$w=2$	10	4	35.5	33	$w=2$	8	3	33.1	32
ϕ-wNAF	$w=4$	8	32	25.0	33	$w=4$	8	24	22.3	32

Table 3: Expected number of elliptic curve operations over $\mathbb{F}_{(2^{31}-1)^7}$

Algorithm	Kobayashi's expansion					Proposed expansion				
	Parameter	Precomputation		Evaluation		Parameter	Precomputation		Evaluation	
		Add	ϕ	Add	Double		Add	ϕ	Add	Double
Basic	—	—	6	78.8	32	—	—	5	92.0	30
SJSF	(2, 2, 3)	14	6	49.3	32	(2, 2, 2)	6	5	47.6	31
ϕ-SJSF	(2, 2, 3)	10	20	49.3	32	(3, 3)	8	43	37.2	31
wNAF	$w = 2$	14	6	51.8	32	$w = 2$	12	5	48.1	31
ϕ-wNAF	$w = 4$	8	48	36.3	32	$w = 4$	8	40	32.6	31

the first two coefficients and separately find the SJSF of the other three coefficients. The two SJSFs are subsequently used in the evaluation stage. In the wNAF and ϕ-wNAF algorithms, the parameter w denotes window size.

Algorithm 5: (ϕ-SJSF) Scalar multiplication with simultaneous ϕ-SJSF

Input: k, P, E, t, p, w, d where E is an elliptic curve, $t = tr(E)$

Output: $Q = kP$

Precomputation stage:

{For $0 \le i < (m-1)/d$, precompute $T_i[I_0 \cdots I_{d-1}] = \sum_{0 \le j \le d-1} I_j \phi^j P$ for some d-tuples $(I_0, \ldots, I_{d-1}) \in \{-(2^w - 1), \ldots, 2^w - 1\}$ depending on d and w.}

1-1 $T_0[I_0 \cdots I_{d-1}] \leftarrow \sum_{0 \le j \le d-1} I_j \phi^j(P)$ for some $(I_0, \ldots, I_{d-1})$
1-2 **for** $i = 1$ **to** $(m-1)/d - 1$ **do**
1-3 $T_i[j] = \phi^d(T_{i-1}[j])$ for all j

Evaluation stage:

2-1 $n \leftarrow (m-1)/d$
2-2 $\{k_0, \ldots, k_{m-2}\} \leftarrow \phi$-EXP($k$, E, t, p)
2-3 $\{N_{d \cdot i}, \ldots, N_{d \cdot i + d - 1}\} \leftarrow$ SJSF($\{k_{d \cdot i}, \ldots, k_{d \cdot i + d - 1}\}$) for $0 \le i \le n - 1$
2-4 $Q \leftarrow \mathcal{O}$, $L = \max\{\|N_i\|: 0 \le i \le m - 2\}$
2-5 **for** $i = 0$ **to** $n-1$ **do**
2-6 $window\_handle_i \leftarrow$ nil
2-7 **for** $j = L-1$ **down to** 0 **do**
2-8 $Q \leftarrow 2Q$
2-9 **for** $i = 0$ **to** $n-1$ **do**
2-10 **if** $window\_handle_i =$ nil and $(N_{d \cdot i}[j], \ldots, N_{d \cdot i + d - 1}[j]) \neq (0 \cdots 0)$ **then**
2-11 $J \leftarrow j - w + 1$
2-12 **while** $(N_{d \cdot i}[j], \ldots, N_{d \cdot i + d - 1}[j]) = (0 \cdots 0)$ **do**
2-13 $J \leftarrow J + 1$
2-14 {now $j \ge J > j - w$ and $J \ge 0$}
2-15 $window\_handle_i \leftarrow J$
2-16 $I_{i,0} \cdots I_{i,d-1} \leftarrow N_{d \cdot i}[j \cdots J] \cdots N_{d \cdot i + d - 1}[j \cdots J]$
2-17 **if** $window\_handle_i = j$ **then**
2-18 $Q \leftarrow Q + T_i[I_{i,0} \cdots I_{i,d-1}]$
2-19 $window\_handle_i \leftarrow$ nil

Without applying any algorithm, the proposed expansion requires many more additions compared with Kobayashi's expansion. However, the combination of the proposed expansion with any one of the algorithms shows better performance, as shown in the tables. In the case of using SJSF and ϕ-SJSF, the required number of additions in the precomputation stage is smaller because the number of considered coefficients is smaller than that in Kobayashi's. Moreover, the number of additions in the evaluation stage is reduced because the joint Hamming weight of two coefficients is lower than that of three coefficients. In the case of using wNAF and ϕ-wNAF, the number of additions in the evaluation stage is reduced significantly due to the decreased number of interleaved coefficients.

The ϕ-SJSF and ϕ-wNAF algorithms reduce the number of additions to make table entries in the precomputation stage by performing ϕ operations instead of additions. Because the ϕ operation is $\sim$20–30 times faster than an addition operation [15], this technique contributes significantly to improving the performance. In the ϕ-SJSF, this makes a window size of $w = 2$ an optimal parameter. In the ϕ-wNAF, this enlarges the optimal window size and consequently reduces the number of additions in the evaluation stage.

In all cases, the combination of the proposed expansion with ϕ-wNAF shows the best performance. In particular, the number of total additions decreases to just 50–65% of Kobayashi's basic algorithm. The combination of the proposed expansion with ϕ-SJSF shows far better performance than Kobayashi's algorithm and requires only a few more additions than the ϕ-wNAF algorithm.

6 Memory usage

The SJSF, ϕ-SJSF, wNAF and ϕ-wNAF algorithms create the table entries and store them in the precomputation stage. The case requiring the largest table size is the ϕ-wNAF over $\mathbb{F}_{(2^{31}-1)^7}$, and a total of 48 points are stored.

Table 4: Expected number of elliptic curve operations over $\mathbb{F}_{(2^{61}-1)^3}$

Algorithm	Kobayashi's expansion					Proposed expansion				
	Parameter	Precomputation		Evaluation		Parameter	Precomputation		Evaluation	
		Add	ϕ	Add	Double		Add	ϕ	Add	Double
Basic	—	—	2	46.1	62	—	—	1	60.0	60
SJSF	(3)	10	2	33.8	62	(2)	2	1	30.2	61
ϕ-SJSF	(3)	8	4	33.8	62	(2)	2	1	30.2	61
wNAF	$w = 3$	12	2	29.1	62	$w = 3$	8	1	24.3	61
ϕ-wNAF	$w = 4$	8	16	25.2	62	$w = 4$	8	8	20.2	61

Table 5: Expected number of elliptic curve operations over $\mathbb{F}_{(2^{61}-1)^5}$

Algorithm	Kobayashi's expansion					Proposed expansion				
	Parameter	Precomputation		Evaluation		Parameter	Precomputation		Evaluation	
		Add	ϕ	Add	Double		Add	ϕ	Add	Double
Basic	—	—	4	97.5	62	—	—	3	121.0	60
SJSF	(2, 3)	12	4	63.4	62	(2, 2)	4	3	61.4	61
ϕ-SJSF	(2, 3)	10	12	63.4	62	(2, 2)[†]	10	25	45.9	61
wNAF	$w=3$	20	4	55.6	62	$w=3$	16	3	49.6	61
ϕ-wNAF	$w=4$	8	32	47.5	62	$w=4$	8	24	41.4	61

[†]Window size $w=2$

If the amount of available memory is restricted, the table size can be reduced by simply storing the table entries for P (eight points in this case) in the precomputation stage, that is $\mathrm{Tab}[j]=jP$ for all odd integers j such that $1 \le j < 2^w$; other table entries are computed in the evaluation stage by applying ϕ operations to existing entries without storing all entries. However, all table entries do not need to be computed for all iterations. Since a non-zero component appears only once within a $(w+1)$-width window, it is sufficient to compute some non-zero entries of the following for each iteration j

$$\begin{gathered}\phi(\mathrm{Tab}[N_1[j]]),\\ \phi^2(\mathrm{Tab}[N_2[j]]),\\ \vdots\\ \phi^{m-2}(\mathrm{Tab}[N_{m-2}[j]])\end{gathered}$$

The total number of ϕ operations required in the evaluation stage is an average of

$$\frac{L}{w+2}\left(\sum_{i=1}^{m-2} i\right)$$

For the above case of the ϕ-wNAF over $\mathbb{F}_{(2_{31}-1)^7}$, an additional 80 ϕ operations are required in the evaluation stage. However, the ϕ operations required to make other table entries are no longer needed in the precomputation stage (in this case, 40 ϕ operations). Thus, this approach significantly reduces the table size at the expense of slightly more computations.

Furthermore, if the algorithm is performed with a left-to-right signed recording scheme such as wMOF [27] on the fly, then further reduction in memory usage will be possible.

7 Conclusions

In this paper, we have proposed an improved base-ϕ expansion method in which the bit-length of coefficients is shorter and the number of coefficients is smaller than in the Kobayashi's expansion method. Moreover, we have presented two efficient algorithms, an interleaved ϕ-wNAF and a simultaneous ϕ-SJSF, which reflect the properties of Frobenius endomorphism, and thus reduce the amount of online precomputation. The combination of the proposed expansion with these algorithms noticeably accelerates the computation of a scalar multiplication on Koblitz curves over OEFs. In particular, for OEFs, where the characteristic is close to 32 or 64 bits, the required number of additions is reduced by up to 50% compared with Kobayashi's base-ϕ scalar multiplication algorithm.

8 Acknowledgments

This work was supported by the Ministry of Science and Technology (MOST)/Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center (AITrc) and the Ministry of Information and Communication (MIC), Republic of Korea, under the Information Technology Research Center (ITRC) support program supervised by the Institute of Information Technology Assessment (IITA), (IITA-2006-C1090-0603-0075).

9 References

1 Miller, V.: 'Use of elliptic curve in cryptography'. Proc. Advances in Cryptology – CRYPTO'85, 1985, (*Lect. Notes Comput. Sci.*, **218**), pp. 417–426
2 Koblitz, N.: 'Elliptic curve cryptosystems', *Math. Comput.*, 1987, **48**, pp. 203–209
3 Lenstra, A.K.: 'Selecting cryptographic key sizes', *J. Cryptol.*, 2001, **14**, pp. 255–293
4 Gordon, D.: 'A survey of fast exponentiation methods', *J. Algorithms*, 1998, **27**, pp. 129–146
5 Koblitz, N.: 'CM-curves with good cryptographic properties'. Proc. Advances in Cryptology – CRYPTO'91, 1991, (*Lect. Notes Comput. Sci.*, **576**), pp. 279–287
6 Meier, W., and Staffelbach, O.: 'Efficient multiplication on certain nonsupersingular elliptic curves'. Proc. Advances in Cryptology – CRYPTO'92, 1992, (*Lect. Notes Comput. Sci.*, **740**), pp. 333–344
7 Solinas, J.A.: 'An improved algorithm for arithmetic on a family of elliptic curves'. Proc. Advances in Cryptology – CRYPTO'97, 1997, (*Lect. Notes Comput. Sci.*, **1294**), pp. 357–371
8 Müller, V.: 'Fast multiplication on elliptic curves over small fields of characteristic two', *J. Cryptol.*, 1998, **11**, pp. 219–234
9 Cheon, J., Park, S., and Kim, D.: 'Two efficient algorithms for arithmetic of elliptic curves using Frobenius map'. Proc. 1st Int. Workshop – PKC'98, 1998, (*Lect. Notes Comput. Sci.*, **1431**), pp. 195–202
10 Smart, N.P.: 'Elliptic curve cryptosystems over small fields of odd characteristic', *J. Cryptol.*, 1999, **12**, pp. 141–151
11 Kobayashi, T., Morita, H., Kobayashi, K., and Hoshino, F.: 'Fast elliptic curve algorithm combining Frobenius map and table reference to adapt to higher characteristic'. Proc. Advances in Cryptology – EUROCRYPT'99, 1999, (*Lect. Notes Comput. Sci.*, **1592**), pp. 176–189
12 Gallant, R.P., Lambert, R.J., and Vanstone, S.A.: 'Faster point multiplication on elliptic curves with efficient endomorphisms'. Proc. Advances in Cryptology – CRYP'2001, 2001, (*Lect. Notes Comput. Sci.*, **2139**), pp. 190–200
13 Ciet, M., Lange, T., Sica, F., and Quisquater, J.-J.: 'Improved algorithms for efficient arithmetic on elliptic curves using fast endomorphisms'. Proc. Advances in Cryptology – EUROCRYPT'03, 2003, (*Lect. Notes Comput. Sci.*, **2656**), pp. 388–400
14 Bailey, D., and Paar, C.: 'Optimal extension fields for fast arithmetic in public-key algorithms'. Proc. Advances in Cryptology – CRYPTO'98, 1998, (*Lect. Notes Comput. Sci.*, **1462**), pp. 472–485
15 Bailey, D., and Paar, C.: 'Efficient arithmetic in finite field extensions with application in elliptic curve cryptography', *J. Cryptol.*, 2001, **14**, pp. 153–176
16 Möller, B.: 'Algorithms for multi-exponentiation'. Proc. Selected Areas in Cryptography – SAC'01, 2001, (*Lect. Notes Comput. Sci.*, **2259**), pp. 165–180

17 Grabner, P.J., Heuberger, C., Prodinger, H., and Thuswaldner, J.M.: 'Analysis of linear combination algorithms in cryptography', *ACM Trans. Algorithms*, 2005, **1**, (1), pp. 123–142
18 Sarkar, P., Mishra, P.K., and Barua, R.: 'New table look-up methods for faster Frobenius map based on scalar multiplication over GF(p^n)'. Proc. Applied Cryptography and Network Security – ACNS'04, 2004, (*Lect. Notes Comput. Sci.*, **3089**), pp. 479–493
19 Gaudry, P.: 'Index calculus for Abelian varieties and the elliptic curve discrete logarithm problem'. Cryptology ePrint Archive: Report 2004/073, 2004
20 Kobayashi, T.: 'Base-ϕ method for elliptic curves over OEF', *IEICE Trans. Fundam.*, 2000, **E83-A**, (4), pp. 679–686
21 Straus, E.G.: 'Addition chains of vectors (problem 5125)', *Am. Math. Mon.*, 1964, **71**, pp. 806–808
22 Jedwab, J., and Mitchell, C.J.: 'Minimum weight modified signed-digit representations and fast exponentiation', *Electron. Lett.*, 1989, **25**, pp. 1171–1172
23 Solinas, J.A.: 'Low-weight binary representations for pairs of integers', Combinatorics and Optimization Research Report CORR 2001-41 Report, 2001
24 Avanzi, R.M.: 'The complexity of certain multi-exponentiation techniques in cryptography', *J. Cryptol.*, 2005, **18**, pp. 357–373
25 Yen, S.-M., Laih, C.-S., and Lenstra, A.K.: 'Multi-exponentiation', *IEE Proc., Comput. Digit. Tech.*, 1994, **141**, pp. 325–326
26 Solinas, J.A.: 'Efficient arithmetic on Koblitz curves', *Des. Codes. Cryptogr.*, 2000, **19**, pp. 195–249
27 Okeya, K., Schmidt-Samoa, K., Spahn, C., and Takagi, T.: 'Signed binary representations revisited'. Proc. Advances in Cryptology – CRYPTO'04, 2004, (*Lect. Notes Comput. Sci.*, **3152**), pp. 123–139