Analysis of the Hamming Weight of the Extended wmbNAF

Ming Li, Ali Miri *Senior Member, IEEE* and Daming Zhu

***Abstract*—Scalar multiplication is an important operation in elliptic curve cryptosystems(ECC). The algorithms for computing scalar multiplication are mostly based on the binary expansions of scalars, such as the non-adjacent form (NAF) and wNAF(sliding window method). Representing scalars using more bases can speed up the scalar multiplication, such as mbNAF, wmbNAF and extended wmbNAF, which was proposed by Longa and Miri in 2008. In this paper, we give a formal analysis of the Hamming weight of the extended wmbNAF method for scalar multiplication on general elliptic curves over large prime fields. Then the cost of this method is compared with NAF and other double-base methods. The analysis shows that we obtain the most efficient algorithm when using $(2,3,5)$NAF$_{1,1,0}$, which is 9.0% faster than the NAF method without extra storage requirement. Moreover, the recoding algorithm of the extended wmbNAF method is just as simple and fast as that of the NAF method.**

***Index Terms*—elliptic curve cryptography, multibase representation, scalar multiplication.**

I. Introduction

Elliptic curve cryptosystem(ECC), which was independently proposed by Miller [2] and Koblitz [1], is based on the elliptic curve discrete logarithm problem (ECDLP). Scalar multiplication [3], [4] is the central and most time-consuming operation in ECC. The traditional methods for computing scalar multiplication are mostly based on the binary expansions of scalars, such as double-and-add, NAF, and wNAF(sliding window) method [3]. The Non-Adjacent Form(NAF) of the scalar helps to speed up the computation of scalar multiplication because of its low Hamming weight, which is equal to the number of point additions. Some other redundant representations are also well studied, such as the radix 4 Booth method [16]. Another approach for reducing the Hamming weight is to represent the scalars using more bases. In 2008, Longa and Miri [9] proposed multi-base NAF(mbNAF), w-multi-base NAF(wmbNAF) and extended w-multi-base NAF(extended wmbNAF), which have properties analogous to NAF, to get some representations with much lower Hamming weight.

In this paper, we compute the average Hamming weight of extended wmbNAF first, which is closely related to the cost of scalar multiplication. Then we compare the average computational cost of this method with the NAF, wNAF and other double-base(multi-base) methods. The analysis shows that when we are working on prime field F_p with $|p| =$ 160bits, the $(2,3,5)$NAF$_{1,1,0}$ is the most efficient method without extra storage requirements. If there is enough space for extra storage, the $(2,3,5)$NAF$_{4,1,0}$ method provides the best perfromance. In the rest part of this paper, F_p denotes the prime field with character p and $|p| = n$ means $\lfloor log_2 p \rfloor = n$.

M. Li is with the School of Computer Science and Technology, Shandong University, China, e-mail: luaming@msn.com. In 2009, he was a visitor in the School of Information Technology and Engineering(SITE), University of Ottawa.

A. Miri is with the School of Information Technology and Engineering(SITE), University of Ottawa, Canada, and the School of Computer Science, Ryerson University, Canada.

D. Zhu is with the School of Computer Science and Technology, Shandong University, China.

II. Backgrounds

An elliptic curve [4] E over a field K is defined by the equation following

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

where $a_1, a_2, a_3, a_4, a_6 \in K$. Since the set $E(K)$ of the rational points (x, y) on the curve E and the special point O at infinity form an abelian group, we design elliptic curve cryptosystems on this point group.

Scalar multiplication sP plays a central role in these cryptosystems, where s is an integer and P is a point on the curve. Signed digit expansions with low Hamming weight are important in implementing scalar multiplication, such as NAF and wNAF. An expression $s = (\ldots, s_2, s_1, s_0)$, where $s_i \in \{-1, 0, 1\}$, is called the non-adjacent form (NAF) of the integer s, if it satisfies $s_i \cdot s_{i+1} = 0$ for $i \in N$. The Hamming weight of s is the number of non-zero digits in s. The expected Hamming weight of a random integer less than $2^n - 1$ is $n/2$. The Hamming weight is reduced to $n/3 + O(1)$ on average if we use NAF to represent the integer. Then the cost of the scalar multiplication is reduced from $nD + \frac{n}{2}A$ to $nD + \frac{n+1}{3}A$, where D and A denote point doubling and point addition on the elliptic curve. The wNAF uses larger signed digits set to reduce the Hamming weight of the scalar further, but it needs precomputations and more storage space. The wNAF is represented as $s = (\ldots, s_2, s_1, s_0)$ with the property that $s_i \cdot s_{i+1} \cdots s_{i+w-1} = 0$, where $s_i \in \{-\frac{2^w-1}{2}, \ldots, -1, 0, 1, \ldots, \frac{2^w-1}{2}\}$. wNAF is also called sliding window method [3], and the window size w fixes the modulus in the recoding algorithm for wNAF to be 2^w.

A. Double-base Chain

One way to reduce the Hamming weight of the scalar is by representing it with more bases. The double-base chain (DB-chain) [5], [6] is one such approach. It expresses the scalar s as $\sum_{i=1}^{l} s_i 2^{a_i} 3^{b_i}$, where $s_i \in \{-1, 1\}$, $a_1 \geq \cdots \geq a_l \geq 0$, $b_1 \geq \cdots \geq b_l \geq 0$ and $l \in Z$. In [5], [6], an integer is recoded using a greedy algorithm, and the analysis showed that the number

Algorithm 1. Extended Double-base Chain Greedy Algorithm

Input: An integer s, parameters a_0 and b_0 and a set S.

Output: The extended double-base chain $\sum_{i=1}^{l} d_i 2^{a_i} 3^{b_i}$ of s.

1. $i = 1; t = s; sign = 1$;
2. while $t > 0$ do
3. find the best approximation $z = d_i 2^{a_i} 3^{b_i}$ of s with $d_i \in S$, $0 \le a_i \le a_{i-1}$ and $0 \le b_i \le b_{i-1}$;
4. $d_i = sign \times d_i$;
5. if $t < z$ then $sign = -sign$;
6. $t = |t - z|$; $i = i + 1$;
7. Return (d_i, a_i, b_i);

of terms in a double-base chain is at most $\frac{n}{logn}$ over F_p with $|p| = n$. Then the computational cost of the double-base chain method is less than $nD+(log_2 3)nT+\frac{n}{logn}A$, where T is point tripling. It is however difficult to formally analyze the average computational cost of this method because of the nature of the greedy algorithm to produce the double-base chain. The cost of the "greedy" algorithm itself is another problem, although most of time it converges fast enough. In case the length of the recoded double-base chain is too long, some restrictions of the exponents can be used to limit the highest power of the output in the recoding algorithm. These restrictions varies in different cases. For example, the restrictions for elliptic curves on large prime fields are different from that for elliptic curves on binary fields. While working with different curves or fields with different formulas, experimental results have to be used to estimate the restrictions required.

If we use the window technique in the recoding algorithm that produces a double-base chain, we get the extended double-base chain [8]. Algorithm 1 is the recoding algorithm for extended double-base chain method. The parameters a_0 and b_0 are the restrictions of the highest power in the output. S is a set of window values. If $S = \{-1, 1\}$, Algorithm 1 is the recoding algorithm for ordinary double-base chain.

When we use three based 2, 3 and 5 in the representation of a scalar, we have a multi-base chain [7], which is called SMBR. The multi-base chain is also produced by a greedy algorithm that is similar to one in Algorithm 1.

The tree-based approach is another method to get the double-base chain [13]. The similar idea is also used for multi-scalar multiplication [14]. The tree-based DB-chain approach is faster than the ordinary DB-chain method when we have enough space for algorithms, because the tree-based recoding algorithm requires more space for the "prediction". For example, if we predict 4 steps in the recoding algorithm in [13], the extra space requirement is 8×sizeof(scalar). Then the extra storage requirement is 8×160 bits when the scalar is 160 bits, which is equal to the storage space for 4 points.

B. extended wmbNAF

Recently, Longa and Miri proposed multi-base non-adjacent form(mbNAF), window multi-base non-adjacent form(wmbNAF) and extended window multi-base non-adjacent form(extended wmbNAF) [9] to represent the scalar using multi-bases. The digit string $s = (s_m^{(a_{i_m})}, \ldots, s_2^{(a_{i_2})}, s_1^{(a_{i_1})})$ is called the extended wmbNAF of an integer if there are at least $(w_1 + \ldots + w_j)$ "0" digits between any two non-zero digits, where $s_l \in \{c | c = z \bmod a_1^{w_1} \cdots a_j^{w_j}, c \neq 0, z \in \mathbb{Z}\} (0 \le l \le m)$, with bases $a_{i_l} \in A = \{a_1, \ldots, a_j\}$ and window set $W = \{w_1, \ldots, w_j\}$. When $a_1 = a_i$, where a_i is one of the bases used (for example $a_i = 2$), $w_1 = 2$ and $w_2 = \ldots = w_j = 0$, s is mbNAF. When $a_1 = a_i$, $w_1 > 2$ and $w_2 = \ldots = w_j = 0$, s is wmbNAF. Therefore the mbNAF and wmbNAF can be treated as special cases of the extended wmbNAF. Algorithm 2 is the recoding algorithm for the extended wmbNAF, also for mbNAF and wmbNAF.

Algorithm 2. extended wmbNAF recoding

Input: An integer s, bases $A = a_1, \ldots, a_j$, where $a_j \in Z^+$ are primes, the modulus $a = a_1^{w_1} \cdots a_j^{w_j}$, $W = \{w_1, \ldots, w_j\}$ is the window set, where $w_j \ge 0$.

Output: The $(a_1, \ldots, a_j)\text{NAF}_{w_1,\ldots,w_j}(s) = (\ldots, s_1^{(a_{i_1})}, s_0^{(a_{i_0})})$.

1. $i = 0$
2. while $s \ge 0$ do
2.1 if s mods $a_1 = 0$ then $s = s/a_1, s_i = 0^{(a_1)}$
2.2 elseif s mods $a_2 = 0$ then $s = s/a_2, s_i = 0^{(a_2)}$
...
2.3 elseif s mods $a_j = 0$ then $s = s/a_j, s_i = 0^{(a_j)}$
2.4 else $s_i = s(\text{mods } a)$, $s = s - s_i$
2.4 $i = i + 1$
3. Return $(\ldots, s_1^{(a_{i_1})}, s_0^{(a_{i_0})})$

The binary/ternary method [11], albeit generated using a different algorithm, can be considered as a special case of extended wmbNAF method. When $a_1 = 2$, $a_2 = 3$, $w_1 = 1$ and $w_2 = 1$, Algorithm 2 is the recoding algorithm for a binary/ternary method.

We use $(a_1, a_2, ...)\text{NAF}_{w_1,w_2,...}$ to represent the extended wmbNAF for short. For example, $(2,3)\text{NAF}_{1,1}$ means an expansion based on 2 and 3 with the window set $W = \{w_1 = 1, w_2 = 1\}$ in Algorithm 2. If we work with $(2,3)$NAF, we have $a_1 = 2$, $a_2 = 3$, $w_1 = 2$ and $w_2 = 0$.

III. Analysis

In this section, we analyze the average Hamming weight of the extended wmbNAF in Theorem 1 using Markov chain [17], which is a tool for modeling randomized algorithms. If a Markov chain with a finite state space is aperiodic and irreducible, then there is a stationary distribution for the state vector in the Markov chain.

Theorem 1: Over the prime field F_p with $|p| = n$, the average Hamming weight of the extended wmbNAF is $\frac{n}{(\frac{1}{a_1-1}+w_1)log_2 a_1+\ldots+(\frac{1}{a_j-1}+w_j)log_2 a_j}$, and the average number of digits based on a_i is $\frac{(\frac{1}{a_i-1}+w_i)n}{(\frac{1}{a_1-1}+w_1)log_2 a_1+\cdots+(\frac{1}{a_j-1}+w_j)log_2 a_j}$, where bases set $A = \{a_1, a_2, \ldots, a_j\}$ and the windows set $W = \{w_1, w_2, \ldots, w_j\}$ for $1 \le i \le j$.

Proof:

Assume that a sequence of states $S_0, \ldots, S_r$ appears, and let $\pi_{i,j}$ denote the probability that the state following S_i is S_j. Then $\pi_{i,j}$ depends only on S_i and S_j, not on the states before S_i. Then we have a $r \times r$ transition matrix $\Pi = (\pi_{i,j})_{i,j=0}^{r}$. Let $\sigma_0, \ldots, \sigma_r$ denote the probabilities that states $S_0, \ldots, S_r$ occur, then $\sigma_0, \ldots, \sigma_r$ sum to 1. We define $j + 1$ different states as follows.

TABLE I
THE TRANSITION MATRIX IN THEOREM 1

$$\Pi = \left(\begin{array}{c|c|c|c|c|c} \frac{1}{a_1} & (1-\frac{1}{a_1})\frac{1}{a_2} & (1-\frac{1}{a_1})(1-\frac{1}{a_2})\frac{1}{a_3} & \cdots & (1-\frac{1}{a_1})\ldots(1-\frac{1}{a_{j-1}})\frac{1}{a_j} & 1-(\frac{1}{a_1}+(1-\frac{1}{a_1})\frac{1}{a_2}\ldots)) \\ \hline 0 & \frac{1}{a_2} & (1-\frac{1}{a_2})\frac{1}{a_3} & \cdots & (1-\frac{1}{a_2})\ldots(1-\frac{1}{a_{j-1}})\frac{1}{a_j} & 1-(\frac{1}{a_2}+(1-\frac{1}{a_2})\frac{1}{a_3}+\cdots)) \\ \hline 0 & 0 & \frac{1}{a_3} & \cdots & \cdots & 1-(\frac{1}{a_3}+\ldots) \\ \hline \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \hline \frac{1}{a_1} & (1-\frac{1}{a_1})\frac{1}{a_2} & (1-\frac{1}{a_1})(1-\frac{1}{a_21})\frac{1}{a_3} & \cdots & (1-\frac{1}{a_1})\ldots(1-\frac{1}{a_{j-1}})\frac{1}{a_j} & 1-(\frac{1}{a_1}+(1-\frac{1}{a_1})\frac{1}{a_2}\ldots)) \end{array}\right)$$

S_0: If $s \equiv 0(\text{mod } a_1)$ then output $(0^{(a_1)})$ and $s = s/a_1$. Check s. If $s \equiv 0(\text{mod } a_1)$ then stay in S_0, else go to S_i, where $1 \le i \le j$.

S_1: If $s \equiv 0(\text{mod } a_2)$ then output $(0^{(a_2)})$ and $s = s/a_2$. Check s. If $s \equiv 0(\text{mod } a_2)$ then stay in S_1, else go to S_i, where $2 \le i \le j$.

...

S_{j-1}: If $s \equiv 0(\text{mod } a_j)$ then output $(0^{(a_j)})$ and $s = s/a_j$. Check s. If $s \equiv 0(\text{mod } a_j)$ then stay in S_{j-1}, else go to S_j.

S_j: If $s' = s(\text{mods } a)$, then output $0^{(a_j)}\ldots 0^{(a_1)}s'^{(a_1)}$ and $s = \frac{s-s'}{a_1^{w_1}\cdots\cdot a_j^{w_j}}$, where there are w_i digits based on a_i for $1 \le i \le j$. Check s. If $s_0 \equiv 0(\text{mod } a_i)(1 \le i \le j)$ then go to S_{i-1}, else stay in S_j.

Since s is random, the probability of $s \equiv 0(\text{mod } a_i)$ is $1/a_i$ for $1 \le i \le j$. When s is not divided by a_1, the probability of $s \equiv 0(\text{mod } a_2)$ is $(1-\frac{1}{a_1})\frac{1}{a_2}$. The integer s will be processed in S_j only if it do not satisfy the conditions of all the other states. Therefore the probability of a number going to state S_j is $1-(\sigma_0+...+\sigma_{j-1})$. We show the probability transition matrix in Table 1. The Markov chain here is aperiodic and irreducible, then there is a unique stationary distribution σ for the state vector. Therefore we have $\sigma \cdot \Pi = \sigma$.

Solving the equations $\sigma_0 + \cdots + \sigma_j = 1$ and $\sigma \cdot \Pi = \sigma$, we get the stationary distribution vector $\sigma = (\frac{1}{a_1-1} \cdot \frac{1}{1+C}, \frac{1}{a_2-1} \cdot \frac{1}{1+C}, \ldots, \frac{1}{a_j-1} \cdot \frac{1}{1+C}, \frac{1}{1+C})$, where $C = \sum_{i=1}^{j} \frac{1}{a_i-1}$.

Assuming that we have m digits in the extended wmbNAF $s = \sum_{i=0}^{m} s_i^{(a_{k_i})}$ and l non-zero digits after λ state transactions. Since we get one digit in state S_i for $0 \le i \le j-1$ and $w_1 + w_2 + \ldots + w_j$ digits in state S_j, $m = \lambda(\sigma_0 + \cdots + \sigma_{j-1} + \sigma_j \cdot (w_1 + \ldots + w_j))$. There is one non-zero digit generated in state S_j and none in other states, hence $l = \lambda(\sigma_j \cdot 1) = \frac{\lambda}{1+C}$. Then the average density of non-zero digits is $d = l/m = \frac{1}{w_1+w_2+\ldots+w_j+C}$. When Algorithm 2 stops, we have a returned value satisfying $a_1^{\sigma_0\lambda} \cdot a_2^{\sigma_1\lambda} \cdot \cdots \cdot a_j^{\sigma_{j-1}\lambda} \cdot (a_1^{w_1} \cdot a_2^{w_2} \cdots a_j^{w_j})^{\sigma_j\lambda} = s \le 2^{n+1} - 1$, which means $\lambda < \frac{n}{(\sigma_0+w_1\sigma_j)log_2a_1+(\sigma_1+w_2\sigma_j)log_2a_2+\cdots+(\sigma_{j-1}+w_j\sigma_j)log_2a_j}$. Therefore the average length of the extended wmbNAF satisfies $m < \lambda(\sigma_0 + \cdots + \sigma_{j-1} + \sigma_j(w_1 + \cdots + w_j)) = \frac{(C+w_1+\cdots+w_j)n}{(\frac{1}{a_1-1}+w_1)log_2a_1+\cdots+(\frac{1}{a_j-1}+w_j)log_2a_j}$. Then the average hamming weight of the extended wmbNAF is approximately $dm = \frac{n}{(\frac{1}{a_1-1}+w_1)log_2a_1+\cdots+(\frac{1}{a_j-1}+w_j)log_2a_j}$. In the m digits, there are $\lambda\sigma_{i-1} + w_i\lambda\sigma_j$ digits in base (a_i), which is $\frac{(\frac{1}{a_i-1}+w_i)n}{(\frac{1}{a_1-1}+w_1\sigma_j)log_2a_1+(\frac{1}{a_2-1}+w_2)log_2a_2+\cdots+(\frac{1}{a_j-1}+w_j)log_2a_j}$ for $1 \le i \le j$. ■

IV. COMPARISON

Now that we have the average Hamming weight of the extended wmbNAF, we compute the average computational cost of the extended wmbNAF method for scalar multiplication in this section. Assume that A, D, T, Q, S separately denote mixed addition, doubling, tripling, quintupling and septupling. When we use the formulas of point operations proposed in [9], [10] for general curves, the computational cost of A, D, T, Q and S are $7M+4S$, $2M+8S$, $6M+10S$, $10M+14S$ and $17M+14S$, where M and S denote the multiplication and square on F_p. When we implement the field operations using software, we have $1S = 0.8M$. Then the cost of doubling, tripling and mixed addition are $8.4M$, $14M$ and $10.2M$.

In Table II, we show the average computational cost of the extended wmbNAF and NAF methods without precomputations over prime field F_p with $|p| = 160$bits. The $(2,3,5)\text{NAF}_{1,1,0}$ method is the fastest one in Table II, which is approximately 9.0% faster than the NAF method without extra storage requirements.

We compare the extended wmbNAF with wNAF methods that need precomputations and extra storage space in Table III. We assume that it is acceptable if the extra storage requirement is less than 10 points. In Table III, we obtain the lowest computational cost when using $(2,3,5)\text{NAF}_{4,1,0}$.

One advantage of the extended wmbNAF method is that the time and space complexity of the recoding algorithm is just the same as the recoding algorithm for NAF. Another advantage of the extended wmbNAF method is that the the window set, which determines the number of doublings and triplings, can adjust to the costs of doubling and tripling. For example, if we work on F_p with $|p| = 160$bits and use bases 2 and 3, the $(2,3)\text{NAF}_{1,1}$ method is faster than the (2,3)NAF method for general curves. But on binary fields, where the doubling is much faster than tripling, the $(2,3)$NAF method is faster. This property is important sometime because there are various elliptic curves and point operation formulas. When we use the curves $y^2 = x^3+x^2+a$ over finite fields of characteristic three [15], where tripling is faster than doubling, the $(2,3)\text{NAF}_{1,3}$ method is the best choice, because more triplings substitute for doublings in the scalar multiplication by making $a = 2^1 3^3$ in Algorithm 2.

TABLE II
PERFORMANCE OF EXTENDED mbNAF AND NAF OVER F_p WITH $|p| = 160$ WITHOUT PRECOMPUTATIONS

Methods	Point operation cost	Total Cost
NAF	160D+53.3A	1887.7M
(2,3)NAF	126.6D+21.1T+42.2A	1789.3M
(2,3)NAF$_{1,1}$	73.1D+54.8T+36.6A	1754.6M
(2,3,5)NAF	109.8D+18.3T+9.1Q+36.6A	1744.8M
(2,3,5)NAF$_{1,1}$	64.5D+48.4T+8.0Q+32.3A	1718.5M
(2,3,5,7)NAF	99.2D+16.5T+8.3Q+5.5S+33.1A	1733.0M

TABLE III
PERFORMANCE OF EXTENDED wmbNAF AND wNAF WITH PRECOMPUTATIONS

Methods	Point operation cost	Total Cost	extra Store(Precomputation)
NAF$_5$(5NAF)	160D+26.7A	1616.3M	7 Points($1D + 7A$)
(2,3)NAF$_{2,1}$	89.3D+44.6T+29.8A	1678.5M	1 Points($1D + 1A$)
(2,3)NAF$_{3,1}$	100.4D+37.6T+25.1A	1625.8M	3 Points($2D + 3A$)
(2,3)NAF$_{4,1}$	108.4D+32.5T+21.7A	1586.9M	7 Points($2D + 7A$)
(2,3)NAF$_{2,2}$	68.9D+57.5T+23.0A	1618.4M	5 Points($2D + 5A$)
(2,3,5)NAF$_{5,0,0}$	130.2D+10.9T+5.4Q+21.7A	1582.1M	7 Points($1D + 7A$)
(2,3,5)NAF$_{4,1,0}$	100.5D+30.2T+5.0Q+20.1A	1578.2M	7 Points($2D + 7A$)

In the extended wmbNAF method, we always have a doubling before the addition if $a_1 = 2$ and $w_1 \neq 0$. Then we can use doubling-addition(denoted by DA) [9] to speed up the implementation further. The cost of doubling-addition is $11M + 7S$, which is 10.8% faster than one doubling and one mixed addition.

We compare the different double-base and multi-base methods in Table IV and Table V. Because it is difficult to formally analyze the average cost of double-base chain methods, we use the data in [6], [12], [13] directly, which was estimated by simulations. In these two tables, the extra storage requirement of the tree-based DB-chain method is in its recoding algorithm. Note that the space for eight scalars is equal to the storage space for four points. In Table V, the (2,3,5)NAF$_{1,1}$ method gains the best improvement without extra storage requirements. When there are enough space for precomputation, we use (2,3,5)NAF$_{4,1}$ to get the fastest method. Even if we recode the scalar using only two bases 2 and 3, we also get the fastest methods (2,3)NAF$_{1,1}$ without extra storage requirements and (2,3)NAF$_{4,1}$ with enough extra storage space.

V. CONCLUSIONS

In this paper, we formally analyze the average complexity of the extended wmbNAF method for scalar multiplication over large prime fields, and compare its performance to other techniques in the literature. The analysis shows that the $(2, 3, 5)$NAF$_{1,1,0}$ method is approximately 9.0% faster than the NAF method without precomputations and extra storage requirements. If we have enough space for precomputations, the $(2, 3, 5)$NAF$_{4,1,0}$ provides the best perfromance. The extended wmbNAF method is the fastest double-base (multi-base) method so far for scalar multiplication. Moreover, the extended wmbNAF method has simple and fast recoding algorithm similar to that of the NAF method.

REFERENCES

[1] N. Koblitz. Elliptic curve cryptosystems. Mathematics of Computation, 48(177):203 – 209, Jan. 1987.

[2] V. S. Miller. Uses of elliptic curves in cryptography. Advances in Cryptology (CRYPTO '85), LNCS 218, pp. 417 – 428, Springer-Verlag, 1986.

[3] Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart. Elliptic Curves in Cryptography. London Mathematical Society Lecture Note Series. Cambridge University Press, Cambridge, 1999.

[4] R. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. Handbook of Elliptic and Hyperelliptic Curve Cryptography, CRC Press, 2005.

[5] Vassil Dimitrov, Laurent Imbert, and Pradeep Kumar Mishra. Efficient and Secure Elliptic Curve Point Multiplication Using Double-Base Chains. Proceedings of ASIACRYPT 2005, LNCS 3788, pp. 59 – 78, Springer-Verlag, 2005.

[6] Vassil Dimitrov, Laurent Imbert, and Pradeep K. Mishra. The Double-Base Number System and Its Application to Elliptic Curve Cryptography. Mathematics Of Computation, 77 (262): 1075 – 1104 , April 2008.

[7] Pradeep Kumar Mishra, Vassil S. Dimitrov. Efficient Quintuple Formulas for Elliptic Curves and Efficient Scalar Multiplication Using Multibase Number Representation, Information Security, LNCS 4779. pp. 390 – 406, Springer-Verlag, 2007.

[8] Doche Christophe, Imbert Laurent. Extended Double-Base Number System with Application to Elliptic Curve Cryptography. Proceedings of INDOCRYPT 2006, LNCS. 4329, pp. 335 – 348, Springer-Verlag, 2006.

[9] P. Longa, and A. Miri. New Multibase Non-Adjacent Form Scalar Multiplication and its Application to Elliptic Curve Cryptosystems. Cryptology ePrint Archive, Report 2008/052, January 2008.

[10] P. Longa, and A. Miri. Fast and Flexible Elliptic Curve Point Arithmetic over Prime Fields. IEEE Transactions on Computers archive, 57(3):289 – 302, March 2008.

[11] M. Ciet, M. Joye, K. Lauter, and P. L. Montgomery. Trading inversions for multiplications in elliptic curve cryptography. Designs, Codes and Cryptography, 39(2):189 – 206, 2006.

[12] C. Doche and L. Imbert, Extended double-base number system with applications to elliptic curve cryptography, Progress in Cryptology, INDOCRYPT'06, LNCS 4329, pp. 335 – 348, Springer-Verlag, 2006.

[13] Christophe Doche and Laurent Habsieger. A Tree-Based Approach for Computing Double-Base Chains, Proceedings of the 13th Australasian conference on Information Security and Privacy, Wollongong, Australia, LNCS 5107, pp. 433 – 446, Springer-Verlag, 2008.

[14] Christophe Doche, David R. Kohel and Francesco Sica. Double-Base Number System for Multi-scalar Multiplications. Advances in Cryptology - EUROCRYPT 2009, LNCS 5479, pp. 502 – 517, Springer-Verlag, 2009.

TABLE IV
NUMBER OF POINT OPERATIONS OF DIFFERENT DOUBLE-BASE AND MULTI-BASE METHODS

	160 bits	200 bits	256 bits	
Methods	Average Computational Cost			Extra Storage
Greedy DB-Chain [13]			150.7D+65.5T+58.0A	0
Tree-based DB-Chain [13]			145.7D+68.6T+52.5A	8 Scalars
SMBR [7]	85D+38T+18Q+31.44A			0
SMBR [7]	84D+36T+16Q+21.14A			7 Points
Extended DB-Chain [8]		118.1D+49.9T+36.8A		1 Points
Extended DB-Chain [8]		117.5D+49.7T+30.7A		3 Points
Extended DB-Chain [8]		117.2D+49.3T+25.9A		8 Points
(2,3)NAF$_{1,1}$	36.5D+54.8T+36.6DA	45.7D+68.5T+45.7DA	58.5D+87.7T+58.5DA	0
(2,3)NAF$_{2,1}$	59.5D+44.6T+29.8DA	74.4D+55.8T+37.2DA	95.2D+71.4T+47.6DA	1 Points
(2,3)NAF$_{3,1}$	75.3D+37.6T+25.1DA	94.0D+47.0T+31.4DA	120.5D+60.2T+40.1DA	3 Points
(2,3)NAF$_{4,1}$	86.7D+32.5T+21.7DA	108.4D+40.7T+27.1DA	138.8D+52.1T+34.7DA	7 Points
(2,3,5)NAF$_{1,1}$	32.2D+48.4T+8.0Q+32.3DA	40.4D+60.5T+10.0Q+40.3DA	51.7D+77.5T+12.9Q+51.6DA	0
(2,3,5)NAF$_{4,1}$	80.4D+30.2T+5.0Q+20.1DA	100.6D+37.7T+6.3Q+25.1DA	128.6D+48.3T+8.0Q+32.2DA	7 Points

TABLE V
NUMBER OF FIELD MULTIPLICATIONS OF DIFFERENT DOUBLE-BASE AND MULTI-BASE METHODS

	160 bits	200 bits	256 bits	
Methods	Number of Point operations			Extra Storage
Greedy DB-Chain [13]			$2774.5M$	0
Tree-based DB-Chain [13]			$2719.8M$	8 Scalars
SMBR [7]	$1948.3M$			0
SMBR [7]	$1764.4M$			7 Points
Extended DB-Chain [8]		$2066.0M$		1 Points
Extended DB-Chain [8]		$1995.9M$		3 Points
Extended DB-Chain [8]		$1938.9M$		8 Points
(2,3)NAF$_{1,1}$	$1681.4M$	$2101.5M$	$2690.3M$	0
(2,3)NAF$_{2,1}$	$1618.9M$	$2023.7M$	$2589.4M$	1 Points
(2,3)NAF$_{3,1}$	$1575.6M$	$1968.8M$	$2520.7M$	3 Points
(2,3)NAF$_{4,1}$	$1543.5M$	$1930.2M$	$2471.3M$	7 Points
(2,3,5)NAF$_{1,1}$	$1653.9M$	$2067.3M$	$2649.3M$	0
(2,3,5)NAF$_{4,1}$	$1537.8M$	$1923.1M$	$2460.6M$	7 Points

[15] N. P. Smart and E. J. Westwood. Point Multiplication on Ordinary Elliptic Curves over Fields of Characteristic Three. Applicable Algebra in Engineering, Communication and Computing. 13(6): 485 – 497, April 2003.

[16] Sangook Moon. A Binary Redundant Scalar Point Multiplication in Secure Elliptic Curve Cryptosystems. International Journal of Network Security, 3(2):132 – 137, Sept. 2006.

[17] O. Haeggstroem. Finite Markov Chains and Algorithmic Applications. Cambridge University Press, 2002.